CHARTGALAXY++: A RICHLY ANNOTATED DATASET FOR CHART UNDERSTANDING AND GENERATION

Anonymous authors
Paper under double-blind review

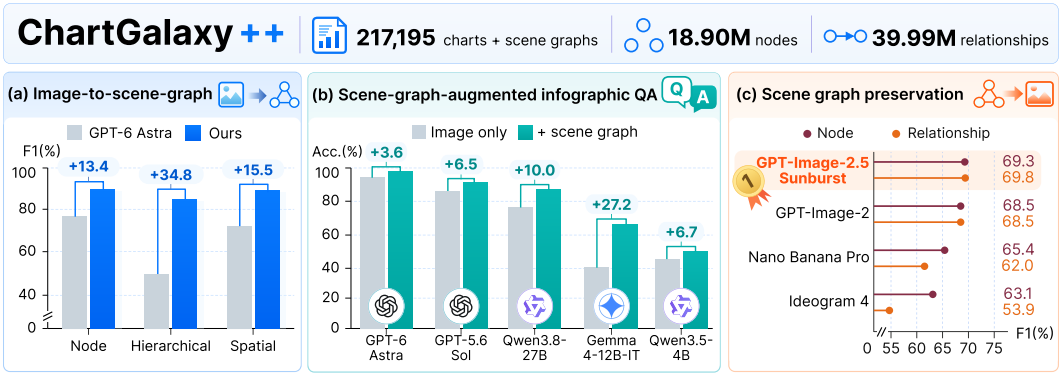


Figure 1: ChartGalaxy++ provides scene graph annotations for 217,195 infographic charts. These annotations support (a) image-to-scene-graph prediction, (b) scene-graph-augmented infographic question answering, and (c) evaluation of scene graph preservation in image generation.

ABSTRACT

Infographic charts communicate data through cohesive compositions of visual elements such as charts, text, and images. Existing infographic chart datasets primarily annotate individual elements with bounding boxes and category labels. Such element-level annotations do not capture hierarchical and spatial relationships, limiting their use for understanding how elements are organized and related within an infographic and for generating well-organized designs. To address this limitation, we introduce ChartGalaxy++, a richly annotated dataset of 217,195 infographic charts. At the core of this dataset is a scene graph representation, in which nodes represent visual elements and semantic groups, and edges encode their hierarchical and spatial relationships. For real infographics, we construct scene graphs by detecting and progressively grouping visual elements; for synthetic ones, we derive them directly from the generation process. We demonstrate the utility of ChartGalaxy++ through three applications: 1) image-to-scene-graph prediction, 2) scene-graph-augmented infographic question answering, and 3) evaluation of scene graph preservation in image generation. Together, these applications highlight the value of ChartGalaxy++ for advancing chart understanding and generation.

Code: https://github.com/ChartGalaxyPP/ChartGalaxyPlusPlus
Data & Dataset Card: https://huggingface.co/datasets/ChartGalaxyPP/ChartGalaxyPlusPlus

1 INTRODUCTION

Infographic charts integrate visual elements, such as charts, text, and images, into cohesive visual designs. By presenting complex data in an accessible and compelling manner, they are widely used in domains such as news reporting, business communication, and public education (Cui et al., 2022; Elaldı & Çifçi, 2021). Despite these benefits, the visual richness and structural complexity of infographic charts pose substantial challenges for foundation models in both understanding and generation. For understanding, even frontier Large Vision-Language Models (LVLMs), such as GPT-5.6 (OpenAI, 2026), still struggle to locate relevant evidence in complex charts, integrate information

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

across elements, and perform reliable multi-step reasoning over the underlying data (Masry et al., 2025; Efat et al., 2026; Xiao et al., 2026). For generation, frontier image-generation models, such as Nano Banana Pro (Google DeepMind, 2025) and GPT-Image-2 (OpenAI, 2026c), can produce visually appealing outputs but still struggle to faithfully represent the underlying data, accurately render visual elements, and organize them into coherent designs (Tang et al., 2026). These limitations suggest that understanding and generating infographic charts require models to capture not only individual elements but also the relationships that organize them into a coherent whole.

Recent studies have made progress toward this goal by constructing large-scale infographic datasets with element-level annotations (Zhu et al., 2026b; Gutflaish et al., 2025; Li et al., 2026b). For example, Zhu et al. (2026b) effectively improve the understanding performance of LVLMs by providing bounding-box annotations for charts. Despite this progress, existing datasets primarily annotate visual elements independently and do not explicitly model the relationships among them. These relationships can be either hierarchical, specifying how individual elements are organized into groups, or spatial, characterizing their relative direction, alignment, overlap, and proximity (Shi et al., 2023). Both provide structural information beyond element-level annotations. Such structural information can, for example, help associate a value label with the correct data mark and preserve their alignment during generation.

To explicitly model these relationships, we introduce a scene graph representation in which leaf nodes represent individual visual elements, internal nodes represent semantic groups, and edges encode hierarchical and spatial relationships. Based on this representation, we construct ChartGalaxy++ by augmenting the real and synthetic infographic charts in ChartGalaxy (Li et al., 2026b) with scene graph annotations. For real infographic charts, we first extract text and non-text elements using HunyuanOCR-1.5 (Li et al., 2026a) and DAD (Zhu et al., 2026a), respectively. We then use Gemini 3.8 Flash (Google DeepMind, 2026a) to annotate element attributes and extract hierarchical relationships by organizing individual elements into semantic groups. Spatial relationships among elements and groups are subsequently derived from their bounding boxes. The resulting annotations are iteratively refined based on errors identified through programmatic checks and LVLM-based verification. For synthetic infographic charts, scene graphs are directly extracted from the generation process, where the underlying element structure and layout information are readily available. In total, ChartGalaxy++ contains 217,195 infographic charts, including 56,744 real ones and 160,451 synthetic ones. Their scene graphs comprise 18.90 million nodes and 39.99 million edges.

We evaluate the utility of ChartGalaxy++ through three applications (Fig. 1). First, we develop an image-to-scene-graph model to predict visual elements and their relationships. Second, we augment infographic question answering (QA) with scene graphs to assess whether explicit structural information improves evidence grounding and reasoning. Third, we evaluate whether frontier image-generation models can preserve the elements and relationships specified by an input scene graph. Together, these applications demonstrate the value of ChartGalaxy++ for developing and evaluating models that extract, use, and preserve infographic structure for understanding and generation.

The main contributions of our work include:

- A scene graph representation for infographic charts that organizes visual elements into semantic groups and models their hierarchical and spatial relationships.
- An automated pipeline for constructing scene graph annotations for both real and synthetic infographic charts.
- A large-scale dataset of 217,195 infographic charts with rich scene graph annotations for structure-aware chart understanding and generation.

2 Related Work

A growing number of datasets provide element-level annotations for plain and infographic charts.

Plain charts present data in a minimal manner using geometric shapes and text. Earlier plain-chart datasets mainly provide bounding boxes and category annotations for these visual elements (Kahou et al., 2017; Methani et al., 2020; Deng et al., 2023; Dou et al., 2024). For example, PlotQA (Methani et al., 2020) contains 224,377 programmatically generated charts with the bounding boxes and categories extracted from the generation process. To support the understanding of chart semantics,

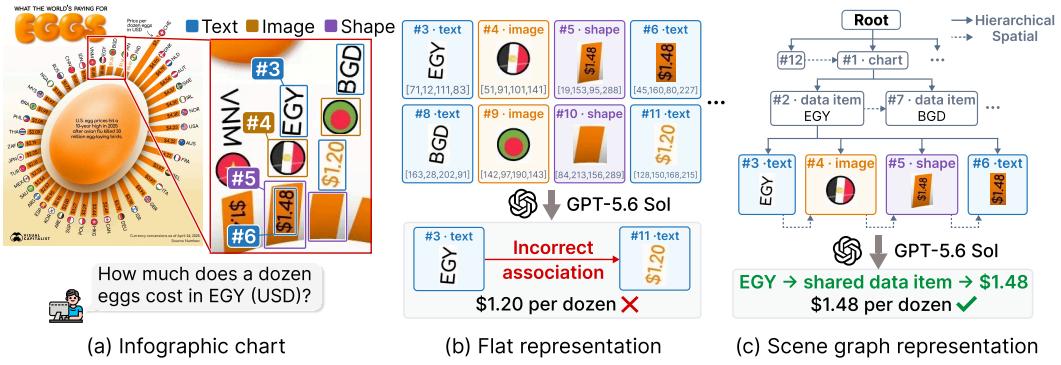


Figure 2: Comparison of flat and scene graph representations for infographic understanding. Flat representations lack explicit structural information, potentially leading to incorrect element associations. Our scene graph representation resolves such ambiguities by explicitly grouping semantically related elements and encoding their hierarchical and spatial relationships.

more recent datasets introduce grounding annotations that identify elements relevant to a given query (Vogel et al., 2025; Suri et al., 2025; Bansal et al., 2025; Ito, 2026; Liu et al., 2026b; Niu et al., 2026) or reasoning annotations that associate intermediate reasoning steps with the involved elements (Xu et al., 2025; Lee et al., 2026). Beyond element-level annotations, several datasets also capture chart structure. VisText (Tang et al., 2023) consists of 12,441 charts programmatically generated using Vega-Lite and provides hierarchical representations based on parent-child relationships. VisAnatomy (Chen et al., 2026a) consists of 942 real-world SVG charts created with over 50 visualization tools, with manually annotated hierarchical element groupings and group layouts. While these datasets provide rich structural annotations for plain charts, they do not capture more diverse visual elements and relationships in infographic charts.

To bridge this gap, several datasets annotate elements in infographic charts (Borkin et al., 2016; Li et al., 2026b; Zhu et al., 2026b). ChartGalaxy (Li et al., 2026b) contains over 1.7 million infographic charts collected online or generated from layout templates, with bounding-box annotations for text, images, and charts. InfoDet (Zhu et al., 2026b) further provides finer-grained annotations for synthetic infographics by additionally labeling 26 categories of chart marks, such as bars, axes, and legends. However, these datasets still treat elements largely in isolation and do not explicitly model how they are grouped and spatially organized. ChartGalaxy++ addresses this limitation by introducing scene-graph-based rich annotations that capture hierarchical and spatial relationships among elements and groups, enabling structure-aware infographic understanding and generation.

3 Scene Graph Representation

Existing infographic-chart representations typically model an image as a flat collection of categorized bounding boxes. While such representations capture individual elements and their locations, they do not encode an important characteristic of infographics: visual elements are compositionally organized into semantic groups, and these elements and groups are spatially organized into coherent layouts. Without explicitly modeling this structure, models must infer element associations and layout organization from isolated elements, making it difficult to distinguish visually adjacent but semantically unrelated elements. For example, as shown in Fig. 2(b), the flat representation provides no explicit indication of which price is associated with EGY. Consequently, GPT-5.6 Sol may associate EGY with the spatially nearby price of BGD ($1.20), rather than its corresponding price ($1.48).

To address this issue, we introduce a scene graph representation that transforms a flat collection of visual elements into a structured composition of semantic groups. Specifically, it organizes visual elements into semantic groups, such as data items, series, axes, and legends, recursively organizes these groups into a hierarchy, and encodes spatial relationships among elements and groups. Formally, we represent each infographic as a scene graph $\mathcal{G} = (V, E)$, where V and E denote the node and edge sets, respectively. The node set $V = V_\ell \cup V_i$ contains leaf and internal nodes. Each leaf node $v \in V_\ell$ corresponds to a visual element, such as a text block, image, or geometric shape. Each internal node

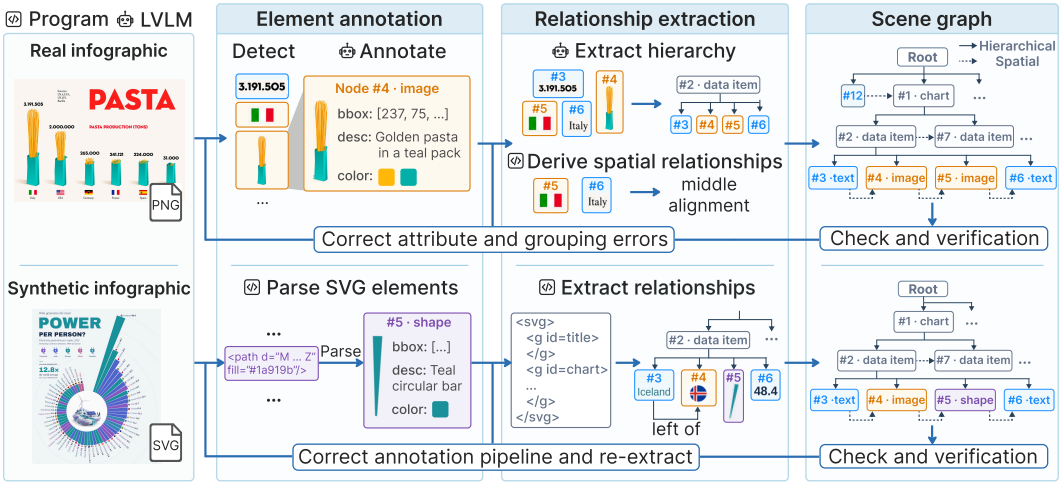


Figure 3: Overview of the scene graph annotation pipeline.

$v \in V_i$ represents a group containing multiple elements and, recursively, other groups. For example, in Fig. 2(c), the country label, flag, bar mark, and value label associated with Egypt are grouped under the same internal node because they jointly encode one data item. This explicit grouping distinguishes the value "$1.48" associated with EGY from nearby values belonging to other countries.

Each node $v \in V$ stores four common attributes: type, bounding box, color palette, and description. Text nodes additionally store their text content. The edge set $E = E_h \cup E_s$ contains hierarchical and spatial-relationship edges. Hierarchical edges E_h encode parent-child relationships between nodes, forming a nested semantic hierarchy. Spatial-relationship edges E_s encode geometric relationships between node pairs, including relative direction, alignment, overlap, and proximity. The detailed schema for nodes and edges is provided in Appendix A.

4 Dataset Construction

ChartGalaxy++ builds on ChartGalaxy (Li et al., 2026b), which contains both real infographic charts collected from the web and synthetic ones programmatically generated from layout templates. We augment both types with scene graph annotations that jointly capture visual elements, as well as their hierarchical and spatial relationships.

4.1 Annotation Pipeline

Real infographic charts are available only as images, while synthetic ones retain their underlying element structure and layout information from the generation process. We therefore use a dedicated annotation pipeline for each source (Fig. 3).

Real infographic chart annotation Given a real infographic chart, our scene graph annotation pipeline consists of two stages: element annotation and relationship extraction. We then iteratively refine the resulting scene graph by identifying errors and correcting them in the corresponding stage.

Element annotation. We first extract the content and bounding boxes of text elements using HunyuanOCR-1.5 (Li et al., 2026a), and the bounding boxes of non-text elements using DAD (Zhu et al., 2026a). Given the image and extracted bounding boxes, we then use Gemini 3.8 Flash (Google DeepMind, 2026a) to annotate each element's type, color palette, and description. These annotations define the leaf nodes of the scene graph.

Relationship extraction. Extracting hierarchical and spatial relationships relies on different types of information. Hierarchical relationships depend on visible region boundaries, relative placement, and how elements jointly convey information. We therefore use Gemini 3.8 Flash to integrate such information and group elements into a hierarchy. Each group is represented as an internal node

connected to its children by hierarchical edges. Spatial relationships can instead be determined directly from node bounding boxes, so we derive them programmatically. Together, the annotated nodes and extracted relationships form the initial scene graph.

Iterative refinement. We validate the initial scene graph using programmatic checks and LVLM-based verification. The programmatic check detects invalid attribute values, unresolved node references, and violations of hierarchy constraints. The LVLM-based verification uses Gemini 3.8 Flash to compare the scene graph with the original infographic to identify missing elements, incorrect attributes, and incorrect groupings. The detected errors are corrected in the corresponding annotation stages, and the updated scene graph is re-verified until no further errors are identified.

The implementation details for real infographic annotation are provided in Appendix B.

Synthetic infographic chart annotation ChartGalaxy generates synthetic infographic charts as SVG files, which explicitly encode visual elements and their layout information. We therefore extract the scene graph directly from these files.

Element annotation. We parse the SVG to extract text, images, and geometric shapes. For each element, we compute its bounding box from its position, size, and transformations encoded in the SVG, and extract other attributes from the corresponding SVG tags and metadata. These extracted elements form the leaf nodes of the scene graph.

Relationship extraction. We derive internal nodes and hierarchical edges from the grouping structure of the SVG. We then derive spatial relationships from the bounding boxes of the resulting nodes.

Iterative refinement. The annotated nodes and extracted relationships form the scene graph. We validate the resulting scene graphs using the programmatic check and LVLM-based verification. If either identifies a potential error, we manually inspect the corresponding infographic and update the annotation pipeline as needed.

4.2 Data Statistics

ChartGalaxy++ comprises 217,195 infographic charts, including 56,744 real and 160,451 synthetic ones, each paired with its scene graph annotation. These scene graph annotations include 15,591,656 elements: 6,044,031 text blocks, 1,107,791 images, and 8,439,834 geometric shapes, averaging 71.8 elements per chart. They further encode 18,680,288 hierarchical and 21,309,060 spatial relationships. On average, each scene graph consists of 15.2 groups and spans 4.5 hierarchy levels. The detailed statistics are provided in Appendix C. For consistent evaluation, we use 1,000 infographic charts as the test set and the remainder as the training set. The test set annotations are manually verified. A three-expert audit of the test set shows 99.81% element precision, 99.78% recall, and 10 incorrect parent assignments among 99,010 non-root nodes (Appendix B.3).

5 Experiments

We evaluate the utility of ChartGalaxy++ through three applications: image-to-scene-graph prediction, scene-graph-augmented QA, and scene graph preservation in image generation.

5.1 Image-to-Scene-Graph Prediction

5.1.1 Experimental Setup

Training setup We build an image-to-scene-graph model by supervised fine-tuning Qwen3.5-4B (Qwen Team, 2026a) on the ChartGalaxy++ training set. Since spatial relationships can be easily derived from bounding boxes using deterministic rules, we train the model to predict only the remaining scene graph annotations and compute spatial relationships programmatically from its predicted bounding boxes. We perform full-parameter fine-tuning for $100,000$ steps with a batch size of 32 and a learning rate of $5e{-}5$. Additional training details are provided in Appendix D.1.

Baseline models We compare our model with ten representative open-source and proprietary LVLMs. Specifically, we select seven open-source models (Molmo2-8B (Allen Institute for AI, 2026), InternVL3.5-8B (OpenGVLab, 2025), Gemma 4-12B-IT (Google DeepMind, 2026b),

Table 1: Comparison of our image-to-scene-graph model with representative LVLMs.

	Node-level						Edge-level	
Model	Overall	Type	Box	Text	Color	Description	Hierarchical	Spatial
Molmo2-8B	8.7	67.3	32.0	43.2	65.0	34.9	4.0	0.9
InternVL3.5-8B	27.7	79.6	34.6	75.1	68.7	36.2	9.8	10.0
Gemma 4-12B-IT	45.8	94.1	51.1	91.8	77.5	53.7	18.1	29.5
Qwen3.5-4B	27.8	90.5	25.3	86.4	80.1	51.3	9.0	10.6
Qwen3.8-27B	60.0	95.6	65.7	96.5	87.3	54.8	27.3	47.1
Qwen3.8-Flash-Next-FP8	60.9	95.6	68.8	96.2	86.3	54.3	32.3	50.2
GLM-5.3-Flash	63.1	93.5	66.5	95.4	88.6	60.3	34.9	50.1
Gemini 3.8 Flash	70.1	95.2	83.8	96.5	90.3	64.1	47.7	61.5
GPT-5.6 Sol	69.8	95.4	77.5	94.9	91.6	59.5	45.8	63.3
GPT-6 Astra	76.0	95.7	87.4	95.6	91.9	59.0	50.3	72.5
Ours	**89.4**	**98.5**	**95.4**	**97.2**	**96.6**	**90.2**	**85.1**	**88.0**

Qwen3.5-4B (Qwen Team, 2026a), Qwen3.8-27B (Qwen Team, 2026b), Qwen3.8-Flash-Next-FP8 (Qwen Team, 2026c), and GLM-5.3-Flash (Z.ai, 2026a)) and three proprietary ones (Gemini 3.8 Flash (Google DeepMind, 2026a), GPT-5.6 Sol (OpenAI, 2026), and GPT-6 Astra (OpenAI, 2026b)). For each image, all models receive a prompt that specifies the scene graph schema and asks the model to predict the scene graph excluding spatial relationships. We then derive the spatial relationships programmatically from the predicted bounding boxes. Additional details of the baseline settings are provided in Appendix D.2.

Evaluation metrics We evaluate predicted scene graphs against their ground-truth annotations at two levels: nodes and edges.

Node-level metrics. For each pair of predicted and ground-truth nodes, we compute similarities for the following attributes: 1) type, measured by exact matching; 2) bounding box, measured by generalized intersection over union (GIoU) (Rezatofighi et al., 2019), which considers both overlap and spatial separation; 3) text content, for text nodes, using normalized Levenshtein similarity (Levenshtein, 1966), which reflects character-level edit distance; 4) color palette, measured by matching colors using CIEDE2000 color difference (Luo et al., 2001) and assessing whether predicted colors remain within and cover the ground-truth palette; and 5) description, evaluated by GPT-5.6 Luna (OpenAI, 2026a) based on semantic similarity. The similarity between two nodes is computed by averaging the applicable attribute similarities. Implementation details of these metrics are provided in Appendix D.3.

Following Jiang et al. (2026), we evaluate overall node-level performance using a similarity-weighted F1 score. We first use the Hungarian algorithm to find the one-to-one matching that maximizes the total similarity between predicted and ground-truth nodes. We then compute precision as the summed similarities of matched pairs divided by the number of predicted nodes, and recall as the same sum divided by the number of ground-truth nodes. Their harmonic mean is used as the F1 score, and the mean similarity of each attribute over the matched pairs is also reported.

Edge-level metrics. Following Wu et al. (2021), we evaluate edge-level performance using the F1 score of predicted edges. A predicted edge is a true positive if both endpoint nodes are matched and the same relationship exists between their corresponding ground-truth nodes. All other predicted edges are false positives, and ground-truth edges without a corresponding true positive are false negatives. We report F1 scores separately for hierarchical and spatial edges.

5.1.2 Results and analysis

Table 1 compares our image-to-scene-graph model with the baselines on the ChartGalaxy++ test set. Our model achieves the highest scores across all eight metrics, with overall node-level, hierarchical, and spatial F1 scores of 89.4, 85.1, and 88.0, respectively. These scores exceed those of GPT-6 Astra, the strongest baseline, by 13.4, 34.8, and 15.5, respectively. Qualitative analysis further highlights two common baseline errors that our model better handles: inaccurate element localization and incorrect grouping. For localization, Fig. 4(a) shows that GPT-6 Astra shifts the bounding box of one Google "G" logo upward, whereas our model localizes it accurately. For grouping, Fig. 4(b) shows

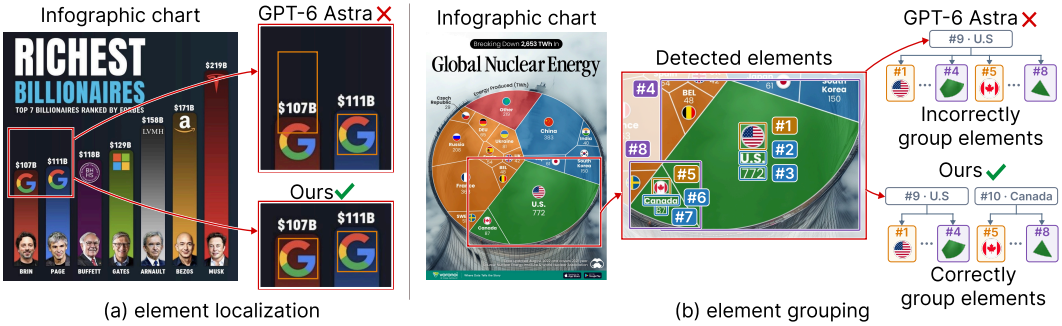


Figure 4: Qualitative comparison of our image-to-scene-graph model and GPT-6 Astra.

that GPT-6 Astra incorrectly groups elements associated with the U.S. and Canada under the same parent, whereas our model correctly assigns them to separate country groups. These results show that ChartGalaxy++ provides effective supervision for image-to-scene-graph prediction.

5.2 Scene-Graph-Augmented Infographic QA

5.2.1 Benchmark Construction

To better assess the structural reasoning capabilities of current LVLMs, we construct a QA benchmark that requires resolving associations among visual elements. The benchmark is built in two stages: manual question construction and model-assisted expansion. In the first stage, we select complex infographic charts from ChartGalaxy++ and manually create question-answer pairs that require 1) **resolving correspondences among elements within a group**, such as matching labels and values to their corresponding data marks; 2) **reasoning across groups**, such as identifying which country has a given value in one year and retrieving its value in another year; or 3) **reasoning about relationships between elements**, such as determining which nodes are connected or share a boundary. This process results in 200 question-answer pairs. In the second stage, GPT-6 Astra (OpenAI, 2026b) uses these pairs to derive an agent skill for identifying additional infographic charts and generating question-answer pairs that require similar forms of structural reasoning. Each generated question-answer pair is verified by three authors. After verification, the resulting benchmark contains 1,266 questions across 1,000 infographic charts. We evaluate model accuracy using the answer-type-specific scoring rules from InfoChartQA (Xie et al., 2025). Example question-answer pairs and details of the scoring are provided in Appendices E.1 and E.2.

5.2.2 Experimental Setup

Scene-graph-augmented prompting In scene-graph-augmented prompting, we provide scene graph annotations alongside the chart image and question to facilitate evidence identification and reasoning. Specifically, we instruct the model to use the annotations to 1) identify relevant labels, data marks, and values; 2) establish their correspondences; and 3) perform the necessary comparison or calculation to derive the answer. To reduce unnecessary detail, we summarize each node's description into a concise label and remove the color palettes before providing the scene graph to the model.

Baseline prompting methods We compare our method with two baselines: 1) **Direct prompting**, which asks the model to answer the question directly from the chart image; and 2) **Evidence-guided prompting**, which follows the same identification and reasoning process as our method but without access to the scene graph annotations. Detailed prompts are provided in Appendix E.3.

5.2.3 Results and Analysis

Table 2 compares QA accuracy under scene-graph-augmented prompting and the two baselines across the same ten LVLMs as in Sec. 5.1. Scene-graph-augmented prompting consistently outperforms both baselines across all ten models. Compared with evidence-guided prompting, it improves accuracy by an average of 8.9%. The largest improvement is observed for Gemma 4-12B-IT, whose accuracy increases by 27.2%, from 40.2% to 67.4%. These results show that explicitly providing

Table 2: Comparison of QA accuracy under different prompting methods.

	Open-source models							Proprietary models		
Prompting	Molmo2 8B	InternVL3.5 8B	Gemma 4 12B-IT	Qwen3.5 4B	Qwen3.8 27B	Qwen3.8 Flash-Next FP8	GLM-5.3 Flash	Gemini 3.8 Flash	GPT-5.6 Sol	GPT-6 Astra
Direct	37.5	31.9	32.4	35.9	75.1	72.7	74.0	90.0	81.4	95.9
Evidence-guided	40.4	33.8	40.2	46.7	76.9	76.0	79.0	88.5	85.1	94.0
Scene-graph-augmented	**46.7**	**45.3**	**67.4**	**53.4**	**86.9**	**86.1**	**83.9**	**90.6**	**91.6**	**97.6**

structural information can improve QA performance by reducing the need for models to infer element associations directly from the image.

To examine the contributions of each scene graph component, we conduct an ablation study on Qwen3.8-27B. Starting from evidence-guided prompting without scene graph annotations, we progressively add node types and bounding boxes, relationships, concise labels, text without digits, and text with digits. As shown in Table 3, accuracy improves as additional scene graph components are introduced. The largest gain comes from adding text with digits, indicating that explicit numerical text is particularly useful for answering questions that involve data values. We further evaluate two alternative representations: adding color palettes and replacing concise labels with full descriptions. Both slightly reduce accuracy, supporting our choice to remove color palettes and use concise labels that focus on what each element represents. Removing relationships also reduces accuracy, suggesting that explicit relationship information can help the model establish correspondences among elements.

Table 3: QA accuracy on Qwen3.8-27B with different scene graph inputs.

Annotations	Accuracy
No scene graph	76.9
+ node types & bounding boxes	77.8
+ relationships	78.9
+ labels	80.2
+ text without digits	81.5
+ text with digits (best)	**86.9**
Best + color palettes	86.5
Best, labels $\rightarrow$ descriptions	86.5
Best - relationships	85.6

5.3 Scene Graph Preservation in Image Generation

5.3.1 Experimental Setup

Models We evaluate how well existing open-source and proprietary image-generation models preserve specified elements and relationships when generating from scene graphs. Specifically, we select seven representative open-source models (Qwen-Image-2512 (Wu et al., 2025), ERNIE-Image-Turbo (Liu et al., 2026a), SenseNova-U1 Infographic V3 (Diao et al., 2026), GLM-Image (Z.ai, 2026b), HiDream-O1-Image (Cai et al., 2026), Ideogram 4 (Ideogram, 2026), and Qwen-Image-SP (Chen et al., 2026b)) and four proprietary ones (GPT-Image-2 (OpenAI, 2026c), GPT-Image-2.5 Sunburst (OpenAI, 2026d), Nano Banana Pro (Google DeepMind, 2025), and Nano Banana 2 (Google DeepMind, 2026c)). For models that support JSON inputs (Qwen-Image-SP and Ideogram 4), we convert the scene graph into the format required by each model. For the remaining models, we provide the scene graph directly in the text prompt. Details of the model inputs are provided in Appendix F.

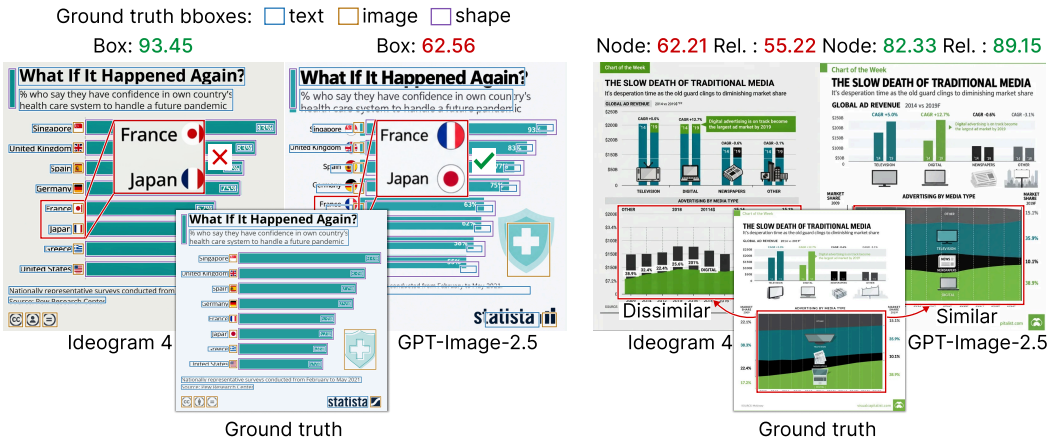
Evaluation metrics To measure scene graph preservation, we extract a scene graph from each generated image using our annotation pipeline and compare it with the input scene graph. We use the same node- and edge-level metrics as in the image-to-scene-graph model evaluation.

5.3.2 Results and analysis

Table 4 compares scene graph preservation of the evaluated models on the ChartGalaxy++ test set. Overall, the proprietary models perform better than the open-source ones, particularly in preserving hierarchical and spatial relationships. GPT-Image-2.5 Sunburst achieves the highest overall node-level, hierarchical, and spatial F1 scores of 69.3, 75.4, and 64.2, respectively. Among the open-source models, Ideogram 4 performs the best, achieving an overall node-level F1 score of 63.1 and even the highest bounding-box similarity of 47.4 across all models. However, its hierarchical and spatial F1 scores of 58.5 and 49.2 remain substantially below those of GPT-Image-2.5 Sunburst. This behavior is consistent with Ideogram 4's JSON input representation, which specifies individual elements and their bounding boxes but does not explicitly encode relationships among them (Ideogram, 2026).

Table 4: Comparison of scene graph preservation across representative image-generation models.

	Node-level						Edge-level	
Model	Overall	Type	Box	Text	Color	Description	Hierarchical	Spatial
HiDream-O1-Image	18.7	92.1	32.8	20.8	75.9	40.1	16.6	6.6
GLM-Image	35.9	83.7	25.0	19.1	74.3	35.9	20.8	14.0
Qwen-Image-2512	40.1	82.5	28.2	15.9	77.8	36.4	28.0	19.1
Qwen-Image-SP	44.9	85.9	31.7	24.9	78.7	38.1	35.2	25.3
ERNIE-Image-Turbo	45.3	89.6	30.3	34.2	78.6	45.1	36.9	23.2
SenseNova-U1 Infographic V3	48.5	92.5	30.9	60.7	77.9	54.6	40.7	26.0
Ideogram 4	63.1	93.0	**47.4**	67.4	80.5	57.8	58.5	49.2
Nano Banana 2	61.7	93.5	38.1	85.6	80.9	66.4	64.8	48.2
Nano Banana Pro	65.4	94.1	39.2	**89.4**	85.0	68.4	70.4	53.5
GPT-Image-2	68.5	95.3	39.0	86.2	88.2	69.4	74.4	62.5
GPT-Image-2.5 Sunburst	**69.3**	**95.6**	43.1	86.2	**89.1**	**70.3**	**75.4**	**64.2**



(a) Infographics with fewer elements and a common chart (b) infographics with more elements and a rare chart

Figure 5: Qualitative comparison of Ideogram 4 and GPT-Image-2.5 Sunburst.

To further examine the gap between the strongest open-source and proprietary models, we qualitatively compare Ideogram 4 and GPT-Image-2.5 Sunburst. We observe that, for infographics with fewer elements and common chart types, Ideogram 4 better preserves the specified layout but is more prone to errors in element content, whereas GPT-Image-2.5 Sunburst better preserves element content but exhibits larger layout deviations. For example, in Fig. 5(a), Ideogram 4 closely follows the specified layout but misrenders the French and Japanese flags, while GPT-Image-2.5 Sunburst renders them correctly but at inaccurate positions. For infographics with more elements and rare chart types, GPT-Image-2.5 Sunburst shows a clearer advantage in preserving both elements and their relationships (Fig. 5(b)).

6 Conclusion

In this paper, we present ChartGalaxy++, a richly annotated dataset for structure-aware chart understanding and generation. It includes 217,195 real and synthetic infographic charts, each paired with a scene graph that captures the chart's visual elements, as well as their hierarchical and spatial relationships. Through three applications, we demonstrate the utility of ChartGalaxy++ for developing and evaluating models that extract, use, and preserve the structure of infographic charts. Despite these promising results, several directions remain for future work. First, representing element locations with segmentation masks or vector paths, rather than bounding boxes, could better capture element shapes. Second, directly training image-generation models with scene graph annotations could further explore their value for structure-aware generation.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic datasets, implementing methods, and cleaning and reformatting datasets. We have not used generative AI tools for proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, supporting qualitative and thematic data analysis, or interpreting results. Developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools for brainstorming and sourcing or searching for information. We have reviewed all AI-assisted work. LLM-generated code for extracting scene graph annotations was manually reviewed by the authors for validity. We also verified the sources of all information provided by LLMs. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We carefully consider copyright and licensing issues in the construction and release of ChartGalaxy++. To avoid redistributing third-party content, we release only the source URLs of the real infographic chart images together with our newly constructed scene graph annotations, and do not host or redistribute the original images. We have reviewed the licenses and reuse conditions of the source platforms and confirmed that our use and release protocol is consistent with these terms. Further details on licenses, reuse conditions, and attribution requirements are provided in Appendix C.1.

REFERENCES

Allen Institute for AI. Molmo2-8B. Model card, 2026. URL `https://huggingface.co/allenai/Molmo2-8B`.

Aniruddh Bansal, Davit Soselia, Dang Nguyen, and Tianyi Zhou. ChartAB: A benchmark for chart grounding and dense alignment. *arXiv preprint arXiv:2510.26781*, 2025.

Michelle A Borkin, Zoya Bylinskii, Nam Wook Kim, Constance May Bainbridge, Chelsea S Yeh, Daniel Borkin, Hanspeter Pfister, and Aude Oliva. Beyond memorability: Visualization recognition and recall. *IEEE transactions on visualization and computer graphics*, 22(1):519–528, 2016.

Qi Cai, Jingwen Chen, Chengmin Gao, Zijian Gong, Yehao Li, Yingwei Pan, et al. Hidream-o1-image: A natively unified image generative foundation model with pixel-level unified transformer. *arXiv preprint arXiv:2605.11061*, 2026.

Chen Chen, Hannah K. Bako, Peihong Yu, John Hooker, Jeffrey Joyal, Simon C. Wang, Samuel Kim, Jessica Wu, Aoxue Ding, Lara Sandeep, Alex Chen, Chayanika Sinha, and Zhicheng Liu. VisAnatomy: An SVG chart corpus with fine-grained semantic labels. *IEEE Transactions on Visualization and Computer Graphics*, 32(1):560–570, 2026a. doi: 10.1109/TVCG.2025.3634263.

Zilong Chen, Chaorui Deng, Kunchang Li, Hongyi Yuan, and Haoqi Fan. Scaling properties of text conditioning in visual generation. *arXiv preprint arXiv:2607.29679*, 2026b.

Weiwei Cui, Jinpeng Wang, He Huang, Yun Wang, Chin-Yew Lin, Haidong Zhang, and Dongmei Zhang. A mixed-initiative approach to reusing infographic charts. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):173–183, 2022. doi: 10.1109/TVCG.2021.3114856.

Dazhen Deng, Yihong Wu, Xinhuan Shu, Jiang Wu, Siwei Fu, Weiwei Cui, and Yingcai Wu. Visimages: A fine-grained expert-annotated visualization dataset. *IEEE Transactions on Visualization and Computer Graphics*, 29(7):3298–3311, 2023.

Haiwen Diao, Penghao Wu, Hanming Deng, Jiahao Wang, Shihao Bai, Silei Wu, et al. Sensenova-u1: Unifying multimodal understanding and generation with neo-unify architecture. *arXiv preprint arXiv:2605.12500*, 2026.

Shuguang Dou, Xinyang Jiang, Lu Liu, Lu Ying, Caihua Shan, Yifei Shen, Xuanyi Dong, Yun Wang, Dongsheng Li, and Cairong Zhao. Hierarchically recognizing vector graphics and a new chart-based vector graphics dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):7556–7573, 2024.

Azher Ahmed Efat, Seok Hwan Song, and Wallapak Tavanapong. Beyond single plots: A benchmark for question answering on multi-charts. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 35374–35411, 2026.

Şenel Elaldı and Taner Çifçi. The effectiveness of using infographics on academic achievement: A meta-analysis and a meta-thematic analysis. *Journal of Pedagogical Research*, 5(4):92–118, 2021. doi: 10.33902/jpr.2021473498.

Google DeepMind. Introducing Nano Banana Pro, 2025. URL https://blog.google/innovation-and-ai/products/nano-banana-pro/. Accessed: 2026-08-29.

Google DeepMind. Gemini 3.8 Flash. Model card, 2026a. URL https://deepmind.google/models/model-cards/gemini-3-8-flash/.

Google DeepMind. Gemma 4-12B-IT. Model card, 2026b. URL https://huggingface.co/google/gemma-4-12B-it.

Google DeepMind. Nano banana 2. https://deepmind.google/models/gemini-image/flash/, 2026c.

Eyal Gutflaish, Eliran Kachlon, Hezi Zisman, Tal Hacham, Nimrod Sarid, Alexander Visheratin, Saar Huberman, Gal Davidi, Guy Bukchin, Kfir Goldberg, and Ron Mokady. Generating an image from 1,000 words: Enhancing text-to-image with structured captions. *arXiv preprint arXiv:2511.06876*, 2025.

Ideogram. Ideogram 4.0, 2026. URL https://ideogram.ai/models/4.0/. Accessed: 2026-08-31.

Koichiro Ito. Phrase-grounding-aware supervised fine-tuning for chart recognition via side-masked attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9501–9511, 2026.

Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 25472–25483, 2026.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. FigureQA: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*, 2017.

Eunsoo Lee, Jeongwoo Lee, Minki Hong, Jangho Choi, and Jihie Kim. Visdot: Enhancing visual reasoning through human-like interpretation grounding and decomposition of thought. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 610–640, 2026.

Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

Gengluo Li, Xingyu Wan, Shangpin Peng, Weinong Wang, Hao Feng, Yongkun Du, et al. HunyuanOCR-1.5: Making lightweight OCR VLMs faster and better. *arXiv preprint arXiv:2607.04884*, 2026a. URL https://arxiv.org/abs/2607.04884.

Zhen Li, Duan Li, Yukai Guo, Xinyuan Guo, Bowen Li, Lanxi Xiao, Shenyu Qiao, Jiashu Chen, Zijian Wu, Hui Zhang, Xinhuan Shu, and Shixia Liu. ChartGalaxy: A dataset for infographic chart understanding and generation. In *Proceedings of the International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=P4lFbvZ4HH.

Jiaxiang Liu, Zhida Feng, Pengyu Zou, Zhenyu Qian, Tianrui Zhu, Jun Xia, et al. Ernie-image technical report. *arXiv preprint arXiv:2605.25347*, 2026a.

Zhuoming Liu, Xiaofeng Gao, Feiyang Niu, Qiaozi Gao, Liu Liu, and Robinson Piramuthu. Start: Spatial and textual learning for chart understanding. In *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 8146–8156. IEEE, 2026b.

M Ronnier Luo, Guihua Cui, and Bryan Rigg. The development of the cie 2000 colour-difference formula: Ciede2000. *Color Research & Application*, 26(5):340–350, 2001.

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, et al. ChartQAPro: A more diverse and challenging benchmark for chart question answering. *arXiv preprint arXiv:2504.05506*, 2025.

Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. PlotQA: Reasoning over scientific plots. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1527–1536, 2020.

Tianhao Niu, Ziyu Han, Xuan Dong, Qingfu Zhu, and Wanxiang Che. Chartreg++: Towards benchmarking and improving chart referring expression grounding under diverse referring clues and multi-target referring. *arXiv preprint arXiv:2605.07415*, 2026.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition, 2026. URL https://openai.com/index/gpt-5-6/. Accessed: 2026-08-29.

OpenAI. GPT-5.6 Luna. Model documentation, 2026a. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna.

OpenAI. GPT-6 Astra. Model documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-6-astra.

OpenAI. Gpt-image-2. https://developers.openai.com/api/docs/models/gpt-image-2, 2026c.

OpenAI. Gpt-image-2.5 sunburst. https://developers.openai.com/api/docs/models/gpt-image-2.5-sunburst, 2026d.

OpenGVLab. InternVL3.5-8B. Model card, 2025. URL https://huggingface.co/OpenGVLab/InternVL3_5-8B.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026a.

Qwen Team. Qwen3.8-27B. Model card, 2026b. URL https://huggingface.co/Qwen/Qwen3.8-27B.

Qwen Team. Qwen3.8-Flash-Next-FP8. Model card, 2026c. URL https://huggingface.co/Qwen/Qwen3.8-Flash-Next-FP8.

Miao Rang, Zhenni Bi, Chuanjian Liu, Yunhe Wang, and Kai Han. An empirical study of scaling law for scene text recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2024.

Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Felix Richter. Are you not entertained? Statista, December 2022. December 12. https://www.statista.com/chart/22392/global-revenue-of-selected-entertainment-industry-sectors/. Accessed: 2026-09-24.

Danqing Shi, Weiwei Cui, Danqing Huang, Haidong Zhang, and Nan Cao. Reverse-engineering information presentations: recovering hierarchical grouping from layouts of visual elements. *Visual Intelligence*, 1(1):9, 2023. doi: 10.1007/S44267-023-00010-1.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A Rossi, and Dinesh Manocha. Chartlens: Fine-grained visual attribution in charts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22447–22462, 2025.

Benny Tang, Angie Boggust, and Arvind Satyanarayan. VisText: A benchmark for semantically rich chart captioning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 7268–7298, 2023.

Yinghao Tang, Xueding Liu, Boyuan Zhang, Tingfeng Lan, Yupeng Xie, Jiale Lao, et al. IGenBench: Benchmarking the reliability of text-to-infographic generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 36919–36954, 2026.

Alexander Vogel, Omar Moured, Yufan Chen, Jiaming Zhang, and Rainer Stiefelhagen. Refchartqa: Grounding visual answer on chart images through instruction tuning. In *International Conference on Document Analysis and Recognition*, pp. 523–537. Springer, 2025.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

Jason Wu, Xiaoyi Zhang, Jeff Nichols, and Jeffrey P Bigham. Screen parsing: Towards reverse engineering of ui models from screenshots. In *The Annual ACM Symposium on User Interface Software and Technology*, pp. 470–483, 2021.

Ziyan Xiao, Yinghao Zhu, Wenting Zhang, Heaju Kim, and Lequan Yu. LongChart VQA: A comprehensive benchmark for MLLMs with complex multi-chart reasoning. *arXiv preprint arXiv:2608.01328*, 2026.

Tianchi Xie, Minzhi Lin, Mengchen Liu, Yilin Ye, Changjian Chen, and Shixia Liu. Infochartqa: A benchmark for multimodal question answering on infographic charts. *arXiv preprint arXiv:2505.19028*, 2025.

Zhengzhuo Xu, SiNan Du, Yiyan Qi, Siwen Lu, Chengjin Xu, Chun Yuan, and Jian Guo. Chartpoint: Guiding mllms with grounding reflection for chart reasoning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 426–436. IEEE, 2025.

Z.ai. GLM-5.3-Flash. Model card, 2026a. URL https://huggingface.co/zai-org/GLM-5.3-Flash.

Z.ai. Glm-image. https://z.ai/blog/glm-image, 2026b.

Jiangning Zhu, Bowen Li, Shenyu Qiao, Yima Gu, Zhao Zhang, Yuhui Yuan, and Shixia Liu. Detect Anything in Graphic Design: Element-Level Rewards for Autoregressive Detection. *arXiv preprint arXiv:2609.07072*, 2026a. URL https://arxiv.org/abs/2609.07072.

Jiangning Zhu, Yuxing Zhou, Zheng Wang, Juntao Yao, Yima Gu, Yuhui Yuan, and Shixia Liu. InfoDet: A dataset for infographic element detection. In *Proceedings of the International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=Wj0Sc9WBHZ.

A SCENE GRAPH SCHEMA

This section specifies the node attributes, hierarchical relationships, and spatial relationships introduced in Sec. 3, together with their JSON representation. The hierarchy records which elements jointly form semantic components, while spatial relationships describe the geometry of their bounding boxes.

A.1 NODE ATTRIBUTE SPECIFICATION

The field `node_kind` distinguishes individual visual elements (`element`) from semantic groups (`group`). An element has one of three types: `text` for a visible text region, `image` for an image object such as an icon, photograph, or illustration, and `shape` for a geometric primitive such as a mark, line, or border. Group types identify the semantic components described below. The corresponding type is stored in `element_type` or `group_type`, respectively.

Every stored node has a bounding box, color palette, and description, in addition to its type and structural identifiers (Table 5). Bounding boxes use integer coordinates in the order $[y_0, x_0, y_1, x_1]$, corresponding to the top, left, bottom, and right boundaries. The image origin is at the top-left corner, and horizontal and vertical coordinates are normalized independently to $[0, 1000]$ using the image width and height. Boxes have positive width and height: $x_0 < x_1$ and $y_0 < y_1$. This convention applies to both elements and groups.

Table 5: Node fields in the JSON representation. Fields marked "element" or "group" apply only to that node kind.

Field	Applies to	Definition
`label`	All	Unique node identifier within the image, combining its type and one-based position in the node array, e.g., `<text 4>`.
`parent`	All	Label of the direct parent group, or the reserved token `ROOT`.
`node_kind`	All	`element` or `group`.
`element_type`	Element	`text`, `image`, or `shape`.
`group_type`	Group	One of the seven semantic group types.
`bbox`	All	Four integers $[y_0, x_0, y_1, x_1]$ in the range $[0, 1000]$.
`color_palette`	All	Array of distinct dominant colors, represented as `#RRGGBB` or `#RRGGBBAA`; the final two digits encode alpha.
`desc`	All	Nonempty natural-language description of the node's visible appearance, composition, and functional role.
`text`	Text element	Exact visible wording, preserving case, spelling, numbers, punctuation, and line breaks.

The `text` field preserves the visible string. The `desc` field describes the node's visible appearance, composition, and functional role within the infographic. For example, the description of a `shape` can indicate that it serves as a data mark, a legend key, or a decorative border. For text regions, it describes typography, arrangement, and function, while the exact wording is stored in `text`. Group descriptions explain the component formed by their members; group nodes do not carry a separate text-transcription field.

A.2 HIERARCHICAL RELATIONSHIPS

Hierarchical edges point from a parent group to its children. In JSON, each child stores its incoming hierarchical edge through its `parent` field; a separate list of hierarchical edges is therefore unnecessary. Every stored node has exactly one parent, which must be an existing group or `ROOT`, and every parent chain terminates at `ROOT` without a cycle. The hierarchy thus forms a rooted tree when the implicit root is included. `ROOT` represents the whole image and is not stored as a node object or counted as an additional ordinary group type.

Groups identify components whose members jointly convey information (Table 6). For example, a `data_item` can associate a mark with its category and value labels, while a `legend_item` associates a legend key with its label. A group can contain individual elements and other groups,

allowing a chart to contain data items, axes, and a legend, with the legend further containing its entries. Parent membership expresses semantic composition; bounding-box containment or spatial proximity alone does not establish this relationship.

Table 6: Group types and their intended meanings.

Group type	Meaning
`chart`	A chart component containing data marks or data items and their associated reading references.
`axis`	An axis and its associated components, such as ticks, tick labels, and an axis title.
`legend`	A legend containing its entries and any associated title.
`legend_item`	One legend entry associating a visual key with its label.
`data_item`	Elements jointly representing one observation, such as a mark with its category label, value, or identifying icon.
`series`	Elements representing one data series, including its marks and associated labels.
`panel`	A coherent layout region, such as a heading block or a distinct component of a multi-part infographic.

A.3 Spatial Relationships

Spatial relationships are computed from node bounding boxes. Let $B_a = (t_a, l_a, b_a, r_a)$ and $B_b = (t_b, l_b, b_b, r_b)$ denote two boxes in normalized coordinates, and let W and H be the original image width and height in pixels. Each spatial record specifies its type through `relation_type` and its endpoints through `from_label` and `to_label`. Following Sec. 3, we describe four types of spatial relationships: direction, alignment, overlap, and proximity. The selection rule below determines which node pairs are included in the serialized representation.

Direction. Direction describes where the first box lies relative to the second, using eight values: `top`, `bottom`, `left`, `right`, `top-left`, `top-right`, `bottom-left`, and `bottom-right`. We determine the horizontal and vertical components using the conditions

$$\begin{aligned} \text{left}: \quad & r_a \le l_b, \qquad \text{right}: \quad l_a \ge r_b, \\ \text{top}: \quad & b_a \le t_b, \quad \text{bottom}: \quad t_a \ge b_b. \end{aligned} \tag{1}$$

When only a horizontal or vertical condition holds, it gives one of the four cardinal directions. When both hold, their combination gives a diagonal direction, written with the vertical component first and joined by a hyphen, e.g., `top-left`. Directional relationships are serialized with `relation_type` set to `relative_position`. The `directions` array stores a single value, e.g., `["top-left"]`. If neither axis is separated, no directional record is stored for the pair.

Alignment. `aligned_with` compares corresponding left, right, top, and bottom boundaries and horizontal and vertical centers. Writing $c_x = (l + r)/2$ and $c_y = (t + b)/2$, an alignment basis $q \in \{l, r, t, b, c_x, c_y\}$ is satisfied when $|q_a - q_b| \le 5$ in normalized coordinates. The `basis` array contains all satisfied bases, encoded as `left`, `right`, `top`, `bottom`, `center_x`, or `center_y`. The tolerance corresponds to 0.5% of image width for horizontal coordinates and 0.5% of image height for vertical coordinates.

Overlap. For a selected pair, `bbox_overlaps` is present exactly when the two boxes have a positive-area intersection:

$$\min(r_a, r_b) > \max(l_a, l_b) \quad \text{and} \quad \min(b_a, b_b) > \max(t_a, t_b). \tag{2}$$

Containment is included as an overlap case, while touching boundaries alone do not satisfy this condition.

Proximity. Proximity records geometric adjacency through the Boolean field `is_near` and also stores the distance between the bounding boxes. The Boolean predicate is independent of the sibling-based pair selection used for sparse storage below. The box distance is the shortest gap between the

two rectangular regions, with a smaller value indicating closer boxes. We first convert horizontal and vertical gaps to pixels:

$$\begin{aligned}\Delta_x &= \frac{W}{1000}\max(0, l_a - r_b, l_b - r_a),\\ \Delta_y &= \frac{H}{1000}\max(0, t_a - b_b, t_b - b_a),\\ d_{\text{px}} &= \sqrt{\Delta_x^2 + \Delta_y^2}, \qquad d_{\text{ratio}} = \frac{d_{\text{px}}}{\sqrt{W^2 + H^2}}.\end{aligned} \tag{3}$$

Proximity is serialized with `relation_type` set to `bbox_distance`. Alongside `is_near`, its fields `distance_px` and `distance_ratio` store these values rounded to six and ten decimal places, respectively. Overlapping or touching boxes have zero distance. Using W and H separately preserves the original image aspect ratio when computing the pixel distance. For valid positive-area boxes, let $A_a = WH(r_a - l_a)(b_a - t_a)/10^6$ and define A_b analogously, both in square pixels. The Boolean predicate is

$$\begin{aligned}\texttt{is_near}(a, b) = \neg\texttt{bbox_overlaps}(a, b)\\ \wedge [d_{\text{px}} \le 0.06\min(W, H)]\\ \wedge [d_{\text{px}} \le \min(\sqrt{A_a}, \sqrt{A_b})].\end{aligned} \tag{4}$$

Thus positive-area overlap, including containment, is not near, whereas exact edge or corner contact is near. Separated pairs must satisfy both the image-scale and node-scale gap limits. Missing or invalid boxes do not create near relationships.

Selection of spatial relationships. Exhaustive pairwise annotation would repeat layout information across many unrelated or redundant node pairs. We therefore select relationships between direct siblings at each level of the semantic hierarchy, including children of `ROOT`. Element-level relationships describe the arrangement within a component, while relationships between sibling groups describe the arrangement of components. Parent membership remains represented by the hierarchical edges.

Within each sibling set, we retain the union of three kinds of node pairs: nearest-sibling candidates, consecutive nodes within aligned sets, and overlapping nodes. Nearest-sibling candidates retain local arrangements; alignment chains describe rows, columns, and shared boundaries without requiring every aligned pair; overlap preserves intersections such as an illustration placed inside a bar. For each retained pair, we record all applicable direction, alignment, and overlap information, together with its proximity flag and box distance. Each relationship type is stored once per unordered pair; direction uses the specified source and target order without a duplicate inverse record. Unrecorded pairs are not treated as negative labels, and their geometry remains available from the node bounding boxes.

For nearest-sibling candidates, each node nominates its two nearest siblings by box distance, and each group additionally nominates its two nearest sibling groups. When fewer than two eligible nodes exist, all are nominated; ties at the distance cutoff are retained. A pair is a nearest-sibling candidate if either endpoint nominates the other; nomination does not determine its Boolean proximity label. For each alignment basis, we sort siblings by that coordinate and greedily form consecutive sets whose coordinate span is at most 5 normalized units. Within each set, consecutive nodes ordered by their centers along the perpendicular axis define the alignment chain; ties are resolved by the aligned coordinate and then node label. The bounded coordinate span prevents approximate alignment from accumulating across a chain. All positive-area sibling overlaps are retained, and the union is deduplicated without an image-wide record cap. These storage rules are separate from the metric computation described in Appendix D.

A.4 JSON Format Example

The JSON representation of a scene graph contains four top-level fields. The `high_level_description` field describes the overall image composition, and `style_description` records its visual style. The `compositional_deconstruction` object contains a background description and the node array, whose `parent` references specify the hierarchy. The `relations` array contains spatial records with the type-specific fields defined above. Image dimensions are read from the image itself and are not included in this JSON representation.

The image-to-scene-graph training target omits `relations`, since these relationships can be computed from the predicted bounding boxes.

Figure 6 uses a published Statista infographic comparing four entertainment sectors (Richter, 2022). The original 1200×1200 image is reproduced alongside an enlarged view of its first data item with manually annotated boxes. This data-item group associates the bar, gaming illustration, revenue label, category label, and category-coverage text. Solid boxes denote elements, and the dashed box denotes their parent group. Numbers match the numeric suffixes of the JSON node labels. All box coordinates are normalized relative to the full source image.

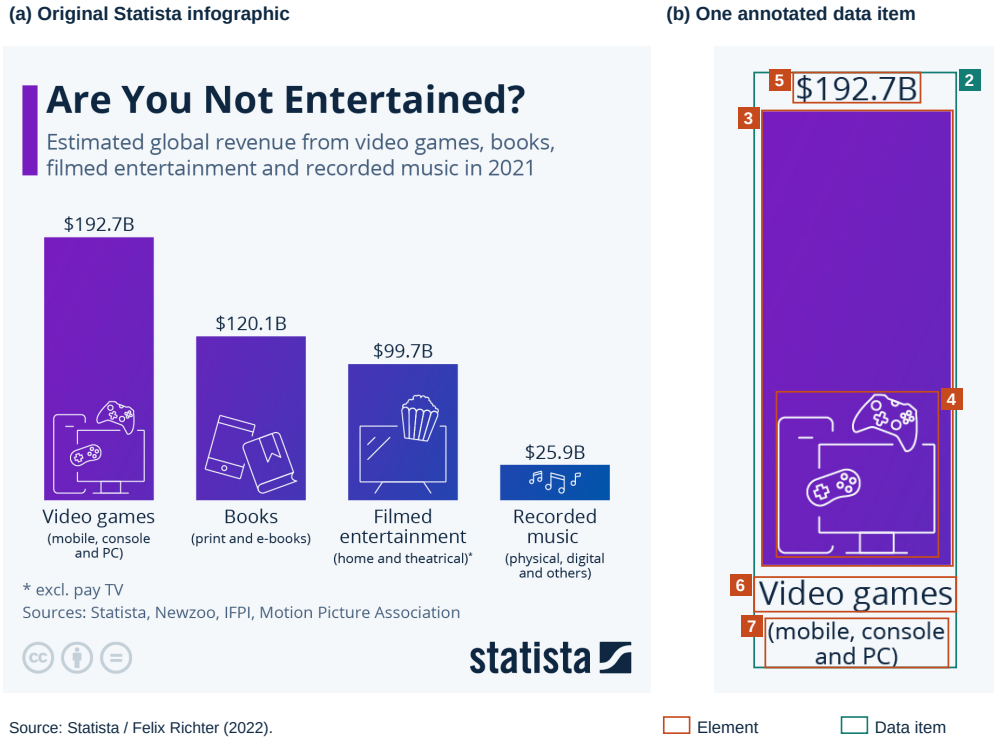


Figure 6: A real infographic and its scene-graph annotation. (a) Original four-bar Statista infographic by Felix Richter (2022), with its source and attribution retained. (b) Enlarged view of the video-games data item: group 2 contains bar 3, illustration 4, revenue label 5, category label 6, and explanatory text 7. The annotation boxes and numbered tags are added for this example and correspond to the JSON excerpt below.

The excerpt retains the chart node, one complete data-item subtree, and the spatial records for two pairs within that data item. The other data items, heading, footer, and spatial records are omitted for readability. The illustration and bar overlap, giving a box distance of zero. The bar is `bottom` relative to its revenue label; their horizontal centers are aligned, and the normalized boxes give a gap of 8.4 pixels. The overlapping illustration–bar pair has `is_near: false`; the separated bar–label pair satisfies both gap limits and has `is_near: true`.

```
{
  "high_level_description":
    "Four entertainment sectors compared by revenue.",
  "style_description": {
    "aesthetics": "Clean editorial infographic.",
    "lighting": "Uniform.", "photo": null, "medium": "graphic_design",
    "art_style": "Flat vector with gradient bars.",
    "color_palette": ["#F3F8FC", "#102A43", "#7420BB",
                      "#0756AC", "#FFFFFF"]
  },
  "compositional_deconstruction": {
    "background": "Pale blue background.",
    "nodes": [
```

```
      {"label": "<chart 1>", "parent": "ROOT",
       "node_kind": "group", "group_type": "chart",
       "bbox": [261, 56, 826, 938],
       "color_palette": ["#102A43", "#7420BB", "#0756AC"],
       "desc": "Four bars with category icons and labels."},
      {"label": "<data_item 2>", "parent": "<chart 1>",
       "node_kind": "group", "group_type": "data_item",
       "bbox": [261, 56, 794, 237],
       "color_palette": ["#7420BB", "#FFFFFF", "#102A43"],
       "desc": "Bar, icon, value, and category labels."},
      {"label": "<shape 3>", "parent": "<data_item 2>",
       "node_kind": "element", "element_type": "shape",
       "bbox": [295, 63, 703, 234],
       "color_palette": ["#7420BB"],
       "desc": "Vertical gradient bar encoding revenue."},
      {"label": "<image 4>", "parent": "<data_item 2>",
       "node_kind": "element", "element_type": "image",
       "bbox": [547, 76, 695, 221],
       "color_palette": ["#FFFFFF"],
       "desc": "White gaming devices used as a category icon."},
      {"label": "<text 5>", "parent": "<data_item 2>",
       "node_kind": "element", "element_type": "text",
       "bbox": [261, 91, 288, 205],
       "color_palette": ["#102A43"],
       "desc": "Navy revenue label above the bar.",
       "text": "$192.7B"},
      {"label": "<text 6>", "parent": "<data_item 2>",
       "node_kind": "element", "element_type": "text",
       "bbox": [713, 56, 744, 237],
       "color_palette": ["#102A43"],
       "desc": "Navy category label below the bar.",
       "text": "Video games"},
      {"label": "<text 7>", "parent": "<data_item 2>",
       "node_kind": "element", "element_type": "text",
       "bbox": [750, 66, 794, 230],
       "color_palette": ["#102A43"],
       "desc": "Navy text specifying category coverage.",
       "text": "(mobile, console\nand PC)"}
    ]
  },
  "relations": [
    {"relation_type": "aligned_with",
     "from_label": "<image 4>", "to_label": "<shape 3>",
     "basis": ["center_x"]},
    {"relation_type": "bbox_overlaps",
     "from_label": "<image 4>", "to_label": "<shape 3>"},
    {"relation_type": "bbox_distance",
     "from_label": "<image 4>", "to_label": "<shape 3>",
     "is_near": false, "distance_px": 0.0,
     "distance_ratio": 0.0},
    {"relation_type": "relative_position",
     "from_label": "<shape 3>", "to_label": "<text 5>",
     "directions": ["bottom"]},
    {"relation_type": "aligned_with",
     "from_label": "<shape 3>", "to_label": "<text 5>",
     "basis": ["center_x"]},
    {"relation_type": "bbox_distance",
     "from_label": "<shape 3>", "to_label": "<text 5>",
     "is_near": true, "distance_px": 8.4,
     "distance_ratio": 0.0049497475}
  ]
}
```

B Annotation Pipelines and Quality Assessment

B.1 Real Infographic Annotation

Following Sec. 4.1, the image-based pipeline constructs a scene graph through element annotation and relationship extraction, followed by iterative refinement. Element annotation identifies the visible objects and their attributes; relationship extraction organizes these elements into a semantic hierarchy and computes their spatial relationships. Table 7 summarizes the information passed between stages. We describe validation in Appendix B.2 and provide the prompt templates in Appendix B.4. The same image-based procedure applies when annotating generated infographics; it uses the image being annotated, without access to a paired reference image or its scene graph.

Element annotation. HunyuanOCR-1.5 provides text content and bounding boxes, while DAD provides candidate bounding boxes for non-text elements. We normalize these boxes to the $[y_0, x_0, y_1, x_1]$ convention in Appendix A.1. Given the original image, contextual crops, and localized candidates, Gemini 3.8 Flash annotates each element's type, color palette, and description, while retaining the supplied text and geometry at this stage. The description includes the element's functional role, such as a data mark, category label, legend key, or decoration. The resulting elements form the leaf nodes used by relationship extraction; visual errors in localization or attributes are handled by the refinement stage.

Table 7: Inputs and outputs of the image-based annotation pipeline. Hierarchical relationships are inferred from the image and annotated elements; spatial relationships are computed from node geometry.

Stage	Input	Output
Element annotation	Image, crops, and localized candidates	Leaf nodes with type, box, palette, description, and text where applicable.
Hierarchical relationship extraction	Image and annotated elements	Semantic groups and parent–child assignments.
Spatial relationship extraction	Element and group boxes	Direction, alignment, overlap, and proximity relationships.
Iterative refinement	Image, current graph, and validation findings	Corrected annotations, rechecked graph, and remaining unresolved findings.

Hierarchical relationship extraction. Gemini 3.8 Flash receives the original image and the annotated elements and identifies components whose members jointly convey information. The model uses visible region boundaries, relative placement, and semantic associations to identify chart groups and associate marks with their labels and values. For example, a country name, flag, bar, and value label can jointly represent one data item. Shared components, such as axes and legends, are represented by the corresponding group types in Appendix A.

The grouping prompt specifies the intended annotation granularity, including what constitutes a data item or continuous series and how shared chart context is represented. These conventions make group meanings consistent across annotations, while the model determines the memberships from each image. The element inventory is fixed during this stage, and the model returns a structured plan specifying semantic groups and element assignments. The program converts that plan into internal nodes and hierarchical edges, with one direct parent per element. Group boxes enclose their descendant elements, group palettes combine the descendants' colors, and group descriptions summarize the resulting components.

Spatial relationship extraction. We compute direction, alignment, overlap, and proximity from node bounding boxes using the definitions and relationship-selection rules in Appendix A.3. This computation applies to the elements and groups in the constructed hierarchy and requires no additional model request. After refinement changes node geometry or group membership, the affected group boxes and spatial relationships are recomputed.

Model and request configuration. Table 8 gives the model settings for element annotation, relationship extraction, and refinement. Gemini requests use JSON response mode; temperature and top-p are left unspecified in the request. Output schemas are checked by the program, including when they are also passed through the API's structured-output parameter. Spatial relationships are computed programmatically, so they require no additional prompt.

Table 8: Model settings for the image-based annotation pipeline. Token limits are per request.

Component	Setting
HunyuanOCR-1.5	`spotting_json`; temperature 0, top-p 1, repetition penalty 1.08; at most 32,768 output tokens.
DAD	Fixed layer-localization instruction; temperature 0, top-p 0.9, top-k 20; deployed output limit 16,384 tokens.
Image and crop inputs	Up to 3×3 ownership cores; grid size depends on image dimensions and candidate count, with a 900-pixel size threshold and 80-pixel context padding.
Element attributes / scene description	Gemini 3.8 Flash; at most 64 candidates per attribute request; LOW thinking; at most 32,768 output tokens.
Localization recovery / crop correction	Gemini 3.8 Flash; LOW thinking; at most 32,768 output tokens.
Whole-image review	Gemini 3.8 Flash; MEDIUM thinking; at most 32,768 output tokens; at most two audit rounds in the recorded extraction configuration.
Semantic grouping	Gemini 3.8 Flash; LOW thinking; 16,384 output tokens, increased to 32,768 when there are more than 350 elements.
Grouping review	Gemini 3.8 Flash; MEDIUM thinking; at most 16,384 output tokens; initial construction/review budget of three rounds.

B.2 Validation and Refinement

We validate the graph structurally and compare its annotations with the source image. Detected errors are routed to element annotation or relationship extraction, and the updated graph is checked again. The two forms of validation address different requirements: a valid graph structure and a faithful representation of visible content.

Programmatic checks. The program verifies required attributes, coordinate ranges, unique node identifiers, valid parent references, and an acyclic hierarchy terminating at `ROOT`. It also checks that the grouping plan accounts for every supplied element exactly once. Responses that fail these checks receive the validation findings as feedback for correction.

LVLM-based verification. Gemini 3.8 Flash compares the graph with the original image to identify missing or extra elements, incorrect attributes or boxes, and incorrect groupings. Whole-image inspection checks the complete scene, while contextual crops support closer inspection of individual regions. For hierarchical relationships, a separate review request examines the image, fixed elements, and proposed grouping plan under the same annotation conventions. It assesses chart boundaries, the completeness of data items or series, and element membership; the program verifies that the review covers every proposed unit.

Iterative correction. Element-level findings are returned to the annotation stage, which produces additions, removals, or updates while preserving unaffected nodes. Grouping findings are returned to relationship extraction to revise the affected assignments. After an element correction, the affected grouping is rebuilt and reviewed, and group geometry and spatial relationships are recomputed. The updated graph must pass structural checks and image-based verification again. The process stops when verification identifies no further actionable errors. The prompt templates for these requests are provided in Appendix B.4.

B.3 Annotation Quality Assessment

The scene graph annotations are assessed by three co-authors, denoted Expert A, Expert B, and Expert C.

Evaluation Dataset. The training and test splits are annotated using the same pipeline. Human verification serves both to correct the test annotations for reliable evaluation and to assess the quality of the shared annotation pipeline by measuring errors before human correction. We manually verify the scene graph annotations in our held-out test set, comprising 500 real and 500 synthetic images across three complexity levels (Table 9). Complexity counts element and group nodes, including the root, before audit corrections. We divide charts into low (≤ 50 nodes), medium (51–91 nodes), and high (≥ 92 nodes) complexity, with cutoffs based on the full dataset's node-count distribution. For quality assessment, the test set contains 99,010 non-root nodes: 82,042 elements and 16,968 groups.

Table 9: Quality-assessment sample by scene graph complexity.

Complexity	Images
Low (≤ 50 nodes)	191
Medium (51–91 nodes)	344
High (≥ 92 nodes)	465
Total	1000

Review system and procedure. We developed an annotation review system that links the source image, bounding-box overlays, scene graph hierarchy, and node attributes. It supports zoomed image inspection, node-level assessments and corrections, separate reviewer records, and comparison of disagreements for adjudication. The experts use programmatic image-inspection tools and structured review records to carry out this workflow.

Experts A and B independently inspect the whole image for missing elements, then check each annotated node for text or numeric errors, incorrect boxes, semantic description errors, and incorrect parent–child relationships. Descriptions include the element's functional role. After both reviews are complete, Expert C compares their findings with the image, resolves disagreements, and checks for mistakes shared by both reviewers. A finding is accepted only when supported by visible evidence; otherwise, it is rejected as unsupported or an acceptable alternative, or retained as unresolved. Equivalent grouping choices, small box-padding differences, color approximations, and paraphrases do not count as errors.

Assessment measures. We summarize findings at the element or node level, using the relevant annotations as denominators. For element completeness, precision is M/N_{orig} and recall is M/N_{ref}: N_{orig} counts original elements, M counts retained original elements, and N_{ref} equals M plus confirmed missing elements. Element correspondence follows the objects identified during review; accepted additions and removals account for omissions and extra elements. For localization, we compute mean intersection-over-union (IoU) between corresponding original and adjudicated element boxes. The reference incorporates supported corrections while preserving acceptable original representations; boxes without a supported reference are excluded from IoU.

We count transcription errors among retained text nodes, and numeric transcription errors among those containing digits. Semantic description errors and incorrect assignments are counted over non-root nodes and their parent links, respectively. Each affected node counts once per measure; a node can have several problems, while dependent updates to ancestor summaries are not counted as additional defects. Unresolved judgments remain separate from confirmed errors.

Assessment results. Table 10 summarizes the node-level assessment. The audit gives an element precision of 99.81% and a recall of 99.78%. Localization corrections affect 645 of 82,042 element boxes; mean IoU is 0.9976. Transcription errors affect 134/34,242 retained text nodes (0.39%). Semantic descriptions or functional roles require correction in 3,053/99,010 nodes (3.08%).

Description and role errors often concern an object's meaning even when its box and text are correct. For example, one chart describes 98 decorative dots as data observations, while 222 nodes across four

Table 10: Node-level assessment before audit corrections. Error rows report affected nodes / eligible nodes. Numeric transcription errors are a subset of text errors. Precision, recall, and IoU are computed against the expert-adjudicated reference.

Measure	Result
Non-root nodes	99,010
Element nodes	82,042
Element precision	99.81%
Element recall	99.78%
Mean box IoU	0.9976
Text transcription errors	134/34,242
Numeric transcription errors	27/21,856
Description / role errors	3,053/99,010
Incorrect parent assignments	10/99,010

charts describe monetary values, percentages, or ratios as ranks. Some group-summary errors arise from fixed templates and summaries left unchanged after child-node corrections. An axis-description template mentions an axis title without checking whether one is present; in a year-mismatch case, a child's description was corrected from 1997 to 1999, but its parent summary retained 1997. The count therefore reflects repeated semantic mistakes across individual nodes, with a single misinterpretation affecting many nodes in one chart. These findings motivate checking descriptions against the actual visual encoding and checking group summaries against their children; wording preferences alone are excluded.

Overall, the audit results support the high quality and reliability of annotations in the test set, particularly in element completeness, localization, and text transcription.

B.4 Prompt Templates

The following reusable templates describe the annotation and refinement requests in Appendices B.1 and B.2. The grouping guidelines define a consistent annotation granularity across images; the model uses the image and element attributes to assign elements to those semantic components. The templates also specify the structured responses needed to construct and validate the graph. Node fields are aligned to the schema in Appendix A. Blue names in double braces denote inputs inserted for each request, such as the candidate inventory, current graph, or response schema. Schema placeholders specify the required JSON fields; they do not contain reference annotations. Shared instructions and grouping guidelines are displayed once and reused by the stages indicated below.

Text and non-text localization. The OCR model receives the image and Prompt B.1, shown as an English translation of the Chinese instruction. The detector receives the image and Prompt B.2. Their XYXY outputs are converted to the YXYX coordinate convention before attribute annotation.

Prompt B.1. Text localization and recognition (English translation)

```
Detect and recognize all text lines in the image, in reading order
from top to bottom and left to right. Output a JSON array. Each item
must contain "box": [xmin, ymin, xmax, ymax], with coordinates
normalized to [0, 1000], and "text": "recognized text content". Output
the JSON array directly without any additional explanation.
```

Prompt B.2. Non-text candidate localization

```
Locate the moderately detailed layers in this design from background
to foreground. Return bbox and type per layer, where t means text and
v means visual. Response format: x1,y1,x2,y2,t;x1,y1,x2,y2,v;
```

Element and scene attributes. Prompt B.3 precedes the attribute, scene-description, localization-recovery, crop-correction, and whole-image-review requests. Attribute annotation receives the original image, a contextual crop, and the localized candidates; it returns attributes and uncertainties while retaining the supplied geometry and text. Scene-description annotation receives the original image and returns the global descriptions and background.

Prompt B.3. Shared visual annotation instructions

```
Annotate original visible pixels, including generated misspellings and
pseudo-text. Never infer missing words from common sense. Use
[illegible] for unreadable spans and record source_illegible findings.
Each distinct text label, bar, stacked segment, flag and icon is an
individual element. Do not collapse separately localizable objects
into one element. During grouping, preserve the supplied elements and
apply the shared-mark rule. Opposite-end labels on a bar are separate
text nodes even when OCR merged them into one line. Do not fragment a
single continuous word or a single recognizable icon into
glyphs/primitives. Keep titles, axis labels, legends, keys, data
marks, backgrounds and footer objects distinct. Tool proposals may be
duplicated, merged, wrong or absent; inspect the pixels independently.
All boxes are tight [y0,x0,y1,x1] integers 0..1000 with positive area.
Do not guess glyphs. Do not include relations, encodings, image
dimensions, versions or audit fields in the graph. Use unique
temporary <type number> labels for new nodes; preserve existing labels
in patches. Group boxes enclose members. Choose parent by actual
belonging, not proximity alone. No human has verified these results.
Descriptions describe visible appearance and functional role; exact
wording belongs in text. Do not output separate role or role_family
fields. Uncertainties are objects {kind,issue,bbox}: kind is
source_illegible, coverage, grouping or geometry. Every uncertainty
must identify the actual unresolved issue and its image region.
```

Prompt B.4. Element attribute annotation

```
ATTRIBUTE ANNOTATION of already localized elements. Return exactly
{attributes,uncertainties}. Every supplied candidate_id MUST occur
exactly once in attributes, including difficult or apparently
duplicated proposals. Each entry has ONLY candidate_id, element_type,
desc, color_palette. Text localization sources ocr/recovery_text MUST
have element_type=text. For non-text candidates choose shape
(geometric marks) or image (recognizable depictions). Use the original
image and localized region to describe appearance, state its
functional role in desc, and give separate #RRGGBB palette entries. No
node-label references in desc. Do not emit bbox, text, groups,
parents, additional instances or replacement IDs in this stage. Their
existing geometry and OCR/recovered text are retained by the program.
If a localization is wrong, merged, duplicated or the text is
incorrect, record a concrete geometry/coverage finding in FULL-IMAGE
normalized coordinates for visual correction. First image is the
original; second is a contextual crop. The supplied boxes remain
FULL-IMAGE normalized YXYX. These are attribute requests, not a new
detection pass.

FULL IMAGE SIZE: {{IMAGE_SIZE}}
TILE PIXEL XYXY: {{TILE}}
LOCALIZED ELEMENTS: {{CANDIDATES}}
```

Prompt B.5. Scene and background descriptions

```
Describe the overall image and canvas background. Return exactly
{high_level_description,style_description,background,uncertainties},
following the supplied schema. Do not reference node labels: element
and group identities belong to later steps. This request does not
create or replace any localized element or hierarchy.

SCENE ATTRIBUTE SCHEMA: {{SCENE_SCHEMA}}
```

Hierarchical relationship extraction. The grouping request receives the image and a fixed inventory of elements, excluding previous groups and parent assignments. The inventory assigns local `E` identifiers to elements; the returned plan assigns chart and observation identifiers and declares each element's owner. The program compiles the plan into the group types, parent links, and group attributes described in Appendix B.1. Both grouping construction and independent grouping review insert the full policy in Prompt B.6 into `GROUPING_POLICY`. The construction request uses Prompt B.7.

Prompt B.6. Shared semantic grouping policy

```
# Semantic units, version 1.6

Reconstruct a deterministic semantic hierarchy from this image and its
immutable elements. Neither its previous groups nor another rendering
is available. This is an annotation convention, not a change to the
evaluation metric. Do not move, rewrite, split, merge, delete or
duplicate an element.

## First identify the measurement unit

Charts are spatially coherent plots: a shared coordinate frame, a
common set of categories compared using the same measurement, or one
part-to-whole whole. Do not split adjacent comparable bars into
separate charts because each has a separate caption, stack palette or
legend. Do split spatially separated panels with independent
measurements/encodings. A title or explanation card is not a chart.
Multiple series on shared axes belong to one chart.

Build **complete semantic data units** before assigning labels. The
compiler creates chart -> unit -> elements. No optional series,
bubble, segment, icon or 3D-face wrappers are possible under a unit.

- Bar/column/lollipop/pictorial bars: one unit per category position.
For a stacked bar, ALL colored segments form ONE whole bar and ONE
unit. A segment is not a separate category. For grouped bars, all
series at the same category position form one unit. Attach the
category label, values, flag/logo identifying the category, and
secondary metric badges directly to that same unit. All faces of a 3D
bar or all pictograms representing one total belong together.
- Histogram: one unit per bin. Waterfall: one unit per step. Do not
merge bins.
- Pie/donut: one unit per sector, including its directly associated
label, leader and value. A donut's central total is chart context, not
a sector.
- Scatter/bubble: one unit per observation, with its own label. Color
alone is never grounds to group unrelated observations. A primary
bubble is the mark, not an additional nested badge.
- Dumbbell/slope/paired-circle comparisons: one unit per entity across
the compared conditions. BOTH endpoint marks, their values and the
connecting line/ribbon form ONE unit. Do not treat endpoints as
independent bubble observations. Shared condition/year labels are
```

Prompt B.6. Shared semantic grouping policy (continued)

```
chart context.
- Line/area/radar: one unit per continuous series (use series native
group type). The connected line/area, its points, inline labels and
series-specific values belong together. Shared axes, gridlines and
legends are chart context. Percentage-change arrows/triangles beside
points, endpoint icons and local explanatory callouts are annotations
of that SAME continuous series and MUST join it directly; they are not
separate units or separate series, even if they have their own
explanation in a legend. A series unit needs evidence of an actual
continuous series in the image, not just an isolated marker. A raster
containing only ONE series belongs to that series. If one immutable
element combines MULTIPLE series, apply the shared-mark rule below: do
not merge independently identifiable series merely because their
graphical marks share one element. Legend entries alone remain legend
context; never duplicate their labels into a series.
- Treemap/waffle/heatmap: one unit per terminal cell/category
represented by a contiguous labeled region; all fragments making one
named terminal region belong together. Ancestor labels are chart
context. Do not invent optional hierarchy tiers based on color alone.
- Map: one unit per visually identifiable geographic observation. Keep
each country/region callout, country name, value and leader together
even when its map region is embedded in a shared raster. A raster
spanning multiple countries follows the shared-mark rule below; it is
never one artificial all-countries observation. Separately extracted
regions join their own units; they do not change the grouping of other
countries.
- Table: one unit per body row; its cells and row label belong
together. Column headers belong to the chart heading, and shared
rules/grid are chart context.
- Timeline/flow/network: one unit per event/node, with its own label;
connecting edges belong to chart context unless exclusively internal
to an event/node.
- Other/mixed: identify the actual compared observation and declare
its visual evidence. Avoid claiming a supported family when the image
shows another.

Every unit includes a marks list (a subset of its members) and
evidence identifying the complete observation. For stacked/grouped
bars identify the WHOLE category. A separately available mark must
join its own unit, not be left elsewhere.

## Shared or missing atomic marks

A visual primitive may be extracted as one immutable element even
though its pixels contain marks belonging to MULTIPLE
convention-defined observations. For example, one Africa map image
contains multiple country regions, one plot image contains multiple
series, or one ring image contains multiple sectors. Place this shared
primitive ONCE in chart.other, at the chart ancestor of those
observations. Do not assign it to one arbitrarily chosen observation.
Do not merge all observations and their labels into one composite unit
merely because the primitive is indivisible. A missing individual
region/mark annotation must not change the semantic grouping of the
labels compared with a rendering whose marks were extracted
separately.

If independently identifiable observations have separable inline
category names, values, leaders, flags or other associated elements,
construct one unit per such observation from those available elements.
Include any separately extracted marks of that observation in its
```

Prompt B.6. Shared semantic grouping policy (continued)

```
marks list. If its mark exists only inside the shared primitive (or is
visible but not separately localized), marks may be empty: record the
exact shared/missing element limitation in defects and state the
visual observation in that unit's evidence. This is a missing
geometric child, not permission to collapse the remaining semantic
relationships. It is also not permission to invent observations from
arbitrary titles, scale ticks, a detached legend, or uninterpretable
prose. A unit still requires visible evidence of an actual observation
in the image.

When no separable elements identify an observation at all, do not
create an empty unit; retain the shared primitive in chart.other and
record the limit. A primitive spanning multiple MARKS OF THE SAME
observation still belongs to that one unit: e.g. all segments of one
stacked bar or all points of one continuous series. The shared-mark
rule concerns multiple semantic observations, not multiple colors,
polygons, time points or physical fragments within one observation.
Never split, duplicate, invent or edit an immutable element to resolve
this limit.

## Context ownership

Attach category names directly to their units, even when aligned like
x ticks. Numeric scale ticks, axis line and axis title form one group
per actual axis. Gridlines and common plot background go in
chart.other. Do not put values or category names in axis groups. A
local plot title/subtitle/unit is chart.heading; the document
title/subtitle is one document header. In a single-chart image use the
top headline as document header, not a redundant local chart heading.
Column headings and their sort/direction indicators form chart.heading
in ALL families, including bar-table hybrids; this ownership must not
depend on choosing the family name table versus bar. For multi-panel
figures attach a heading to its panel only when it names that specific
measurement. Global heading stays in the document header.

All legend entries are mandatory semantic units, including an entry
already represented by one atomic element. Each entry's key and label
go together. Inline labels printed inside/adjacent to their own area
band, pie sector, line endpoint or mark are DATA LABELS and belong to
that data unit, including a category icon shown beside such a label.
They are not legend entries. A legend entry needs a separate key-label
example or an explicit detached legend region; thematic icons alone do
not turn inline data labels into a legend. Legend headings are
separate legend context. The compiler always creates a legend and
legend_item tier, for one entry as well as multiple entries.

Document groups are flat at ROOT: header, source, brand, license,
illustration, annotation. No footer super-wrapper. Combine adjacent
source/footnote lines belonging to the same credit block. Keep
distinct columns/visibly separate credit blocks separate. A publisher
handle belongs to its brand, not the data source. A textual footnote
such as a forecast/estimate marker or sample-size qualification is a
source note even when placed near a plot; it is not chart.other or a
standalone annotation. Group it with adjacent source notes only when
they form the same text block. Always represent a visible brand as a
brand unit, even one atomic logo. A logo identifying a data category
goes to that category, not a publisher brand. A small title accent bar
belongs to the document header. Generic atomic decorative artwork and
pure styling/background swashes remain at ROOT. A rounded
card/backplate merely surrounding a chart, document paragraph or
```

Prompt B.6. Shared semantic grouping policy (continued)

```
takeaway remains at ROOT, not inside a data/annotation unit; its
geometry does not express ownership. Keep actual plot-area background
and gridlines inside chart.other. An external takeaway/explanation
card is a document annotation block; its title, text and semantic icon
form that block, while its decorative backplate remains at ROOT. Use
an illustration group only for multiple elements forming ONE spatially
connected illustrated object. Never combine disconnected decorative
objects merely because they share a theme, and never assign decoration
to the nearest data unit. Explanatory prose about an entire chart goes
to chart.other; category-specific explanatory prose goes to its unit.
Pure background elements remain at ROOT.

## Integrity and review

Every E identifier occurs exactly once in root, document.members, or a
chart's unit.members/axes/legend_entries/legend_titles/heading/other.
The separate marks list only identifies the mark subset; it does not
give a second parent. Use no old node labels, invented element IDs, or
paired-image assumptions. Groups and descriptions are compiled from
this semantic plan. The independent review must check the complete
observation units, every chart boundary, every element's ownership and
immutable granularity defects. A plausible alternative tree is
insufficient if it violates this convention.
```

Prompt B.7. Semantic grouping construction

```
From your own image and immutable elements, identify complete
observations and return the semantic plan in the exact JSON schema.
Apply the frozen convention. Do not return arbitrary groups or edit
elements. Unit IDs are globally unique within this image. Evidence is
brief. Return all required arrays, including empty ones. Chart
families come from the image, not a prior annotation.

POLICY:
{{GROUPING_POLICY}}
IMMUTABLE ELEMENTS:
{{ELEMENT_CONTEXT}}
SCHEMA:
{{GROUP_PLAN_SCHEMA}}
```

Correction and review prompts. The templates below implement the localization-recovery, visual-correction, and grouping-review stages described in Appendix B.2. They use the same placeholder notation as the annotation prompts. The shared visual instructions in Prompt B.3 precede localization recovery, crop correction, and whole-image review. Independent grouping review instead receives the full grouping policy in Prompt B.6.

Localization recovery. Recovery requests receive a crop of the failed tool region, the region bounds, and successful candidates that must be preserved. Text and non-text recovery use separate instructions and return only localized elements and uncertainties. Their crop-local boxes are mapped back to full-image coordinates before subsequent annotation.

Prompt B.8. Recovery of text localization

```
EXPLICIT LOCALIZATION RECOVERY: the text tool failed in the listed
regions. Return exactly {elements,uncertainties}. Locate ONLY text
labels; each element has bbox,text,reason. Use CROP-LOCAL 0..1000 YXYX
```

Prompt B.8. Recovery of text localization (continued)

```
coordinates. Include a complete visible instance only if its center is
inside both the tile ownership core and a failed region. Existing
successful tool candidates remain preserved; avoid duplicating them.
No groups, semantic roles, palettes or descriptions here: a separate
attribute stage follows localization recovery. For unreadable source
text retain visible spans and [illegible]; record source_illegible.

TILE PIXEL XYXY: {{TILE}}
FULL IMAGE SIZE: {{IMAGE_SIZE}}
FAILED REGIONS (FULL IMAGE PIXEL XYXY): {{FAILED_REGIONS}}
EXISTING TOOL CANDIDATES (FULL NORMALIZED YXYX):
{{EXISTING_CANDIDATES}}
```

Prompt B.9. Recovery of non-text localization

```
EXPLICIT LOCALIZATION RECOVERY: the visual tool failed in the listed
regions. Return exactly {elements,uncertainties}. Locate ONLY non-text
instances; each element has bbox,reason. Use CROP-LOCAL 0..1000 YXYX
coordinates. Include a complete visible instance only if its center is
inside both the tile ownership core and a failed region. Existing
successful tool candidates remain preserved; avoid duplicating them.
No groups, semantic roles, palettes or descriptions here: a separate
attribute stage follows localization recovery. For unreadable source
text retain visible spans and [illegible]; record source_illegible.

TILE PIXEL XYXY: {{TILE}}
FULL IMAGE SIZE: {{IMAGE_SIZE}}
FAILED REGIONS (FULL IMAGE PIXEL XYXY): {{FAILED_REGIONS}}
EXISTING TOOL CANDIDATES (FULL NORMALIZED YXYX):
{{EXISTING_CANDIDATES}}
```

Local and whole-image correction. Prompt B.10 defines the shared patch output contract: additions or updates, removals, unresolved findings, and candidate dispositions. It is inserted into `PATCH_CONTRACT` for crop correction. For whole-image review, it is inserted into `PATCH_CONTRACT_AS_NESTED_OBJECT`, with its opening phrase changed from "Return exactly" to "The nested patch object contains exactly". Crop correction receives the original image followed by its crop, the local graph, candidates, and known issues. Whole-image review receives the original image, complete graph, indexed findings, and tool-coverage gaps.

Prompt B.10. Shared correction-patch contract

```
Return exactly {upsert,remove,uncertainties,candidate_dispositions}.
upsert entries: {node:<complete node matching
schema>,reason:<pixel-backed reason>}. remove entries:
{label,replacement:<existing/new group or element label, or
null>,reason}. Never delete a parent without explicitly reparenting
its children, or providing a replacement. Unchanged nodes are retained
automatically. Never return a whole replacement graph.
candidate_dispositions entries:
{candidate_id,state,node_labels,reason}. state is represented (cite
actual element labels), rejected (false/duplicate tool proposal,
explain visual evidence; empty node_labels), or unresolved (empty
node_labels). Only dispose candidate IDs provided in this request. A
merged OCR proposal may map to several separate text nodes. A detector
container may map to its separate visible child instances. Return
empty arrays when no edits/findings/dispositions are needed. ALL
```

Prompt B.10. Shared correction-patch contract (continued)

```
output boxes refer to FULL-IMAGE normalized coordinates, including
when looking at the second image crop.
```

Prompt B.11. Crop-based visual correction

```
{{PATCH_CONTRACT}}

First image: original full image; second: crop. The supplied graph is
a local view containing all groups and intersecting elements; other
elements remain in the full graph. Inspect EVERY object in this crop,
including undetected icons and text. Fix merged instances, misplaced
parents, missing marks, text errors and duplicate boundary objects.
TILE PIXEL XYXY: {{TILE}}
FULL IMAGE SIZE: {{IMAGE_SIZE}}
TARGET SCHEMA: {{TARGET_SCHEMA}}
CURRENT GRAPH: {{LOCAL_GRAPH}}
UNACCOUNTED TOOL CANDIDATES: {{CANDIDATES}}
KNOWN ISSUES: {{ISSUES}}
```

Prompt B.12. Whole-image visual review

```
Perform a final whole-image visual audit. Correct remaining defects
through a sparse patch. Check compound DATA graphics at the level of
individual encoded observations. A choropleth map is a chart/group
with separate shape elements described as data marks for independently
colored geographic regions, not one image described as an
illustration. Keep whole illustrative maps as images only when their
regions are not independently encoding data. Recount both filled and
outline comparison bars, stacked segments and pictogram units. Do not
accept many-to-one collapse of distinct data marks as deduplication;
merge only genuinely duplicate detections or parts of the same mark.
Return exactly {patch,finding_dispositions,gap_dispositions}. patch
follows the contract below. finding_dispositions contains EVERY
supplied finding index exactly once: {index,state,reason}, state
resolved or unresolved. Resolve only if pixels and the resulting graph
demonstrate repair. Preserve genuine source_illegible limitations.
gap_dispositions contains EVERY tool gap index exactly once:
{index,state,reason}, state resolved or unresolved. Resolved means
this entire region was visually re-observed; it does NOT retroactively
make the tool successful. State remaining coverage problems in
patch.uncertainties.
{{PATCH_CONTRACT_AS_NESTED_OBJECT}}

TARGET SCHEMA: {{TARGET_SCHEMA}}
CURRENT GRAPH: {{GRAPH}}
UNACCOUNTED TOOL CANDIDATES: {{CANDIDATES}}
INDEXED FINDINGS: {{FINDINGS}}
INDEXED TOOL GAPS: {{TOOL_GAPS}}
GEOMETRY DIAGNOSTICS: {{GEOMETRY}}
```

Grouping verification and repair. The independent reviewer receives the image, fixed elements, shared grouping policy, proposed plan, and its fingerprint. Prompt B.13 requires an accept-or-repair decision with explicit chart and observation coverage. When a response fails structural validation, Prompt B.14 is appended to the corresponding request. When grouping review rejects a plan, Prompt B.15 is appended to a new grouping-construction request, after which the plan is reviewed again.

Prompt B.13. Independent grouping review

```
Independently inspect the image and each immutable element against the
proposed semantic plan. Check every complete observation unit and
every chart boundary. Especially reject segment-per-unit stacked bars,
series-per-unit grouped bars, split endpoints of a connected dumbbell,
table/bar-dependent column-header ownership, inconsistent legend
entries, misplaced category labels, and spurious extra chart panels.
Reject treating derived point annotations as independent line/area
series, and reject inline data labels misclassified as legend entries.
Check all ownership, not merely the JSON shape. Return accept only if
the convention is satisfied; otherwise precise repair issues.
Immutable element defects do not authorize element edits: record them
separately. If a composite element makes perfect units impossible, the
least ambiguous faithful single ownership is acceptable WITH an
explicit defect, never duplication.

POLICY:
{{GROUPING_POLICY}}
IMMUTABLE ELEMENTS:
{{ELEMENT_CONTEXT}}
PROPOSED PLAN:
{{PLAN}}
FINGERPRINT: {{PLAN_FINGERPRINT}}
REVIEW WIRE SCHEMA:
{{GROUP_REVIEW_SCHEMA}}

Pay particular attention to shared primitives: a
multi-country/multi-category raster must NOT cause all identifiable
observations and their labels to become one composite unit. Its single
owner is chart.other; separable observation labels/values/leaders
retain their own units even if individual region/mark children are
unavailable. A raster belonging to only ONE semantic observation
remains with that observation. Do not invent units from legends or
scale ticks. Explicitly report ALL checked units using inclusive
numeric U-ID intervals: [[1,4],[8,8]] means U1,U2,U3,U4,U8 only. Do
not omit IDs, claim nonexistent IDs, or check only the first 100. The
decoder rejects incomplete coverage.
```

Prompt B.14. Response-format correction

```
Previous response violated the contract: {{VALIDATION_ERROR}}
Correct the response using the original image and contract.
```

Prompt B.15. Grouping-plan correction

```
PREVIOUS PROPOSAL:
{{PLAN}}
REQUIRED CORRECTIONS:
{{ISSUES}}
Preserve already-correct observations. Shared composite graphics
belong at their chart; identifiable country/category/sector labels
remain in separate observation units. An empty marks list is permitted
only for a visually supported shared/missing mark with a recorded
source limitation.
```

C Dataset Sources, Licenses, and Extended Statistics

C.1 Sources and Licenses

ChartGalaxy++ augments real and synthetic infographic charts from ChartGalaxy (Li et al., 2026b) with scene graph annotations. The real charts originate from publisher websites and sharing platforms. Table 11 summarizes licenses and reuse conditions for selected chart sources documented by ChartGalaxy, with links to the official policies. These include Creative Commons licenses and source-specific terms covering permitted uses, attribution, and modification.

Table 11: Selected chart sources and their licenses, following ChartGalaxy (Li et al., 2026b). Source names and license entries link to official policies.

Name	URL	License
Statista (Daily Data)	statista.com	CC BY-ND 4.0
Visual Capitalist	visualcapitalist.com	Customized
Our World in Data	ourworldindata.org	CC BY
OECD	oecd.org	CC BY 4.0
Office for National Statistics	ons.gov.uk	OGL
Kaiser Family Foundation	kff.org	CC BY-NC-ND 4.0
Information Is Beautiful	informationisbeautiful.net	Customized
Chit Chart	chitchart.com	Customized

For real charts and images, we only provide their source URL, image URL, and scene graph annotations, following the original content's license and attribution requirements.

For template-based synthetic charts, ChartGalaxy documents the licenses of the input tables and visual assets (Li et al., 2026b). Its reported tabular resources include UN Data (public domain), Our World in Data (CC BY), VizNet (CC BY-NC-SA), and Papers with Code (CC BY-SA). Its visual resources include Icon645 (CC BY-NC-SA), Flaticon and Iconshock (royalty-free licenses), Heroicons (MIT), and Google Material Icons (Apache 2.0). These licenses apply to their respective input resources, with the associated attribution and reuse conditions.

C.2 Selection and Evaluation Splits

Subset selection. We target approximately 200,000 training charts with a real-to-synthetic ratio of about $1:3$, a mixture found effective in studies (Rang et al., 2024). The size target limits annotation and fine-tuning costs, while sampling source collections separately prevents the much larger layout-generated collection from determining the training mixture. We retain real charts to cover real-world designs and use synthetic charts to expand structural and stylistic variation. ChartGalaxy also reports gains from combining real and synthetic training data (Li et al., 2026b). This yields 56,244 real and 159,951 synthetic training charts (26.0% and 74.0%), for a total of 216,195. Together with the 1,000-image main test set, these form the 217,195 charts in ChartGalaxy++.

Deduplication and split independence. We inherit image deduplication from ChartGalaxy, whose real-image collection uses perceptual hashing and CLIP similarity (Li et al., 2026b). Subset selection preserves the original image bytes, consolidates exact duplicate image files, and excludes registered evaluation and development samples by sample identifier and image hash. The image-to-scene-graph model is trained on the 216,195 training images without including images from the main test set.

Test sets across applications. The 1,000-image main test set contains 500 real charts and 500 synthetic charts, curated for source coverage and annotation quality. It is used for both image-to-scene-graph prediction and scene graph preservation evaluation. The QA benchmark is constructed separately from the full ChartGalaxy++ collection of 217,195 images and contains 1,266 questions about 1,000 charts. The QA selection is not restricted to the main test set. QA evaluates the benefit of annotated scene graphs during inference without additional model fine-tuning, as detailed in Appendix E.3.

C.3 Extended Annotation Statistics

Counting conventions. All statistics below use the same 217,195 charts as Sec. 4.2. We distinguish the 3,088,632 annotated group nodes from the implicit root of each image. Including these roots yields 3,305,827 groups, matching the main-text average of 15.2 groups per chart. The root is excluded from the seven group-type counts in Table 12. Each stored element or group has one parent, so the number of hierarchical edges is the number of elements plus the number of non-root groups: $15{,}591{,}656 + 3{,}088{,}632 = 18{,}680{,}288$. The depth of a scene graph is the number of nodes on its longest root-to-element path, counting both the root and the terminal element. A graph whose elements are all direct children of the root therefore has depth two.

Table 12: Element and group-type distributions. Percentages use all 15,591,656 elements or all 3,088,632 non-root groups as the denominator, respectively.

Node kind	Type	Count	Percentage
Element	`text`	6,044,031	38.76%
	`image`	1,107,791	7.11%
	`shape`	8,439,834	54.13%
Group	`chart`	221,006	7.16%
	`axis`	72,699	2.35%
	`legend`	88,976	2.88%
	`legend_item`	166,058	5.38%
	`data_item`	2,074,082	67.15%
	`series`	73,181	2.37%
	`panel`	392,630	12.71%

Graph size and hierarchy. Table 13 summarizes the per-chart distributions without truncating large graphs. The median chart contains 54 elements and 10 non-root groups, while the 99th percentiles are 336 and 57, respectively. Nested groups occur in 211,736 charts (97.49%): at least one non-root group directly contains another group. The median hierarchy depth is four, with a maximum of eight.

Table 13: Per-chart distributions over all 217,195 charts. Total nodes include one implicit root per chart; non-root groups exclude it. Quantiles use linear interpolation of the ordered observations.

Quantity	Min	P25	Median	P75	P90	P99	Max	Mean
Elements	1	35	54	85	121	336	3,682	71.79
Non-root groups	0	7	10	18	31	57	234	14.22
Total nodes	2	44	67	108	151	344	3,725	87.01
Hierarchy depth	2	4	4	5	6	7	8	4.54

Fig. 7 compares the annotated structure of real and synthetic charts. Synthetic charts have a higher median element count (58 versus 49). Both subsets have a median of 11 groups when the root is included, but their 90th-percentile group counts differ: 34 for synthetic charts and 19 for real charts. Thus, the same median group count masks a difference in the frequency of charts with many groups.

D Image-to-Scene-Graph Prediction

D.1 Training Configurations

We build our image-to-scene-graph model through supervised fine-tuning of Qwen3.5-4B (Qwen Team, 2026a) on the ChartGalaxy++ training set. We perform full-parameter fine-tuning for $100{,}000$ steps. Training samples are packed into sequences of up to $14{,}336$ tokens, with a global batch size of 32 packed sequences. We use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.1, and gradient-norm clipping at 1.0. The learning rate warms up from zero to 5×10^{-5} over the first 500 steps, followed by cosine decay to zero over the remainder of training. The target serializes node attributes and parent assignments, excluding spatial relationships that are derived from bounding boxes. The reported model uses the checkpoint at step $100{,}000$. At inference, it uses

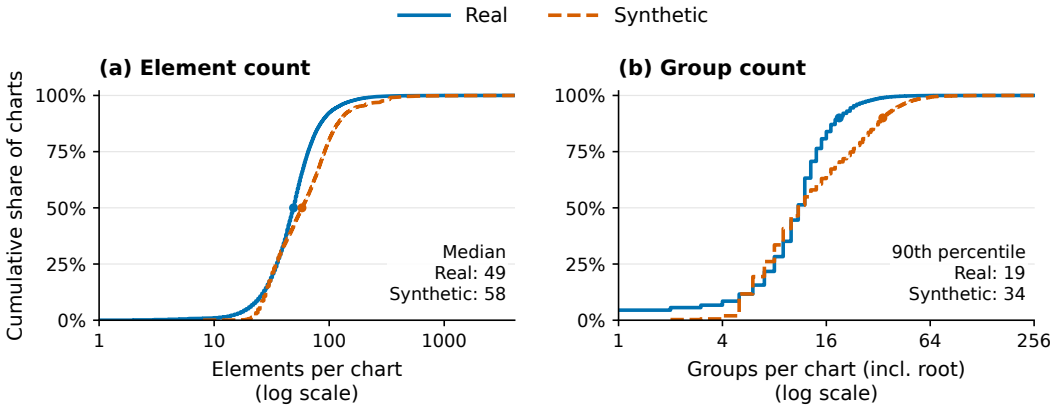


Figure 7: Empirical cumulative distributions of (a) element counts and (b) group counts for real and synthetic infographic charts. Both count axes are logarithmic and include all charts. Markers identify the median element count and the 90th-percentile group count. Each distribution is normalized within its real or synthetic subset. Group counts include the implicit root, matching the main-text convention.

vLLM 0.20.2, a context limit of 32,768 tokens, a fixed output limit of 20,000 tokens, temperature 0, and top-p of 1.

D.2 Baseline Settings

All models are evaluated on the same 1,000 test images and reference graphs. Inference receives the original image and task specification, without reference annotations or source data tables. The common task predicts node attributes and hierarchical relationships; spatial relationships are computed afterward. The recorded runs use model-specific serialization and decoding settings, summarized in Table 14. Thus, the comparison evaluates the specified inference pipelines, whose call counts and token budgets differ.

Table 14: Inference settings for image-to-scene-graph prediction. Context and output limits are in tokens; an API dash denotes a context limit not explicitly set by the client. Open-model revision prefixes identify the archived weight revisions; full identifiers are retained in the accompanying inference configuration files.

Model (revision prefix)	Context	Output	Decoding
Molmo2-8B (`e28fa28`)	36,864	24,576	Sampling; inventory first
InternVL3.5-8B (`9bb6a56`)	32,768	24,576	Temperature 0
Gemma 4-12B-IT (`707f0a3`)	65,536	24,576	Temperature 0
Qwen3.5-4B (`851bf6e`)	32,768	24,576	Sampling; thinking off
Qwen3.8-27B (`1d4bf0f`)	65,536	24,576	Temperature 0; thinking off
Qwen3.8-Flash-Next-FP8 (`236dfdf`)	65,536	24,576	Temperature 0; thinking off
GLM-5.3-Flash (`eb9eb20`)	65,536	24,576	Temperature 0; high reasoning
Gemini 3.8 Flash	–	24,576	LOW thinking
GPT-5.6 Sol	–	24,576	Medium reasoning
GPT-6 Astra	–	24,576	Medium reasoning

Decoding and format adaptation. The two GPT models use structured JSON output and one prediction request per image, without format-feedback generation. Their API identifiers are `gpt-5.6-sol` and `gpt-6-astra`. Gemini uses `gemini-3.8-flash`, a JSON response schema, and up to three generation attempts with validation feedback, stopping at the first valid graph. Open models use a compact JSON grammar and up to three generation attempts under the same stopping rule. Molmo2 instead uses one unconstrained visual-inventory request, capped at 8,192 tokens, followed by at most two graph requests; up to 10,000 characters of its inventory are included in the graph prompt. For Qwen3.5-4B and Molmo2, sampling uses temperature 0.7, top-p

0.8, top-k 20, minimum-p 0, presence penalty 1.5, repetition penalty 1, and seed 20260916. Recorded operational retries address request failures and do not select predictions by evaluation score.

GPT predicts native $[y_0, x_0, y_1, x_1]$ boxes and hexadecimal colors. The other baseline interfaces use $[x_0, y_0, x_1, y_1]$ boxes and integer RGBA tuples, except that Molmo2 names the four coordinate fields explicitly. These forms are converted deterministically to the common coordinate and color representation before scoring. Reversed endpoint pairs are sorted within their own axes; unambiguous label renaming updates all associated references. These operations do not use reference graphs or estimate missing coordinates.

Invalid and partial outputs. Every test image remains in the evaluation. An unrecovered failure contributes zero node and spatial F1, and its reference parent edges remain in the hierarchical denominator. The recorded InternVL3.5 and Molmo2 recovery retains individually evaluable nodes from partial outputs while preserving rejected records in the node and parent denominators. Nodes with usable boxes can contribute predicted spatial relations even when their other fields prevent matching; records without usable boxes create no predicted near edges, while missed reference near edges remain false negatives and node and parent penalties are retained. For our model, the fixed postprocessing also retains parseable prefixes of truncated outputs, removes identical duplicate keys, and resolves unambiguous serialization inconsistencies. An incomplete terminal node may be quarantined; missing visual objects are not generated during this recovery. These are scores of the recorded prediction and postprocessing pipelines, rather than raw JSON-validity rates.

D.3 Evaluation Metrics

D.3.1 Node Matching

The node-level columns in Table 1 evaluate both element and group nodes. Let V_i and $\widehat{V}_i$ be the reference and predicted node sets for image i, excluding the virtual `ROOT`. Global style and composition descriptions are not scored as nodes. Only nodes of the same kind, element or group, are eligible to match. Their concrete types need not agree: type agreement is a scored attribute rather than a hard constraint. Node labels and parent assignments do not enter the matching objective, and there is no IoU threshold.

For an eligible reference–prediction pair $(v, \hat{v})$, the applicable attribute set $\mathcal{A}(v)$ always contains type, box, and description. It additionally contains text when the reference has nonempty normalized text, and color when its palette is nonempty. Applicability is determined by the reference; omitting a required predicted attribute does not remove it from the score. All five attributes have nominal weight 0.2, renormalized over $\mathcal{A}(v)$. Thus, the pair score is their arithmetic mean:

$$\widetilde{s}(v, \hat{v}) = \frac{1}{|\mathcal{A}(v)|} \sum_{a \in \mathcal{A}(v)} \widetilde{s}_a(v, \hat{v}). \tag{5}$$

We apply maximum-weight Hungarian assignment using $\widetilde{s}$, assigning zero weight to ineligible pairs and discarding zero-weight assignments. Denote the retained one-to-one matches by M_i and the induced prediction-to-reference map by π_i. We additionally set $\pi_i(\texttt{ROOT}) = \texttt{ROOT}$ for hierarchical evaluation.

To control the cost of description judging, we use a lightweight lexical proxy during assignment and apply the description judge only to the matched node pairs. After case folding and whitespace collapse, the proxy is the maximum of RapidFuzz string-ratio and token-set-ratio similarities, divided by 100. After fixing M_i, we replace this proxy with the description-judge score in Appendix D.3.2 and recompute the pair mean $s(v, \hat{v})$. We do not rerun assignment after judging. All reported attributes and relationship scores use this same fixed matching.

D.3.2 Attribute Similarities

Type. Type similarity is 1 for equal concrete types and 0 otherwise. For elements, the types are text, image, and shape; for groups, they are the semantic group types.

Bounding box. For boxes B and $\widehat{B}$, let I, U, and C be the areas of their intersection, union, and smallest enclosing axis-aligned rectangle. We normalize GIoU (Rezatofighi et al., 2019) to $[0, 1]$:

$$s_{\text{box}} = \frac{1}{2}\left(1 + \frac{I}{U} - \frac{C - U}{C}\right). \tag{6}$$

The normalized score is used both during assignment and in the reported Box column. Scored boxes have positive area and coordinates within the normalized image canvas.

Text. We apply Unicode NFKC normalization, replace the Unicode minus sign with an ASCII hyphen, and collapse whitespace without changing letter case. For resulting strings t and $\hat{t}$, the base similarity is

$$s_{\text{text}}^{\text{base}} = 1 - \frac{\text{Lev}(t, \hat{t})}{\max(|t|, |\hat{t}|)}. \tag{7}$$

A missing or empty prediction receives zero when reference text is nonempty. To prevent long strings from masking a wrong value, we cap the score at 0.5 when the ordered lists of numeric values, adjacent units, or currency signs differ. Numeric comparison normalizes equivalent decimal/scientific spellings and unambiguous three-digit comma grouping; unit spelling and case are retained. The rule does not infer locale or convert between units.

Color palette. We canonicalize hexadecimal RGB/RGBA colors, expand three-digit RGB, set omitted alpha to 1, remove duplicates, and identify all fully transparent colors with one transparent entry. For colors c and $\hat{c}$ with alpha values α and $\hat{\alpha}$, define

$$k(c, \hat{c}) = \max\left(0, 1 - \frac{\Delta E_{00}(c, \hat{c})}{100}\right)(1 - |\alpha - \hat{\alpha}|), \tag{8}$$

where ΔE_{00} is CIEDE2000 (Luo et al., 2001) on the RGB colors converted to CIELAB. When both colors are fully transparent, $k = 1$ regardless of RGB. For reference palette $\mathcal{C}$ and predicted palette $\widehat{\mathcal{C}}$, we measure bidirectional coverage:

$$P_c = \frac{\sum_{\hat{c}\in\widehat{\mathcal{C}}} \max_{c\in\mathcal{C}} k(c, \hat{c})}{|\widehat{\mathcal{C}}|}, \qquad R_c = \frac{\sum_{c\in\mathcal{C}} \max_{\hat{c}\in\widehat{\mathcal{C}}} k(c, \hat{c})}{|\mathcal{C}|}, \qquad s_{\text{color}} = \frac{2P_cR_c}{P_c + R_c}. \tag{9}$$

Colors may provide coverage to more than one counterpart; this is not one-to-one palette assignment. An empty predicted palette or zero combined coverage receives zero. An empty reference palette makes this attribute inapplicable.

Description. For each matched pair, GPT-5.6 Luna compares the predicted and reference description strings. It receives no image and measures preservation of reference meaning, rather than independently verifying visual truth. Each request contains one comparison, uses low reasoning effort and a 4,096-token output cap, and requires a strict JSON result with an integer score and rationale. The returned score $q \in \{0, 1, 2, 3, 4, 5\}$ gives $s_{\text{description}} = q/5$. Harmless paraphrases retain credit; omissions, unsupported additions, and contradictions reduce the score.

D.3.3 Node and Edge Scores

Overall node score and attribute columns. Let $S_i = \sum_{(v,\hat{v})\in M_i} s(v, \hat{v})$. The similarity-weighted node F1 for image i is

$$P_i^{\text{node}} = \frac{S_i}{|\widehat{V}_i|}, \qquad R_i^{\text{node}} = \frac{S_i}{|V_i|}, \qquad F_i^{\text{node}} = \frac{2S_i}{|V_i| + |\widehat{V}_i|}. \tag{10}$$

Unmatched nodes receive no credit but remain in the corresponding denominators. The direct F1 expression also gives zero for an empty prediction when the reference is nonempty. The Overall column averages F_i^{node} over all $N = 1{,}000$ images. Each separate attribute column instead pools the sum and count of its applicable matched-node scores across images. These conditional attribute means describe matched-node quality; the Overall column additionally penalizes omissions and extra nodes.

Hierarchical relationships. Each non-root node contributes one direct parent edge, including an edge to `ROOT` when appropriate. A predicted edge is correct when its child is matched and its parent maps to the reference child's parent:

$$C_i^h = \sum_{(v,\hat{v})\in M_i} \mathbf{1}[\pi_i(\text{parent}(\hat{v})) = \text{parent}(v)]\,. \tag{11}$$

Unknown, unmatched, or invalid parents receive no credit. We score direct parents, without substituting ancestor agreement.

$$F^h = \frac{2\sum_i C_i^h}{\sum_i(|V_i| + |\widehat{V}_i|)}. \tag{12}$$

Spatial relationships. We derive geometry for all selected pairs of non-root nodes on each side, independently of the sparse serialized relationship selection in Appendix A.3. One node pair and one relationship type form one scoring unit. Direction, alignment, and overlap use the bounding-box predicates in Appendix A.3, with alignment tolerance 5. A diagonal direction such as `top-left` is one directional relationship, and all satisfied alignment bases form one alignment relationship. Only nonempty direction/alignment and positive-area overlap are counted; agreement on absence is not credited. A categorical relation receives one unit of credit only if both endpoints match and the direction or complete alignment set, or the overlap predicate, agrees under the mapped endpoint order.

The proximity component scores positive Boolean near relationships using the predicate in Appendix A.3, with the actual image width and height on each side. Let E_i^{near} and $\widehat{E}_i^{\text{near}}$ be the reference and predicted near-pair sets over all unordered non-root node pairs. A predicted near pair receives one unit of credit only when both endpoints are matched and their mapped reference pair is also near:

$$C_{i,\text{near}}^s = \sum_{\{\hat{u},\hat{v}\}\in\widehat{E}_i^{\text{near}}} \mathbf{1}[\{\pi_i(\hat{u}), \pi_i(\hat{v})\} \in E_i^{\text{near}}]\,, \tag{13}$$

where an unmatched endpoint makes the indicator zero. The proximity reference and prediction counts are $|E_i^{\text{near}}|$ and $|\widehat{E}_i^{\text{near}}|$; their F1 is $2C_{i,\text{near}}^s/(|E_i^{\text{near}}| + |\widehat{E}_i^{\text{near}}|)$ when the denominator is positive. If both sets are empty, the proximity component is undefined and omitted from its component mean; its zero counts contribute nothing to the pooled spatial score below. True negatives receive no credit, and continuous distance similarity is not scored. We recompute the predicate from boxes, so changing which relations are stored does not change the reported spatial score. All reported Spatial scores for image-to-scene-graph prediction and scene graph preservation use this Boolean rule, while keeping the recorded model outputs, node matches, and other spatial components fixed.

Let $C_{i,t}^s$, $R_{i,t}^s$, and $P_{i,t}^s$ be the credit, reference count, and prediction count for spatial type t. We pool the four types within each image, then average images equally:

$$F_i^s = \frac{2\sum_t C_{i,t}^s}{\sum_t(R_{i,t}^s + P_{i,t}^s)}, \qquad F^s = \frac{1}{N}\sum_i F_i^s. \tag{14}$$

For partial outputs, missing or invalid boxes create no predicted near edges; reference near edges without a corresponding correct prediction remain false negatives. Spatial counts use available geometry, while the node and parent penalties above remain unchanged. Every reference image in this cohort has spatial relationships, so the per-image denominator is nonzero.

D.4 Further Analysis

We analyze how extraction performance varies with the number of reference nodes, counting both elements and groups and excluding `ROOT`. The 1,000 test images are divided into five bins: $n < 50$, $50 \le n < 75$, $75 \le n < 100$, $100 \le n \le 125$, and $n > 125$, containing 191, 218, 193, 178, and 220 images, respectively. All three metrics are averaged equally over images within each bin. Predictions, node assignments, and per-image scores are unchanged; all models use the same images, including failed outputs.

Figure 8 reports all eleven models. Our model has the highest Overall, Hierarchical, and Spatial scores in each of the five bins. The leading baselines generally obtain lower scores on images with

more nodes, while our model has smaller differences between the first and last bins. For Hierarchical, GPT-6 Astra decreases from 74.8 in the $n < 50$ bin to 35.0 in the $n > 125$ bin, whereas ours changes from 90.4 to 84.7. The corresponding gap increases from 15.6 to 49.7 points. Overall and Spatial show a similar increase in the first-to-last-bin gap. Our scores do not decrease monotonically: they rise in the 100–125-node bin. Because the bins contain different mixtures of image sources, these curves characterize observed performance across graph sizes rather than isolate a causal effect of node count.

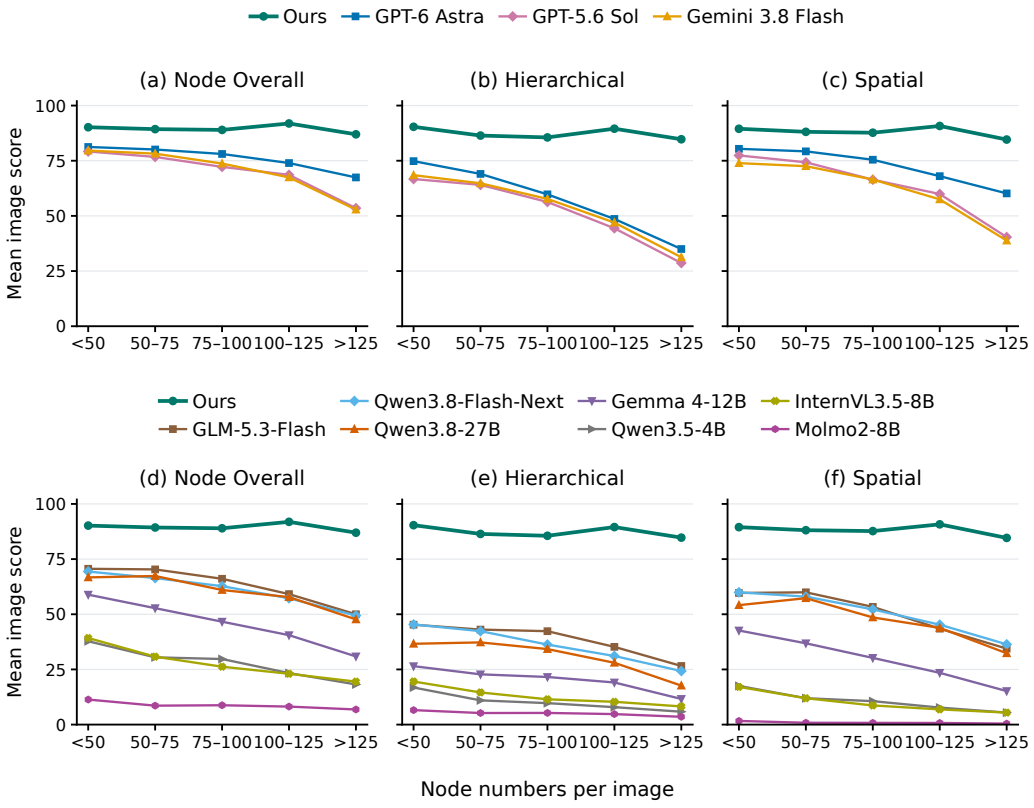


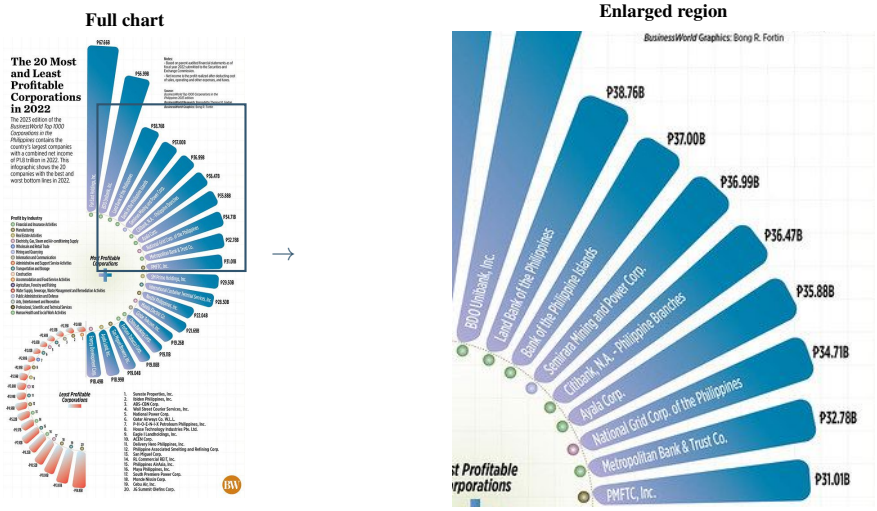
Figure 8: Performance by reference node count. The top row compares our model with the three proprietary baselines; the bottom row includes all seven open-source baselines. All panels use equal image weights within each bin.

E Scene-Graph-Augmented Question Answering

E.1 Example Question-Answer Pairs

Figures 9 and 10 present examples from real infographic charts. Each row pairs the full chart and an enlarged region with its question and reference answer.

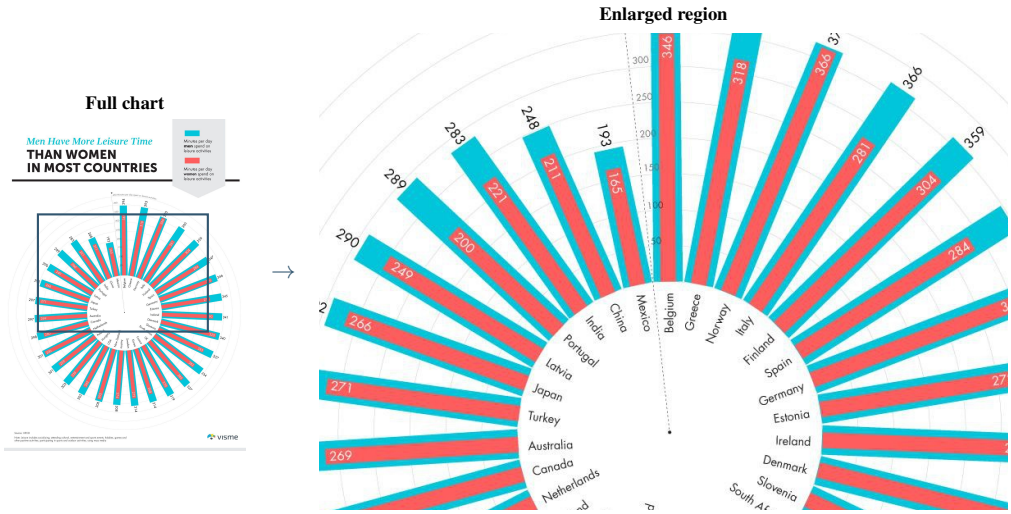
(a) Within-group correspondences



Question. Which corporation has a profit of PHP 38.76B?

Answer. Land Bank of the Philippines

(b) Across-group reasoning



Blue: men; red: women. Values are minutes per day.

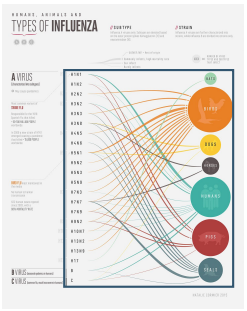
Question. Which country has the larger gap between men's and women's daily leisure time, Portugal or Italy?

Answer. Portugal

Figure 9: QA examples of within-group correspondences and across-group reasoning. Boxes on the full charts mark the regions enlarged on the right.

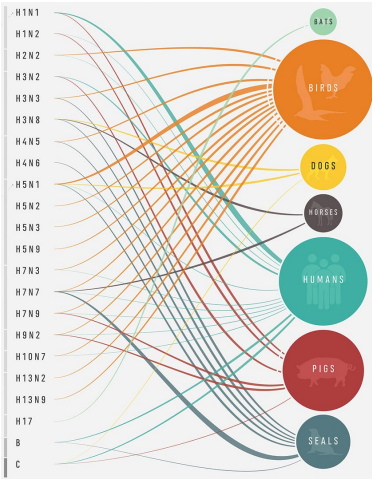
(c) Connected elements

Full chart



→

Enlarged region



Question. Which hosts are connected to H4N5 or H4N6?

Answer. Birds, Seals

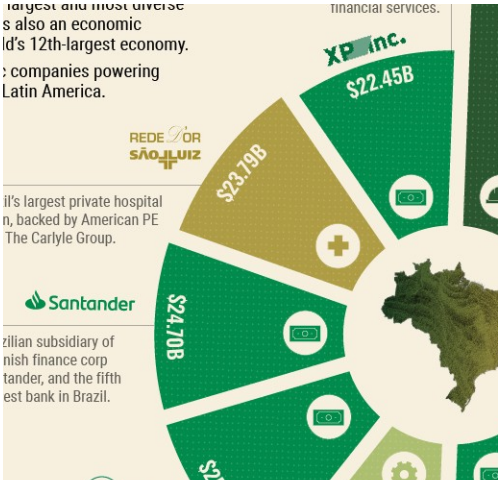
(d) Adjacent elements

Full chart



→

Enlarged region



Question. Which two companies immediately border the wedge labeled $23.79B?

Answer. Santander, XP Inc.

Figure 10: QA examples of connections and spatial adjacency. Boxes on the full charts mark the regions enlarged on the right.

E.2 Details of the Scoring Rules

Answer extraction. Direct responses are read as the answer, while guided responses are scored using only `final_answer`. The parser also accepts an unambiguous final-answer field or section surrounded by imperfect formatting, without consulting the reference answer. Empty or unrecoverable answers count as incorrect, including outputs truncated before the final answer. Accuracy gives equal weight to every question.

Answer-type-specific matching. We use the InfoChartQA evaluator (Xie et al., 2025) with the stored answer type. Text matching is case-insensitive: the released checker accepts exact or contained reference strings, otherwise requiring a `SequenceMatcher` ratio of at least 0.8. Numerical answers use the evaluator's parsing and supported unit, percentage, and magnitude conversions. A converted reference–prediction pair $(v, \hat{v})$ is accepted when

$$\frac{|v - \hat{v}|}{0.01 + |v|} < \epsilon,$$

with $\epsilon = 0.05$ for relaxed numerical questions and $\epsilon = 0.001$ for the stricter branch. The mixed value/text type selects a branch from the reference's format. Lists use ordered or unordered matching; ordered answers must preserve both order and length. Fixed-form answers, such as yes/no responses, use the corresponding evaluator branch.

Equivalent formats. Following InfoChartQA (Xie et al., 2025), equivalent answer representations are normalized before comparison, covering numerical expressions (e.g., "1K" and "1,000"), date formats (e.g., "January 1, 2024" and "2024-01-01"), list representations (e.g., "A, B" and "A and B"), and predefined name aliases (e.g., "USA" and "United States").

E.3 Prompting Settings

Evaluation protocol. All QA models are evaluated without additional fine-tuning for this study. All prompting conditions use the same QA charts and questions. We use low reasoning effort for each model and across the three prompting conditions. The comparison measures the benefit of supplying annotated scene graphs at inference time.

Scene graph input. The scene graphs provided in the QA experiments are produced by our dataset annotation pipeline rather than predicted by the fine-tuned image-to-scene-graph model. Before constructing the QA input, we convert the normalized $[y_0, x_0, y_1, x_1]$ boxes in Appendix A.1 to $[x_{\min}, y_{\min}, x_{\max}, y_{\max}]$ in original-image pixels. For an image of width W and height H, this conversion reorders the coordinates and scales the horizontal and vertical components by $W/1000$ and $H/1000$, respectively. The input is serialized as JSON containing nodes and spatial relations. Each node stores a concise label, type, bounding box, parent reference, and available text; spatial relations refer to the same labels. A null parent denotes a root. Labels describe element roles and associated entities using the descriptions and grouping context.

Prompt templates. Boxes E.1 and E.3 reproduce the evaluated system instructions. Box E.2 shows the user-input template; blue placeholders are replaced for each question. The reference answer is withheld.

Prompt E.1. System instruction: Direct prompting

Answer the chart question from the image and follow the requested answer format exactly.

Prompt E.2. User input: Shared template

Text

```
Question: {question}
Answer format: {answer_format}
```

Image

```
{chart_image}
```

Additional input for scene-graph-augmented prompting

```
{scene_graph_json}
```

The answer-format instruction specifies the required answer form, units, precision, or list order and is shared across prompting methods. For guided prompting, it applies to `final_answer`. Box titles and placeholders are presentation aids, not additional model instructions.

Prompt E.3. Shared system instruction: Guided methods

Answer the chart question using the image and any supplied scene graph. Follow the requested answer format exactly and preserve necessary units. Use boxes, group membership, and spatial relations to associate the relevant entities, marks, and values; use node text when available and otherwise read it from the image. Node identifiers, coordinates, and distances are not measured chart values. Bounding boxes use integer [x_min,y_min,x_max,y_max] in original-image pixels, with the origin at the top left. Directions describe the source box relative to the target box; alignment and overlap describe geometry, not semantic identity. Use only the relevant evidence and trust the image if annotations conflict with it.

Locate the evidence relevant to the question in the image first. Then use the supplied annotations, when available, to associate labels, data marks and values within the appropriate groups. Record those key elements and groups before giving the result of the requested lookup, comparison or calculation. When annotations contain no text, read the text and values from the image.

For this condition, use the JSON output format below instead of a final-answer-only response. Apply the question's answer-format instruction, including units, precision and list order, only inside the final_answer string. Return exactly one JSON object, with no Markdown or surrounding prose. Do not include extended reasoning. First output a compact evidence record, then the final_answer field. Use this form:

```
{
  "evidence": [{
    "entity": "relevant chart entity or mark",
    "group": "its relevant visual group or series",
    "node_refs": ["exact supplied node labels, when available"],
    "observed_value": "visible evidence, with units when relevant"
  }],
  "final_answer": "the requested answer"
}
```

Keep the evidence list focused on the key elements needed for this question. It is an evidence record, not a detailed reasoning transcript. Use only node references actually present in the supplied annotations; if no annotations are supplied, use an empty node_refs list and identify the evidence and groups from the image. Do not copy coordinates or spatial distances as measured chart values.

F Scene-Graph-Conditioned Image Generation

F.1 Model-Specific Input Construction

All models generate from textual scene graph specifications. Reference-image pixels are not supplied as conditioning. The scoring-stage hierarchy reconstruction described below can change group counts. The evaluation always compares the generated result with the reference graph.

Native JSON inputs: Ideogram 4 and Qwen-Image-SP. We convert the scene graph to each model's native JSON format, without an additional natural-language instruction prefix. Ideogram 4 receives all source nodes as caption entries, with `type=text` for text and `type=obj` for other nodes. It preserves literal text and normalized $[y_0, x_0, y_1, x_1]$ boxes. Colors beyond the native palette capacity and RGBA opacity are retained in descriptive text. For Qwen-Image-SP, element boxes become `position` strings in `<bbox>left top right bottom</bbox>` format; literal text and visual descriptions populate the `caption` and `appearance` fields. Identifiers in descriptive prose are adapted to the retained records without changing visible text.

Free-form inputs: all other models. The remaining models receive rendering instructions followed by a serialized scene graph, preserving literal text, normalized bounding boxes, and the distinction between layout groups and visible elements. For each model, the input token budget follows either the hard limit of its inference interface or the largest budget found to support stable image generation in preliminary experiments. These limits characterize the model's practical capacity to process structured specifications under the evaluated configuration. When the input exceeds this budget, we truncate the serialized scene graph to a breadth-first prefix of complete nodes that fits within the limit. Evaluation remains against the full reference scene graph, so the reported scores reflect end-to-end performance, including both input-capacity limitations and the preservation of supplied content.

The recorded free-form inputs use $[x_0, y_0, x_1, y_1]$ boxes, converted from the paper's $[y_0, x_0, y_1, x_1]$ convention.

F.2 Generation Configurations

Open-source generation settings. Table 15 lists the checkpoints and sampling settings. The seed is $42 + i$, where i is the sample's zero-based index in the fixed 1,000-image manifest. Generation uses one output per model and sample, with no selection among multiple candidate images. The six non-Ideogram local models use bfloat16.

Table 15: Recorded checkpoints and generation settings. CFG denotes classifier-free guidance.

Model	Checkpoint	Sampling
HiDream-O1-Image	`HiDream-ai/HiDream-O1-Image`	50 steps; CFG 5; shift 3; UniPC
GLM-Image	`zai-org/GLM-Image`	50 steps; CFG 1.5
Qwen-Image-2512	`Qwen/Qwen-Image-2512`	50 steps; CFG 4
Qwen-Image-SP	`Qwen/Qwen-Image` with `heheyas/Qwen-Image-SP` transformer and text-encoder overlays	25 steps; CFG 4
ERNIE-Image-Turbo	`baidu/ERNIE-Image-Turbo`	8 steps; CFG 1; prompt enhancement disabled
SenseNova-U1 Infographic V3	`sensenova/SenseNova-U1-8B-MoT-Infographic-V3`	50 steps; CFG 4; shift 3; thinking disabled
Ideogram 4	Ideogram 4, recorded ComfyUI Quality48 workflow	48 Euler steps; guidance 7 for 45 steps, then 3 for 3 steps; $\mu = 0$, $\sigma = 1.5$

Qwen-Image-2512 uses a fixed negative prompt discouraging poor rendering quality, malformed text, and distorted chart structure. Qwen-Image-SP uses an empty negative prompt. SenseNova uses CFG over the full sampling interval with no CFG normalization. GLM's glyph input contains the literal text of retained nodes; its semantic input contains the compact JSON. The recorded GLM implementation samples the required image-code grids before emitting the terminal end-of-sequence token, and treats glyph-capacity overflow as a failure instead of truncating text.

Proprietary models. The GPT-Image API identifiers are `gpt-image-2` and `gpt-image-2.5-sunburst`. Requests use `size=auto`, `quality=high`, and PNG output. The Nano Banana identifiers are `gemini-3.1-flash-image` and `gemini-3-pro-image` for Nano Banana 2 and Pro, respectively. Both use Vertex `generate_content` with image-only response modality. Image size, maximum output tokens, temperature, and candidate count are left at provider defaults in these requests. The proprietary requests do not set a random seed.

Graph extraction and scoring. Generated images are annotated using the shared procedure in Appendix B; the reference graph is not supplied to this extraction stage. Element text, boxes, types, and colors remain fixed during this grouping step. Matching and attribute similarities follow Appendix D. Overall Node, Hierarchical, and Spatial scores are averaged across the 1,000 images; individual attribute columns pool applicable matched-node scores.

F.3 Further Analysis

Generation performance by input graph size. We analyze how generation performance varies with the number of nodes in the input scene graph, counting elements and groups and excluding `ROOT`. Node counts are measured before model-specific input construction. Using the same bin boundaries as the image-to-scene-graph analysis, we divide the 1,000 inputs into $n < 50$, $50 \le n < 75$, $75 \le n < 100$, $100 \le n \le 125$, and $n > 125$, containing 191, 218, 193, 178, and 220 images, respectively. All models use the same images within each bin, including failed outputs, and all three scores are averaged equally over images. Spatial uses the same Boolean proximity definition as the main results.

Figure 11 reports all eleven models. Every model has lower Overall, Hierarchical, and Spatial scores in the largest-node bin than in the smallest-node bin, although the intermediate curves are not always monotonic. GPT-Image-2.5 Sunburst and Ideogram 4 have similar Overall scores for $n < 50$ (75.3 versus 75.0), while their gap increases to 10.7 points for $n > 125$ (58.7 versus 48.0). Their Hierarchical gap increases from 5.0 to 29.5 points between these two bins, and their Spatial gap increases from 5.9 to 20.4 points. Both models lose accuracy on larger graphs, but Sunburst retains a greater advantage over Ideogram 4 in preserving elements and relationships. Because image-source composition also varies across bins, these curves characterize performance across graph sizes rather than isolate a causal effect of node count.

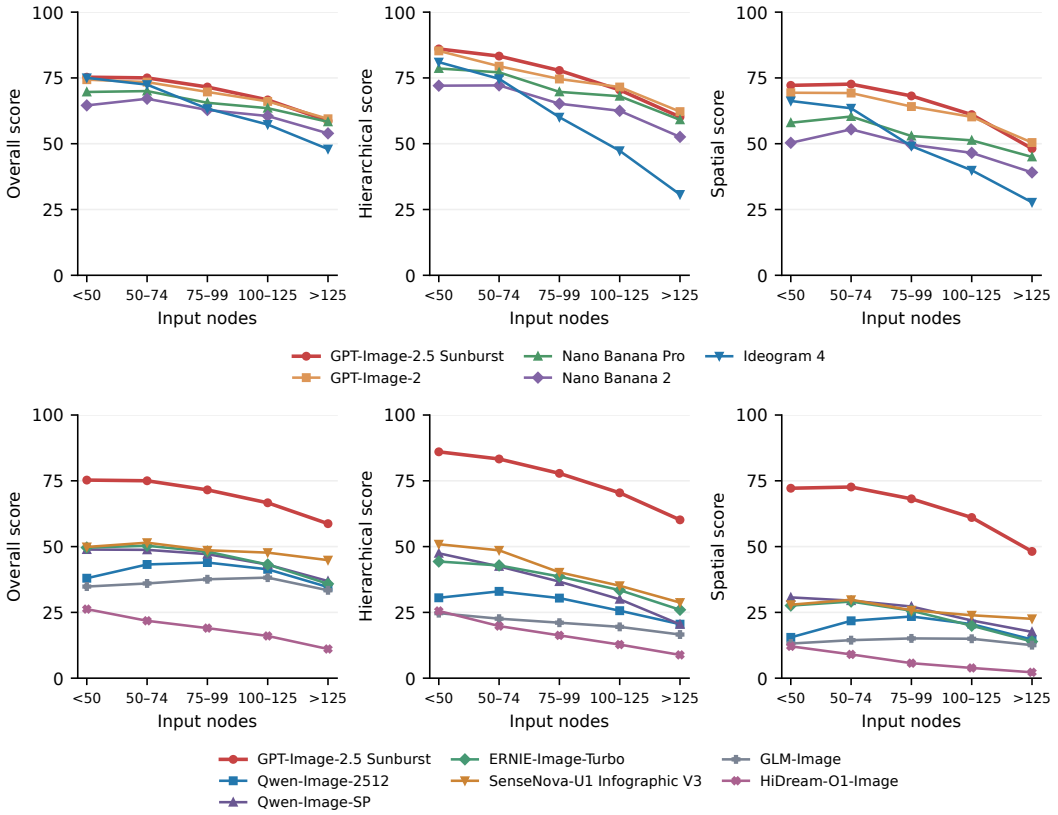


Figure 11: Generation performance by input node count. The top row compares the GPT-Image and Nano Banana variants with Ideogram 4; the bottom row includes the remaining six models, with GPT-Image-2.5 Sunburst repeated for reference. Each point is the mean score over the images in its bin.

Small boxes and positional sensitivity. In the final evaluation references, 71,721 of 99,010 non-root nodes (72.4%) have boxes occupying at most 0.5% of the image area. Across the eleven reported models, these nodes account for 501,672 of 786,305 matched pairs (63.8%). Their mean normalized GIoU similarity is 24.9, compared with 54.8 for larger boxes, on the 0–100 reporting scale. These are conditional means over matched pairs, rather than detection recalls. Small regions have lower Box scores and are sensitive to displacements that remove much of their overlap. Box should therefore be considered alongside text and relationship preservation when assessing a generated chart.

Spatial components. Table 16 separates the four spatial components using the final matching and image-level aggregation. Alignment is the lowest-scoring component for every model. For example, GPT-Image-2.5 Sunburst scores 68.5 for direction but 43.0 for alignment; Ideogram 4 scores 53.0 and 30.2, respectively. Preserving broad directional organization therefore does not imply preserving exact shared edges or centers. Proximity measures positive Boolean near-pair F1, with no credit for pairs that are non-near on both sides, using the geometric rule in Appendix A.3.

Table 16: Spatial relationship scores by component, averaged over all 1,000 images per model. Each column is multiplied by 100. The overall Spatial score in Table 4 pools relationship counts within each image; it is not the arithmetic mean of these four columns.

Model	Direction	Alignment	Overlap	Proximity
HiDream-O1-Image	6.7	3.5	13.5	5.0
GLM-Image	16.0	3.6	19.7	7.5
Qwen-Image-2512	21.9	5.2	25.8	11.1
Qwen-Image-SP	28.2	9.1	34.1	15.2
ERNIE-Image-Turbo	25.5	9.5	34.0	15.2
SenseNova-U1 Infographic V3	28.2	11.1	36.3	16.8
Ideogram 4	53.0	30.2	55.1	36.8
Nano Banana 2	51.5	28.8	57.9	35.3
Nano Banana Pro	57.3	32.6	62.7	39.0
GPT-Image-2	67.2	39.3	69.0	45.4
GPT-Image-2.5 Sunburst	68.5	43.0	69.7	49.7