Beyond Pixels: Introspective and Interactive Grounding for Visualization Agents*

Anonymous ACL submission

Abstract

Vision-Language Models (VLMs) frequently misread values, hallucinate details, and confuse overlapping elements in charts. Current approaches rely solely on pixel interpretation, creating a Pixel-Only Bottleneck: agents treat interactive charts as static images, losing access to the structured specification that encodes exact values. We introduce Introspective and Interactive Visual Grounding (IVG), a framework that combines (1) spec-grounded introspection, which queries the underlying specification for deterministic evidence, with (2) view-grounded interaction, which manipulates the view to resolve visual ambiguity. To enable evaluation without VLM bias, we present iPlotBench, a benchmark of 500 interactive Plotly figures with 6,706 binary questions and ground-truth specifications. Experiments show that introspection improves data reconstruction fidelity, while the combination with interaction achieves the highest QA accuracy (0.81), with +6.7% gains on overlapping geometries. We further demonstrate IVG in deployed agents that explore data autonomously and collaborate with human users in real time.

1 Introduction

Large language models are increasingly trusted to answer complex queries, analyze technical documents, and assist in high-stakes decision-making. Yet this trust is fragile. A persistent limitation is hallucination: generating fluent, confident text that fabricates facts, misremembers details, or contradicts sources. As we move toward multimodal systems, Vision-Language Models inherit and compound this problem. While VLMs extend the capabilities of LLMs to the visual domain, they introduce new forms of error. They exhibit perceptual hallucinations: misreading numbers, confusing visually similar elements, or reporting details not present in the image (Kaul et al., 2024; Jiang et al., 2024; Rohrbach et al., 2018; Li et al., 2023; Leng et al., 2024). In domains like data analysis and chart interpretation, these errors are not cosmetic—they can lead to fundamentally flawed conclusions. Unlike text, where a hallucination might be a subtle twist of phrase, a VLM misreading a trend line or data point breaks the chain of reasoning entirely.

For photographs, resolving such errors is difficult—there is no structured ground truth to verify against. But charts are different. A chart is rendered from a specification that encodes exact values, bindings, and relationships. Whether agents generate charts, answer questions about them, or refine visualizations based on feedback, they could consult this specification directly. Yet today's agents never do. They interpret charts through pixels alone—a static, lossy representation where values become approximate and overlapping elements become ambiguous (Yang et al., 2025; Zhao et al., 2025; Seo et al., 2025; Hong et al., 2025; Li et al., 2025). The specification contains exact data, axis ranges, and trace definitions, yet it goes unused. The view is fixed—agents cannot zoom, toggle, or select to resolve ambiguity. We call this the **Pixel-Only Bottleneck**: reasoning is limited to static pixels, with no access to the underlying specification and no ability to manipulate the view.

Humans routinely resolve such ambiguity by interacting with charts: zooming, toggling traces, and inspecting exact values. Why not give agents the same capabilities? Visualization libraries such as Plotly, Vega-Lite, and ECharts render charts from structured specifications that separate data values from the rendered view. This architecture makes it possible for agents to operate on the visualization's source of truth rather than guessing from pixels. We introduce **Introspective and Interactive Visual Grounding (IVG)**, a framework that equips agents with two complementary mechanisms:

*Code and data available at https://anonymous.4open.science/r/IVG-D433 and https://anonymous.4open.science/r/iPlotBench-D0C4

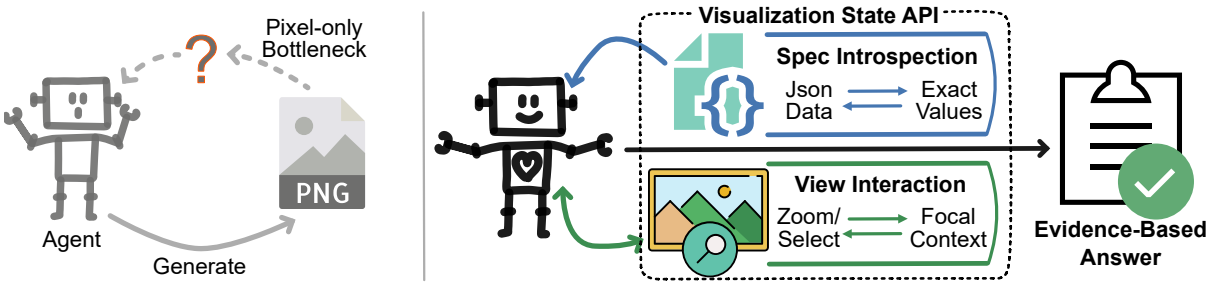

Figure 1: Overview of IVG. Rather than reasoning by interpreting rendered pixels with a VLM (left), the agent accesses the visualization state to (i) perform **spec-grounded introspection** over the chart specification for deterministic evidence and (ii) perform **view-grounded interaction** (e.g., zoom, toggle, select) to obtain focal context in visually ambiguous regions. This transforms agents from passive observers into active owners of their visualizations.

- **Spec-grounded introspection** provides direct access to the chart's underlying specification, allowing agents to verify exact values and attributes.
- **View-grounded interaction** provides tools for manipulating the view (zoom, pan, toggle, select), allowing agents to resolve ambiguity in dense or overlapping regions. Each interaction produces *focal context*—axis ranges, trace indices, or selected points—that tells the agent where to look in the specification.

Together, these mechanisms enable grounded reasoning at two levels. Between agent and data, IVG transforms agents from passive observers into active explorers of their own visualizations. Between human and agent, the same visualization state serves as a shared communication channel: a user's interactions provide context that the agent can interpret and respond to, without the user needing to verbally describe what they see (Figure 1). IVG is implemented as a Visualization State API using Model Context Protocol (MCP) tools, allowing any compatible agent to access these capabilities.

Evaluating such capabilities is challenging. Existing benchmarks like ChartQA (Masry et al., 2022) and ChartMimic (Yang et al., 2025) treat charts as static images and rely on VLM-based evaluation, but VLMs make the same perceptual errors we want to avoid. We introduce **iPlotBench**, a benchmark of 500 interactive Plotly figures with 6,706 binary questions and ground-truth specifications, designed for full deterministic evaluation. Our metric, **Semantic Structural Similarity**, evaluates chart recreation by directly comparing agent-produced and ground-truth specifications.

In summary, our contributions are as follows:

1. **IVG Toolkit:** A set of MCP tools that let agents query chart specifications and manipulate views, enabling deterministic reasoning over visualizations.
2. **iPlotBench:** A benchmark of 500 interactive Plotly figures with 6,706 binary questions, ground-truth specifications, and Semantic Structural Similarity for deterministic evaluation.
3. Our experiments on Claude Haiku 4.5 and Qwen demonstrate that spec-grounded introspection improves data reconstruction fidelity (S_{Data}: 0.88→0.90), while view-grounded interaction yields the largest QA accuracy gains (+6.7%) on questions involving complex overlapping geometries. The combination achieves the highest overall QA accuracy.
4. Beyond controlled evaluation, we deploy IVG in three agent workflows: real-time collaboration, autonomous exploration, and ML solution search. These deployments demonstrate that spec-grounded introspection and view-grounded interaction serve as general-purpose capabilities across diverse analytical tasks.

2 Related Work

Chart Understanding and QA. Benchmarks like FigureQA (Kahou et al., 2018) and DVQA (Kafle et al., 2018) establish the task of Visual Question Answering (VQA) on synthetic plots. Subsequent datasets like ChartQA (Masry et al., 2022) and PlotQA (Methani et al., 2020) extend this to real-world charts requiring complex reasoning. However, these approaches treat charts as static pixel arrays, forcing models to rely on visual approximations. Our work moves beyond static pixel interpretation by enabling agents to query chart specifications and interact with views directly.

Chart-to-Code and Recreation. Existing work focuses on agents that generate code to render charts. ChartMimic (Yang et al., 2025), Plot2code

(Wu et al., 2025), Chart2Code53 (Niu et al., 2025) and ChartCoder (Zhao et al., 2025) evaluate agents on their ability to visually reproduce reference charts. However, these benchmarks rely on VLM-based evaluation (e.g., GPT-4V) or pixel-level metrics, which cannot verify whether the underlying data values are correct. We introduce *Semantic Structural Similarity*, a metric that deterministically evaluates chart reconstruction by directly comparing the agent-produced Plotly JSON against the released ground-truth specification.

Tool-Integrated Coding Agents. Coding agents (Zhang et al., 2024a; Yang et al., 2024; Guo et al., 2025; Wang et al., 2024; Zhang et al., 2024b; Xia et al., 2024; Schick et al., 2023; Yao et al., 2022) demonstrate the power of structured state access for code-related tasks, using interfaces such as terminals, file systems, and linters. However, these agents lack structured access to their visual outputs: a generated chart is interpreted back through pixels, discarding the specification that produced it. IVG extends structured state access to visualization.

Interactive Visual Analytics. The principle that analysts should interact with visualizations rather than passively observe them is well established in the visualization community. Systems such as Voyager (Wongsuphasawat et al., 2016, 2017) and Draco (Yang et al., 2023) leverage visualization grammars to generate chart designs from specifications. Conversely, IVG helps agents reason about existing charts by reading specifications back and returning structured state that agents can compose with specification queries.

3 IVG Design

IVG enables agents to shift reasoning from probabilistic pixel inspection to deterministic, structured access. It builds on a key property of modern visualization libraries: libraries such as Plotly, Vega-Lite, and ECharts render charts from structured specifications that encode exact data values and visual attributes (Satyanarayan et al., 2016; Zhu, 2013; Wickham, 2011; Wilkinson, 2011). As illustrated in Figure 2, IVG combines *spec-grounded introspection* (reading the underlying specification) with *view-grounded interaction* (changing the rendered view to elicit structured state) into an iterative workflow for resolving visual ambiguity.

Spec-grounded introspection gives agents direct access to the JSON specification underlying a chart. This specification encodes the complete chart state: data arrays, bindings, and visual attributes. Rather than estimating values from pixels, agents query the specification directly. To verify whether a bar represents the maximum, the agent extracts the data array and computes the answer symbolically. To confirm a legend entry, the agent reads the corresponding field rather than interpreting rendered text. This directness matters because pixel-based perception is inherently approximate. Two bars of similar height may be indistinguishable visually but differ meaningfully in value; the specification reveals the exact numbers. By grounding verification in structured data, agents avoid the estimation errors and hallucinations that arise from pixel interpretation.

While spec-grounded introspection provides exact values, it requires knowing which part of the specification to examine. A chart may contain hundreds of points across multiple traces; retrieving and reasoning over the full specification is often unnecessary. View-grounded interaction addresses this challenge by generating **focal context**—explicit view-state parameters that constrain the search space. For example, zooming or panning yields precise **axis ranges** (e.g., `layout.xaxis.range`); toggling a trace identifies a **trace/curve index** (e.g., `curve_number`); selecting a region yields a **data-space bounding box** and the points it contains. These interaction outputs act as **search keys**: instead of parsing the entire specification, the agent can filter the JSON returned by introspection to the subset relevant to its current focus. This transforms interaction history into a structured query mechanism that bridges visual attention and specification data.

These two mechanisms work together in a unified workflow, as shown in figure 2. If the relevant data can be directly located, introspection suffices. Otherwise, the agent first uses interaction to obtain focal context, then queries the specification within that scope. For example, to determine whether two lines intersect, an agent zooms into the region where they appear close, then queries for exact coordinates. The workflow is iterative: the agent refines its focus until sufficient evidence emerges

The Visualization State API is implemented as a set of MCP tools, summarized in Table 1. The interaction tools cover the three fundamental view-manipulation primitives identified by existing work (Brehmer and Munzner, 2013) : navigate (relayout), select (selected), and filter (legendclick). These tools operate at the figure level and propagate

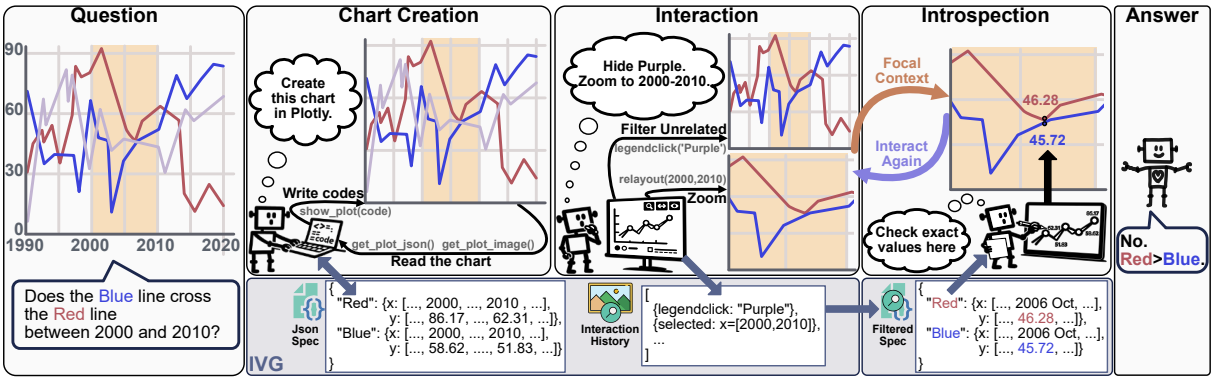

Figure 2: IVG workflow on a concrete example. Given a question, the agent recreates the chart (Chart Creation), gaining access to its specification and interaction tools. It then alternates between Interaction (toggling traces, zooming to generate focal context) and Introspection (querying the specification for exact values in the focused region) until sufficient evidence is gathered to produce a grounded answer.

Category	Tool	Args	Function
Base	show_plot	plotly_codes	Creates figure; returns plot_id
	get_plot_image	plot_id	Returns current view as PNG
Introspection	get_plot_json	plot_id	Returns full Plotly specification
Interaction	relayout	plot_id, x/y_range	Zoom/Pan axis bounds
	legendclick	plot_id, curve_number	Toggle trace visibility
	selected	plot_id, x/y_range	Returns points in region
	query_interactions	plot_id	Returns interaction history

Table 1: Visualization State API grouped by category.

across subplots with shared axes, supporting multi-view analysis. To ensure valid evaluation, agents can only access specifications and interaction state for figures they create (via returned `plot_ids`), and they do not receive ground-truth datasets or reference solutions through the tool interface.

4 iPlotBench

Existing visualization benchmarks, such as ChartMimic (Yang et al., 2025) and FigureQA (Kahou et al., 2018), treat charts as static images. This makes it difficult to evaluate agents that (1) use view interaction (e.g., zooming into dense regions or toggling traces) and (2) require a rendered chart specification for deterministic, structure-level evaluation, rather than pixel- or VLM-based scoring.

To bridge this gap, we introduce iPlotBench, a benchmark for evaluating **spec-grounded introspection** and **view-grounded interaction** in visualization agents. iPlotBench procedurally generates interactive Plotly figures and releases their ground-truth specifications, enabling deterministic evaluation of chart recreation and visual reasoning.

4.1 Dataset Construction

The benchmark has 500 figures balanced across five chart types: Line, Dot-Line, Vertical Bar, Horizontal Bar, and Pie. We adapt the procedural generation pipeline from FigureQA (Kahou et al., 2018) and convert each instance into an interactive Plotly figure. All figures support standard interactions (zoom/pan, legend toggling, selection), enabling agents to actively probe ambiguous regions.

Each figure is paired with a rendered PNG input and its ground-truth Plotly JSON specification. The dataset contains 6,706 binary questions across 15 templates, with a rigorous 50/50 Yes/No balance per figure to prevent class-imbalance shortcuts (full statistics in Table 9).

The 15 question templates span three types: *Aggregation* (min/max/median, e.g., "Is Blue Violet the minimum?"), *Comparison* (value ordering, e.g., "Is Blue less than Red?"), and *Topology* (intersection/smoothness, e.g., "Does Blue intersect Red?").

4.2 Tasks

We define two evaluation tasks that target complementary capabilities. In our evaluation, the tasks are sequential within the same agent session: the agent first recreates the chart (Task 1), then answers questions based on the recreated chart (Task 2).

Task 1: Chart Recreation. Given a static reference image, the agent must generate the underlying Plotly JSON specification that faithfully recreates the chart. Unlike previous "chart-to-code" tasks that allow arbitrary styling, this task requires precise structural grounding: mapping visual elements (e.g., bar heights, colors, labels) to exact attributes in the JSON schema. Success in this task demonstrates the agent's ability to accurately reconstruct the structured specification from a visual reference.

Semantic structural similarity For Task 1, we propose Semantic Structural Similarity, which evaluates semantic correctness of agent-generated

Plotly figures via direct JSON inspection. Unlike recent benchmarks such as ChartMimic which rely on VLM-based scoring (e.g., GPT-4o) to assess data trends and can be sensitive to VLM hallucinations, our metrics provide deterministic, symbolic verification. Exact JSON matching is impractical due to trace reordering and variable sampling density (e.g., different discretizations of the same curve). We therefore align predicted traces (P) to ground-truth traces (T) using Hungarian matching on data geometry: we compute pairwise Chamfer distances and solve the linear assignment to obtain a set of matched trace pairs $Pairs$. All trace-based metrics use the shared denominator $\max(|T|, |P|)$ to penalize extra/missing traces; unmatched traces contribute zero.

We intentionally omit separate layout and clarity metrics: in our single-plot figures, layout is largely captured by role-aware text, while clarity is subjective and is typically reflected as data/style errors. **Chart Type** (S_{Type}) We verify the trace type (e.g., *scatter*, *bar*) for each matched pair, scoring 1 for a match and 0 otherwise, normalized by $\max(|T|, |P|)$.

Data Accuracy (S_{Data}) We treat each trace as a point cloud and compute Chamfer distance between matched pairs, robust to sampling differences. Within Chamfer, we use Euclidean distance on range-normalized numerical dimensions and a Jaccard-based penalty for categorical/text dimensions (e.g., bar labels):

$$S_{Data} = \sum_{(t,p)\in Pairs} \frac{e^{-\lambda D_{Chamfer}(t,p)}}{\max(|T|, |P|)} \quad (1)$$

We set $\lambda = 5$ to balance sensitivity: at $\lambda = 1$, even worst-case reconstructions score 0.37, whereas at $\lambda = 10$, minor noise is penalized too heavily. A sensitivity analysis over λ confirms that configuration rankings remain consistent across all values (Appendix D).

Text Correctness (S_{Text}) We use a role-aware text metric to avoid "right text, wrong place" errors. Text is bucketed into semantic roles $\mathcal{R}$ (Title, Axis, Legend, Data/Annotations), and compared with a fuzzy Jaccard similarity:

$$S_{Text} = \sum_{r\in\mathcal{R}} w_r \cdot \mathrm{Sim}(Text_T^r, Text_P^r) \quad (2)$$

Visual Style (S_{Style}) We average style similarity over matched traces and properties $\mathcal{P} = \{color, mode, symbol, size, dash, width\}$:

$$S_{Style} = \frac{\sum_{(t,p)\in Pairs} \sum_{prop\in\mathcal{P}} S_{prop}(t,p)}{\max(|T|, |P|)\,|\mathcal{P}|} \quad (3)$$

Here $S_{prop}(t, p)$ compares the values of property $prop$ in traces t and p. For color, we compare in CIELAB space (and use EMD for color arrays); categorical attributes use exact match and numerical attributes use normalized error, comparing against Plotly defaults when an attribute is omitted.

Task 2: Visual QA. Given a chart and a binary question, the agent must answer yes or no; we report standard accuracy. Many questions are designed to be visually ambiguous in a static view (e.g., detecting intersections in overlapping line charts or identifying the minimum in a dense bar chart). We use binary questions since they enable fully deterministic evaluation without VLM-based or human judgment. Open-ended reasoning with IVG is demonstrated in Section 6.

5 Evaluation

5.1 Evaluation Setup

We compare four configurations: *Vision* (no tools), *+Inter* (interaction-only tools), *+Intro* (introspection-only via *get_plot_json*), and *Full* (both introspection and interaction tools). Agents follow the two-task protocol in Section 4.2. Tools are available but tool use is optional; all configurations share the same prompt template and tool schemas (Appendix B).

Models. Our main ablation uses Claude Haiku 4.5. We also evaluate the Qwen-VL family (Table 2) for model scaling analysis. Because provider runtimes may include undisclosed system prompts, we emphasize *within-model* deltas (e.g., Vision vs Full) for causal claims.

Model	Architecture	Active Params
Qwen3-VL-30B-A3B	MoE	3B
Qwen3-VL-32B	Dense	32B
Qwen3-VL-235B-A22B	MoE	22B
Qwen-VL-Max	multimodal VLM	>72B (est.)

Table 2: Qwen-VL models evaluated.

Results summary. We evaluate four agent configurations on iPlotBench. Within each episode, the agent first recreates the chart from the static reference image (Task 1), then answers binary questions by reasoning over its recreated interactive figure (Task 2). Overall, spec-grounded introspection primarily improves chart reconstruction, while

combining introspection with interaction yields the strongest QA performance.

5.2 Task 1: Chart Recreation

As shown in Table 3, equipping agents with introspection (+*Intro*) drives the most significant gains in semantic reconstruction, achieving the highest scores in trace typing (S_{Type}), data fidelity (S_{Data}), and style (S_{Style}). While interaction (+*Inter*) slightly outperforms in text extraction (S_{Text}), likely by exposing occluded labels, it provides limited structural benefit. Notably, the *Full* agent does not surpass +*Intro*: adding interaction expands the action space, and unnecessary view changes can distract from spec-level fixes during reconstruction.

Metric	Vision	+Inter	+Intro	Full
S_{Type}	0.9542	0.9742	**0.9807**	0.9744
S_{Data}	0.8847	0.8935	**0.9016**	0.8907
S_{Text}	0.9755	**0.9903**	0.9847	0.9877
S_{Style}	0.8699	0.8708	**0.8749**	0.8679

Table 3: Semantic Structural Similarity for Task 1.

5.3 Task 2: Visual QA

Table 4 highlights the synergy of introspection and interaction: the *Full* agent achieves the highest overall accuracy (0.8062). The benefits of interaction are most pronounced for Line and Dot-Line charts, where zooming and selective inspection help resolve overlaps. For Bar and Pie charts, +*Intro* achieves the best accuracy, suggesting that direct spec access is sufficient when the geometry is less ambiguous.

Figure Type	Vision	+Inter	+Intro	Full
Line	0.7012	0.7030	0.7085	**0.7292**
Dot-Line	0.7155	0.7178	0.7353	**0.7522**
Vertical Bar	0.7993	0.8261	**0.8529**	0.8434
Horizontal Bar	0.8354	0.8388	**0.8623**	0.8463
Pie	0.8986	0.9019	**0.9197**	0.9146
OVERALL	0.7783	0.7850	0.8023	**0.8062**

Table 4: Question-level accuracy for Task 2.

5.4 Deconfounding Reconstruction Quality

Since Task 2 is performed on the chart recreated in Task 1, QA accuracy is bounded by reconstruction fidelity. To isolate reasoning, we condition on well-reconstructed figures ($S_{Data} \geq 0.9$). Table 5 reports per-figure accuracy, which averages accuracy within each figure before averaging across figures to avoid overweighting chart types with more questions.

Config	Per-figure Acc	Cond. Acc ($S_{Data} \geq 0.9$)
Vision	0.7941	0.8321
+Inter	0.8013	0.8538
+Intro	0.8190	0.8606
Full	**0.8199**	**0.8637**

Table 5: Per-figure and conditional QA accuracy.

Table 6 reveals complementary strengths: +*Intro* excels at Comparison questions where direct spec access enables precise value retrieval, while *Full* performs best on Topology questions where interaction helps resolve spatial relationships.

Config	Aggregation	Comparison	Topology
Vision	0.8354	0.8391	0.7378
+Inter	0.8591	0.8522	0.8043
+Intro	0.8846	**0.8555**	0.7567
Full	**0.8864**	0.8514	**0.8052**

Table 6: QA accuracy on well-reconstructed figures. Aggregation: min/max/median; Comparison: less/greater; Topology: intersection, smoothness, AUC.

Config	Chart Type	`relayout` (%)	`legend` (%)	`selected` (%)	Total (calls)
+Inter	Line	92 %	52	38	349
	Dot-Line	88	53	29	343
	Vertical Bar	51	3	12	82
	Horizontal Bar	59	2	11	92
	Pie	1	31	5	62
Full	Line	32	29	19	112
	Dot-Line	25	25	12	90
	Vertical Bar	13	3	5	27
	Horizontal Bar	15	1	5	25
	Pie	0	7	1	13

Table 7: View-manipulation tool usage by chart type and configuration. Columns show the percentage of sessions invoking each tool; Total is the number of calls.

5.5 IVG Usage Analysis

To understand when and why each mechanism helps, Table 7 reports interaction tool usage across agent sessions by chart type and configuration.

Agents use zoom/pan (relayout), trace toggling (legend), and region selection (selected) heavily on Line and Dot-Line charts, where overlapping traces create spatial ambiguity, but rarely on Bar and Pie charts, where values are visually distinct. When introspection is also available (Full), total interaction calls drop sharply, indicating that agents rely on direct spec access instead of view manipulation when both options are available. This pattern is consistent across Tasks 1 and 2, and explains why

Full underperforms +*Intro* on S_{Data} (in Table 3): unnecessary interactions consume the tool-call budget without providing information beyond what the specification already contains.

5.6 Model Scaling Analysis

We examine how IVG benefits scale with model capability using the Qwen-VL family. Table 8 presents Task 1 (S_{Data}) and Task 2 (QA accuracy) for Vision and Full configurations. We observe that IVG benefits scale with model capability. Models with larger active parameters (Qwen3-VL-235B-A22B, Qwen-VL-Max) show consistent improvements, while smaller models face challenges in effectively orchestrating tool use alongside their primary reasoning tasks.

Model	S_{Data} (V)	S_{Data} (F)	QA (V)	QA (F)
Qwen3-VL-30B-A3B	0.852	0.642	0.948	0.870
Qwen3-VL-32B	0.881	**0.883**	0.956	0.933
Qwen3-VL-235B-A22B	0.878	**0.904**	0.951	**0.960**
Qwen-VL-Max	0.916	**0.922**	0.971	**0.973**

Table 8: Task 1 and Task 2 results for Qwen-VL models. V=Vision (no tools), F=Full (IVG tools).

Conditional analysis in Appendix E confirms these trends hold when controlling for reconstruction quality ($S_{Data} \geq 0.9$), with larger models showing consistent gains and their largest improvements on Topology questions where view-grounded interaction resolves spatial ambiguity . Effective use of IVG requires models with sufficient reasoning capacity to orchestrate tool calls alongside their primary tasks; in our evaluation, models with approximately 20B or more active parameters benefit consistently. For frontier models approaching ceiling performance, IVG's primary value shifts from accuracy improvement to auditability, providing explicit and traceable evidence for each reasoning step.

6 IVG-Enabled Agents

Beyond benchmark evaluation, we demonstrate that IVG enables a new class of visualization agents that can collaborate with users, explore data autonomously, and ground complex decisions in verifiable evidence.

Real-time collaboration. In real-time collaboration, IVG bridges vague human gestures and precise machine reasoning. When a user points at a chart and asks "Why did it drop here?" (Figure 3), the question is inherently ambiguous. But the user's interaction history provides the missing context: the system captures the user's selection as focal context, telling the agent exactly which region the user is referring to. The agent retrieves the relevant data and provides a grounded response, enabling natural dialogue without the friction of verbal description. Details are in Appendix F.1.

Autonomous exploration with interactive deliverables. In autonomous exploration, agents independently analyze datasets and produce evidence-grounded reports. The key challenge is verification: how can an agent ensure its claims are supported by data, rather than hallucinated? Figure 4 illustrates the IVG cycle. Given a dataset, the agent forms initial claims, creates visualizations, and applies IVG to verify them: introspection retrieves exact values, while interaction focuses attention on relevant regions. Claims supported by deterministic evidence become grounded; the exploration process reveals new questions that feed back into the cycle. The final report contains only grounded claims, each traceable to visualization evidence. Crucially, the report remains interactive: users can ask follow-up questions grounded in their current view. Details and prompts are in Appendix F.2.

ML solution search. The same IVG mechanisms extend from reporting to decision-making. In machine learning workflows, agents search over candidate solutions, each producing visualizations (training curves, confusion matrices, performance comparisons). Rather than estimating from rendered images, the agent retrieves exact values from each candidate's visualization state, transforming search decisions from heuristic judgments into evidence-grounded choices (details in Appendix F.3).

User study. To evaluate whether IVG improves agent-assisted analysis in practice, we conducted a blinded within-subjects study with 12 researchers with data analysis experience. Each participant used both an IVG-enabled agent and a vision-only baseline (in randomized order) to explore the same dataset, then completed a survey comparing the two. 11 of 12 participants preferred the IVG-enabled agent for interactive data exploration. Participants rated the preferred agent highly on citing correct values (4.5/5) and grounding claims in visible evidence (4.4/5). When compared against alternatives including general-purpose chatbots and manual analysis, 9 participants rated their preferred agent as the best option. Full survey design and

Figure 3: Real-time collaboration. The user points at a region of interest and asks a question; IVG captures this as focal context, enabling the agent to understand what the user is referring to.

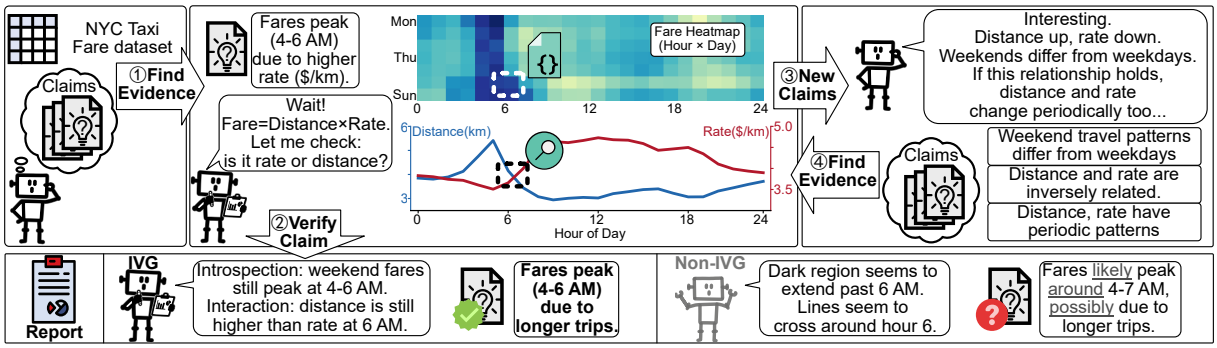

Figure 4: Autonomous exploration: IVG enables evidence-grounded analysis. The agent ①finds evidence for an initial claim, ②verifies it through IVG, discovers ③new claims, and ④gathers further evidence. Bottom: IVG yields a precise, verified claim (4-6 AM); without IVG, the agent produces a vague, incorrect one (4-7 AM).

per-participant results are in Appendix F.4.

7 Discussion

Our benchmark results demonstrate that IVG addresses the Pixel-Only Bottleneck, and Section 6 shows these benefits extend to diverse real-world scenarios.

Tool synergy and focal context. A critical insight from our study is the synergy between spec-grounded introspection and view-grounded interaction. Introspection provides exact values but can overwhelm an agent with irrelevant structural details. Interaction provides focal context that localizes which subset of the specification is relevant, mimicking how a human analyst focuses attention before inspecting values. As shown in Section 5.5, agents learn this balance: when introspection is available, total interaction calls drop by 71% (Table 7), as agents substitute spec queries for view manipulation. However, when agents fail to make this substitution, unnecessary interactions consume the tool-call budget without providing new information, explaining why Full occasionally underperforms +*Intro* on S_{Data} (Table 3).

Generality beyond plotly. IVG builds on a property shared across visualization ecosystems: the separation of specification from rendering. Any library that maintains a structured specification can support spec-grounded introspection; any library that exposes interaction events as structured state can support view-grounded interaction. Plotly provides complete support for both, making it a natural first target. Other libraries offer partial coverage: Vega-Lite and ECharts maintain structured specifications with interaction events, while Matplotlib supports specification access but lacks native interaction. As visualization agents expand to these and other libraries, the same two-mechanism decomposition can guide tool design, identifying what specifications agents can read and what interactions produce structured context.

8 Conclusion

We present Introspective and Interactive Visual Grounding (IVG), a framework that gives visualization agents structured access to their own outputs. Evaluation on iPlotBench shows that introspection and interaction are complementary: one provides deterministic evidence, the other provides focal context. Tool usage analysis confirms that agents balance these mechanisms based on chart complexity. Beyond the benchmark, IVG enables agents to collaborate with users, explore data autonomously, and ground decisions in verifiable evidence.

9 Limitations

Our empirical validation uses Plotly as the target library. While the underlying abstraction generalizes to other grammar-based libraries (Section 7), adapting the Visualization State API to each library's specification format and event model requires engineering effort. Our benchmark evaluates IVG through binary QA, which enables fully deterministic evaluation but does not capture the full range of analytical tasks. Section 6 demonstrates IVG on open-ended tasks qualitatively; a systematic open-ended benchmark is future work.

References

Matthew Brehmer and Tamara Munzner. 2013. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385.

Jinyao Guo, Chengpeng Wang, Xiangzhe Xu, Zian Su, and Xiangyu Zhang. 2025. RepoAudit: An autonomous LLM-agent for repository-level code auditing. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 21083–21100. PMLR.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Robert Tang, Xiangtao Lu, and 9 others. 2025. Data interpreter: An LLM agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19796–19821, Vienna, Austria. Association for Computational Linguistics.

Chaoya Jiang, Haiyang Xu, Mengfan Dong, Jiaxing Chen, Wei Ye, Ming Yan, Qinghao Ye, Ji Zhang, Fei Huang, and Shikun Zhang. 2024. Hallucination augmented contrastive learning for multimodal large language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27036–27046.

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656.

Samira Ebrahimi Kahou, Adam Atkinson, Vincent Michalski, Ákos Kádár, Adam Trischler, and Yoshua Bengio. 2018. Figureqa: An annotated figure dataset for visual reasoning. In *International Conference on Learning Representations (ICLR) Workshop*.

Prannay Kaul, Zhizhong Li, Hao Yang, Yonatan Dukler, Ashwin Swaminathan, CJ Taylor, and Stefano Soatto. 2024. Throne: An object-based hallucination benchmark for the free-form generations of large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27228–27238.

Sicong Leng, Hang Zhang, Guanzheng Chen, Xin Li, Shijian Lu, Chunyan Miao, and Lidong Bing. 2024. Mitigating object hallucinations in large vision-language models through visual contrastive decoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13872–13882.

Bingxuan Li, Yiwei Wang, Jiuxiang Gu, Kai-Wei Chang, and Nanyun Peng. 2025. Metal: A multi-agent framework for chart generation with test-time scaling. *CoRR*.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 292–305.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pages 2263–2279.

Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. In *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pages 1527–1536.

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

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object hallucination in image captioning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4035–4045.

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Wonduk Seo, Seungyong Lee, Daye Kang, Hyunjin An, Zonghao Yuan, and Seunghyun Lee. 2025. Automated visualization code synthesis via multi-path reasoning and feedback-driven optimization. *Preprint*, arXiv:2502.11140.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, and 1 others. 2024. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*.

Hadley Wickham. 2011. ggplot2. *Wiley interdisciplinary reviews: computational statistics*, 3(2):180–185.

Leland Wilkinson. 2011. The grammar of graphics. In *Handbook of computational statistics: Concepts and methods*, pages 375–414. Springer.

Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):649–658.

Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 2648–2659, New York, NY, USA. Association for Computing Machinery.

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

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. 2024. Agentless: Demystifying llm-based software engineering agents. *CoRR*.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran XU, Xinyu Zhu, Siheng Li, Yuxiang Zhang, and 1 others. 2025. Chartmimic: Evaluating lmm's cross-modal reasoning capability via chart-to-code generation. In *The Thirteenth International Conference on Learning Representations*.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Junran Yang, Péter Ferenc Gyarmati, Zehua Zeng, and Dominik Moritz. 2023. Draco 2: An extensible platform to model visualization design. In *2023 IEEE Visualization and Visual Analytics (VIS)*, pages 166–170.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024a. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13643–13658.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. 2024b. Autocoderover: Autonomous program improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1592–1604.

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

Nick Qi Zhu. 2013. *Data visualization with D3. js cookbook*. Packt Publishing Ltd.

A Visualization State API: MCP Tool Schemas

The Visualization State API is implemented as a set of Model Context Protocol (MCP) tools. MCP provides a standardized interface for language model agents to invoke external functions with typed arguments and structured JSON responses. All tools return JSON-formatted output for deterministic parsing.

A.1 Base Tools

show_plot Creates a Plotly figure from agent-generated Python code and registers it for subsequent operations.

- **Arguments:**
  - plotly_codes (string, required): Python code defining a variable fig as a Plotly figure object. Must include necessary imports.
- **Returns:** A unique identifier for referencing this figure in subsequent tool calls.

  {plot_id: int}

get_plot_image Captures the current rendered view of a figure as a PNG screenshot.

- **Arguments:**
  - plot_id (integer, required): Plot identifier from show_plot.
  - interaction_id (integer, optional): Specific interaction snapshot ID. If omitted, returns the latest view state.
- **Returns:** Absolute path to the PNG file, which can be passed to the VLM for pixel-based inspection.

  {image_path: str}

A.2 Spec-Grounded Introspection

get_plot_json Retrieves the full Plotly specification of the agent's constructed figure, enabling deterministic verification of data values and visual encodings.

- **Arguments:**
  - plot_id (integer, required): Plot identifier from show_plot.
- **Returns:** The complete Plotly JSON specification, where data contains trace objects and layout contains axis and styling configurations.

  {data: [...], layout: {...}}

Trust Boundary: This tool exposes *only* the agent's own constructed figure specification—not the ground-truth dataset or reference solution.

A.3 View-Grounded Interaction

relayout Programmatically zooms or pans the view by setting axis ranges, providing focal context for dense or occluded regions.

- **Arguments:**
  - plot_id (integer, required): Plot identifier.
  - x_min, x_max (number, optional): X-axis range bounds.
  - y_min, y_max (number, optional): Y-axis range bounds.
- **Returns:** Success status.
- **Event Payload:** Records axis ranges.

```
{
  "xaxis.range[0]": 1981.97,
  "xaxis.range[1]": 2001.99,
  "yaxis.range[0]": 70.98,
  "yaxis.range[1]": 73.81
}
```

legendclick Toggles the visibility of a specific trace, enabling isolation of overlapping series.

- **Arguments:**
  - plot_id (integer, required): Plot identifier.
  - curve_number (integer, required): Zero-indexed trace index.
- **Returns:** Success status.
- **Event Payload:** Records the curve number. Visibility changes are captured in the associated plotly_restyle event.

```
{
  curve_number: int,
  expanded_index: int
}
```

selected Performs a box selection over a specified region and returns information about selected data points.

- **Arguments:**
  - plot_id (integer, required): Plot identifier.
  - x_min, x_max (number, optional): X-axis selection bounds.
  - y_min, y_max (number, optional): Y-axis selection bounds.
- **Returns:** Selection information.
- **Event Payload:** Records the selection range and point count.

```
{
  point_count: int,
  range: {
    x: [min, max],
    y: [min, max]
  }
}
```

query_interactions Retrieves the interaction history for a figure, enabling the agent to reason about prior view manipulations.

- **Arguments:**
  - plot_id (integer, required): Plot identifier.
  - event_type (string, optional): Filter by event type (init, relayout, legendclick, selected).
- **Returns:** Chronological list of interaction events, each with:

- id: Unique interaction ID (can be passed to get_plot_image to retrieve historical snapshots).
- event_type: One of init, relayout, legendclick, selected.
- payload: Event-specific details (e.g., axis ranges for relayout, visibility state for legendclick).
- has_screenshot: Whether a screenshot was captured for this interaction state.

```
{
  events: [
    {id, event_type, payload, has_screenshot},
    ...
  ]
}
```

B Controller and Evaluation Protocol

This section details the prompt templates, stop conditions, and execution environment used in our evaluation.

B.1 Prompt Templates

All agent configurations share identical prompt templates. We use minimal prompts to avoid biasing tool usage patterns.

Task 1: Chart Recreation The agent receives the reference image (base64-encoded PNG) along with the following text prompt:

> Read ./input.png and recreate this plot.
>
> Output the Plotly figure as JSON with "data" and "layout" keys: {"data": [...], "layout": {...}}

For configurations with tools enabled, the agent may invoke Visualization State API tools before producing the final JSON output. For the Vision baseline, the agent directly outputs JSON without tool access.

Task 2: Visual QA For each binary question, the agent receives:

> <question text>
>
> Reply with ONLY a single digit: 0 or 1

Task 2 questions are answered within the same session as Task 1, meaning the agent retains context from chart recreation and can reference its constructed figure state.

B.2 Stop Conditions

Each *task invocation* is bounded by two termination conditions:

- **Tool round limit:** Maximum 5 tool-calling rounds per invocation. If the model continues requesting tool calls after 5 rounds, the invocation terminates with partial results.
- **API timeout:** 120 seconds per API call. If the model does not respond within this window, the invocation is marked as failed.

Invocation structure: An episode consists of multiple invocations within a persistent session. Task 1 (chart recreation) is a single invocation. Task 2 questions are each processed as *separate invocations*, meaning each question receives its own 5-round tool budget. Critically, the session context (conversation history and figure state) persists across all invocations, allowing the agent to reference its Task 1 reconstruction when answering Task 2 questions.
An invocation terminates successfully when at least one valid answer is extracted: a Plotly JSON object with non-empty data array for Task 1, or a parseable 0/1 response for Task 2.

B.3 Execution Environment

Claude Agents (Main Ablation). Claude 4.5 Haiku agents run in isolated Docker containers based on "python:3.10-slim". Each container includes:

- Python 3.10 with Plotly for figure generation
- Kaleido for server-side PNG rendering
- MCP server implementing the Visualization State API
- Isolated filesystem with read-only access to reference images

The containerized environment ensures reproducibility and prevents agents from accessing ground-truth data or other test cases.

Qwen-VL Agents (Scaling Study). Qwen-VL models are accessed via OpenAI-compatible APIs (local vLLM or OpenRouter). The same MCP tools are available, with images passed as base64-encoded PNGs in the image_url message format. Key differences from the Claude setup:

- No Docker isolation (API-based execution)
- Session state managed by Python runner script
- Same prompt templates and tool schemas
- Vendor runtimes may include undisclosed system prompts

To account for potential confounds from vendor-specific behaviors, we emphasize within-model comparisons (Vision vs Full) rather than cross-model rankings.

B.4 Tool Schema Availability

In all tool-enabled configurations (+Inter, +Intro, Full), the complete tool schema is provided to the model at session start. Tools are *available* but their use is *optional*—models autonomously decide when and whether to invoke tools based on task requirements. This design isolates model capability from prompt engineering that might bias tool usage.

C Dataset Statistics

Table 9 provides the full breakdown of iPlotBench by chart type.

Type	#Figures	#Questions	Q/Fig
Line	100	1,504	15.0
Dot-Line	100	1,690	16.9
Vertical Bar	100	1,142	11.4
Horizontal Bar	100	1,187	11.9
Pie	100	1,183	11.8
Total	500	6,706	13.4

Table 9: Dataset statistics. The benchmark features a balanced distribution of questions across five chart types, with an average of 13.4 questions per figure.

D λ Sensitivity Analysis

We evaluate the sensitivity of S_{Data} to the exponential penalty parameter λ in Equation 2 by recomputing scores with $\lambda \in \{1, 3, 5, 7, 10\}$ across both model families.

Table 10 reports results for Claude Haiku 4.5. The ranking of configurations is preserved across all λ values: *+Intro* consistently achieves the highest S_{Data}, followed by Full, +Inter, and Vision.

Table 11 reports results for the Qwen-VL family. Full consistently improves over Vision for models with sufficient capacity (Qwen3-VL-235B-A22B, Qwen-VL-Max) and degrades for the smallest model, consistent with the tool capability threshold discussed in Section 5.6. Minor ranking variations occur at boundary cases (e.g., Qwen3-VL-32B at $\lambda = 7$, margin of 0.004), but overall patterns hold across all λ values.

Config	$\lambda=1$	$\lambda=3$	$\lambda=5$	$\lambda=7$	$\lambda=10$
Vision	0.9603	0.9191	0.8847	0.8548	0.8158
+Inter	0.9719	0.9297	0.8935	0.8618	0.8203
+Intro	**0.9742**	**0.9352**	**0.9016**	**0.8720**	**0.8327**
Full	0.9696	0.9268	0.8907	0.8592	0.8182

Table 10: S_{Data} across λ values for Claude Haiku 4.5 configurations.

Model	Config	$\lambda=1$	$\lambda=3$	$\lambda=5$	$\lambda=7$	$\lambda=10$
Qwen3-VL-30B-A3B	Vision	0.959	0.899	0.852	0.813	0.765
	Full	0.783	0.667	0.642	0.630	0.616
Qwen3-VL-32B	Vision	0.939	0.907	0.881	0.859	0.831
	Full	0.959	0.916	0.883	0.855	0.821
Qwen3-VL-235B-A22B	Vision	0.918	0.897	0.878	0.860	0.836
	Full	**0.962**	**0.929**	**0.904**	**0.881**	**0.851**
Qwen-VL-Max	Vision	0.965	0.939	0.916	0.896	0.868
	Full	**0.971**	**0.945**	**0.922**	**0.901**	**0.874**

Table 11: S_{Data} across λ values for Qwen-VL models (Vision vs. Full).

E Model Scaling: Conditional Analysis

To isolate verification capability from reconstruction quality, Table 12 reports QA accuracy conditioned on successful reconstruction ($S_{Data} \geq 0.9$).

Model	Cond. (V)	Cond. (F)	Δ
Qwen3-VL-30B-A3B	0.959	0.900	-0.059
Qwen3-VL-32B	0.973	0.964	-0.009
Qwen3-VL-235B-A22B	0.961	**0.966**	$+0.005$
Qwen-VL-Max	0.983	**0.985**	$+0.002$

Table 12: Conditional QA accuracy ($S_{Data} \geq 0.9$) for Qwen-VL models.

Larger models (Qwen3-VL-235B-A22B, Qwen-VL-Max) show consistent gains even when controlling for reconstruction quality, with Qwen-VL-Max approaching ceiling performance. The smaller Qwen3-VL-30B-A3B model does not benefit from tools in this setting, suggesting that effective tool orchestration requires sufficient model capacity.

Table 13 reveals which question types benefit most from IVG across model scales.

Capable models (Qwen3-VL-235B-A22B, Qwen-VL-Max) show their largest gains on Topology questions (+0.008), where view-grounded

Model	ΔAgg	ΔComp	ΔTopo
Qwen3-VL-30B-A3B	−0.002	−0.094	−0.032
Qwen3-VL-32B	−0.003	−0.001	−0.074
Qwen3-VL-235B-A22B	+0.001	+0.007	+0.008
Qwen-VL-Max	0.000	+0.003	+0.008

Table 13: Conditional QA delta by question family ($S_{Data} \geq 0.9$). Agg=Aggregation, Comp=Comparison, Topo=Topology. Δ denotes Full − Vision.

interaction (zoom, selection) helps resolve spatial relationships. This aligns with IVG's design goal of providing focal context for visually ambiguous regions.

F Deployment Details

This section provides implementation details for the three deployments described in Section 6: prompt templates and interface screenshots.

F.1 Real-time Collaboration

In real-time collaboration mode, IVG bridges vague human gestures and precise machine reasoning. When a user points at a chart and asks "What's happening here?", the question is inherently ambiguous, but the user's interaction history provides the missing context. As users explore a visualization, view-grounded interaction captures each action as structured context that the agent can retrieve and interpret.

For example, a user viewing a stock market chart notices a sharp drop and asks "Why did it drop here?" Rather than verbally describing the region of interest, the user simply points and asks. The system captures the user's selection as focal context, telling the agent exactly which region the user is referring to. The agent can then retrieve the relevant data and provide a grounded response. This transforms the agent from a passive chart reader into an active collaborator that shares the user's visual attention, enabling natural dialogue without the friction of verbal description.

Figure 5 illustrates a concrete example of IVG-enabled agent working in real-time collaboration. The user clicks three timestamps on a stock price chart where Meta experienced drops and asks the agent to investigate. The agent begins by calling `query_interactions` to retrieve the user's click history, identifying the exact dates and prices: April 22 at $499.15, July 30 at $694.17, and November 20 at $588.67. These values come directly from the chart specification, not from pixel estimation. The agent then uses web search to find explanations for each drop and extends the analysis to other tech giants during the same periods. This demonstrates how IVG bridges the user's visual gesture (clicking on drops) with precise, verifiable data retrieval, enabling a grounded analytical dialogue.

F.2 Autonomous Exploration

The autonomous exploration agent (Deep Plot) operates in a time-bounded loop, iteratively analyzing data until timeout. IVG serves a dual role in this cycle. First, introspection verifies whether evidence supports a claim: the agent retrieves exact values rather than estimating from rendered pixels. Second, interaction enables discovery: by manipulating views to focus on specific regions, the agent uncovers patterns that prompt new questions. The same mechanism that grounds existing claims also expands the space of inquiry.

Three prompts guide the process.

Initial Prompt. The agent begins with an exploration phase:

> Explore the working directory and find data files to analyze. You have {timeout_seconds} seconds.
> **Research process:**
>
> 1. **Look** – List files in the directory. Load any data files you find (csv, json, parquet, etc.).
> 2. **Ask** – What is this dataset about? What questions are worth investigating?
> 3. **Investigate** – Answer your questions. Support each finding with evidence (numbers or plotly visualizations).
> 4. **Synthesize** – What did you learn?

Improvement Prompt. The agent iteratively refines its analysis:

> You have {time_remaining} seconds remaining.
> **Reflect:**
>
> - Does your evidence support your findings?
> - Dig deeper. Check specific values. Use get_plot_image to view plots, and relayout/legendclick/selected to interact.
> - What new questions emerge?
>
> Continue investigating.

Final Report Prompt. After timeout, the agent produces a structured report:

> Write your **Final Report**. **Structure:**
>
> 1. **Data** – Describe the dataset
> 2. **Findings** – List each finding with evidence (numbers, tables, or plot references)
>
> **To finish:**
>
> 1. Redraw any unclear plots (fix overlapping, ensure readable).

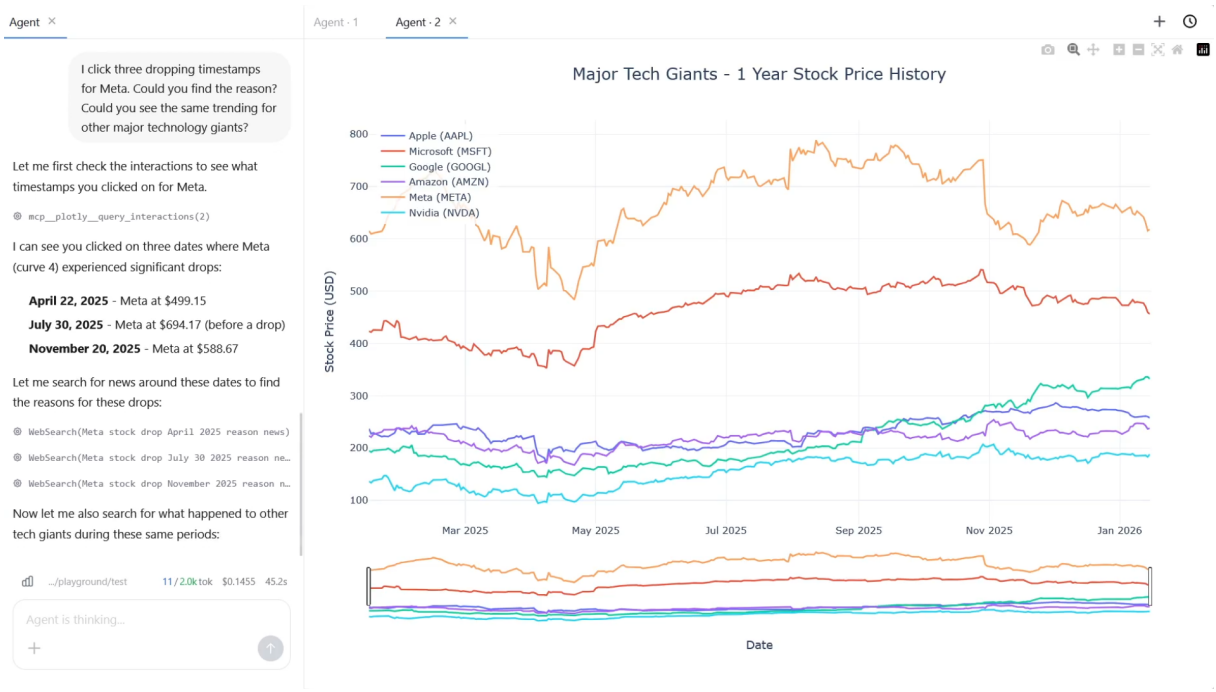

Figure 5: Real-time collaboration interface. Left: chat panel where users interact with the agent. Right: interactive Plotly visualizations created by the agent. Users can manipulate plots directly, and their interaction history informs subsequent agent responses.

2. Decide which plots are evidence and their display order (e.g., plots 4, 7, 2).
3. Write "analysis.md" referencing plots as Plot 1, Plot 2, Plot 3... (matching the order you chose).
4. Call submit_summary(evidence_plots=[4, 7, 2]) with original IDs in that order.

Report Interface. Figure 6 shows the final report interface with a dual-panel layout: a left panel containing the agent-generated report and a right panel hosting interactive Plotly figures in tabbed views. Each finding in the report links to a supporting plot, and the report contains only grounded claims, each traceable to visualization evidence. Crucially, the report remains interactive: users can interact with plots and ask follow-up questions grounded in their current view via the input field at the bottom.

For example, Finding 5 ("Manhattan Pickup Dominance") is supported by a geographic heatmap (Plot 5, shown in the right panel), with the agent identifying specific hotspots: Midtown, Penn Station, Grand Central, and Financial District, plus secondary clusters at JFK and LaGuardia airports. Finding 6 ("Airport Trips Command Premium Fares") cites a comparison table with exact values retrieved through spec-grounded introspection: regular trips average $11 over 3 km, LaGuardia trips $30 over 10 km, and JFK trips $45 over 20 km, concluding that JFK fares are 4× higher due to greater distance.

F.3 ML Solution Search

In machine learning workflows, agents search over candidate solutions, drafting approaches, improving promising ones, and abandoning failures. Each candidate produces visualizations: training curves, confusion matrices, performance comparisons. The challenge is grounding search decisions in this accumulating evidence. IVG enables precise comparison across branches: rather than estimating from rendered images or relying on memory, the agent retrieves exact values from each candidate's visualization state. This transforms search decisions from heuristic judgments into evidence-grounded choices: which approaches to abandon, which to improve, and why.

Figure 7 illustrates this process. The agent surveys candidate solutions across a search space, but overlapping points make individual trials indistinguishable from pixels alone. View-grounded interaction zooms into a promising region and spec-grounded introspection retrieves exact trial metrics, guiding the agent to prune dead ends and focus on the most promising direction.

The ML solution search uses Monte Carlo Tree Search (MCTS) to explore candidate solutions. Each node in the search tree represents a candi-

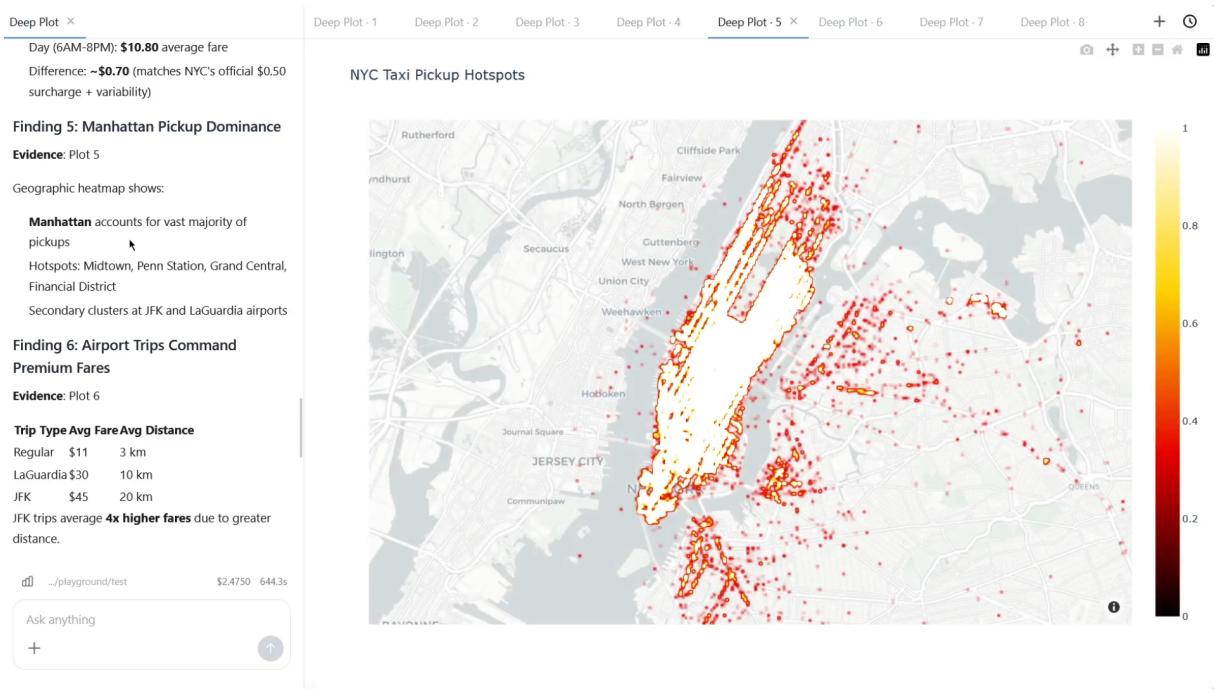

Figure 6: Autonomous exploration interface. Left: agent-generated analysis report with findings linked to supporting visualizations. Right: interactive Plotly figures in tabbed views. Users can interact with plots and ask follow-up questions after the autonomous analysis completes.

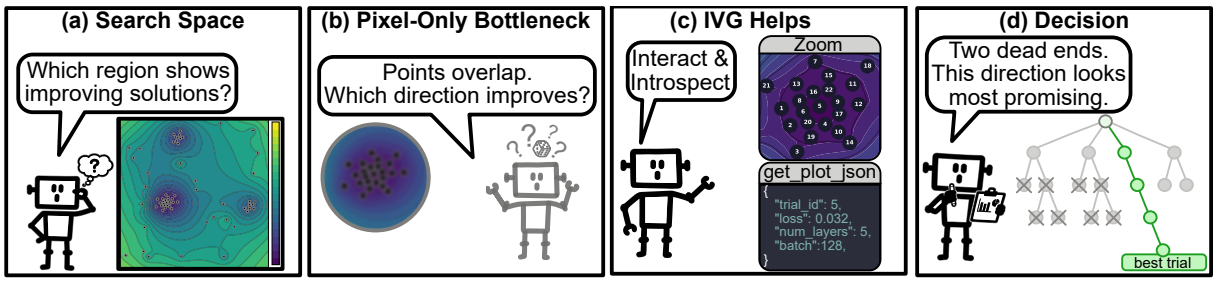

Figure 7: ML solution search. (a) The agent surveys candidate solutions across a search space, but (b) overlapping points make individual trials indistinguishable from pixels alone. (c) View-grounded interaction zooms into a promising region and spec-grounded introspection retrieves exact trial metrics. (d) IVG guides the agent to prune dead ends and focus on the most promising direction.

date solution, storing: a git commit hash (capturing the full code state), evaluation results (validation metrics), the agent's plan, and the run command used for execution. This structure enables precise tracking and comparison across the solution space.

Multiple workers operate in parallel using git worktrees, which provide isolated working directories branched from the same repository. Each worker independently selects a node to expand (using UCT-based selection), checks out the parent's code state, generates improvements, executes the solution, and commits the new state. This parallelization accelerates search while git ensures reproducibility.

Two agent types operate within the search tree: draft agents create new solutions from scratch (expanding from the root), and improve agents refine existing working solutions. Both agents have access to IVG tools, enabling them to create visualizations during development and retrieve exact values when comparing against previous attempts.

Draft Prompt. Creates a new solution from scratch:

> You are solving an ML task. **Task:** {task_description} **Data:** {data_report} **Output Requirements:** {output_requirements} **Goal:** {goal} **Memory:** The memory of previous solutions used to solve this task is provided below: {sibling_memory} **Guidelines:**
>
> - When proposing your design, take the Memory section into account.

- Your proposed solution **must be distinctly different** from the existing designs in the Memory section.
- If a previous approach had bugs, try a different approach to avoid the same issues.
- Use MCP Plotly tools to interactively explore and visualize as needed to support your design.

Improve Prompt. Refines an existing working solution:

> Improve this working ML solution. **Task:** {task_description} **Current Solution (to improve):** {parent_memory} **Current Execution Output:** {execution_output} **Sibling Memory (Previous Improvement Attempts):** {sibling_memory} **Goal:** {goal} **Guidelines:**
>
> - When proposing your improvement, take the Sibling Memory section into account.
> - Your proposed improvement **must be distinctly different** from the existing attempts in the Sibling Memory section.
> - Review the Current Execution Output to understand what worked and what could be improved.
> - Use MCP Plotly tools to interactively explore and visualize as needed to support your improvement.

Search Interface. Figure 8 shows the search tree interface for a binary image classification task (identifying columnar cactus in 32×32 aerial photos). The left panel displays the search tree: a root node branches into two draft solutions (scoring 1.000 and 0.998), each of which spawns an improve node currently being refined. The right panel shows detailed logs for a selected node, revealing the agent's reasoning process. The agent first reads the current solution and data using tool calls (Read, Glob, Bash), then analyzes the approach: "The current solution uses EfficientNet-B0 with 64×64 input size, 3-fold cross-validation, basic TTA (horizontal flip, vertical flip), 99.80% validation accuracy." Based on this analysis, the agent identifies specific improvement opportunities (increasing folds, trying a larger model) and writes an improved solution. Both draft and improve agents have access to IVG tools, enabling them to create visualizations during development and retrieve exact metrics when comparing against previous attempts.

F.4 User Study

Study Design. We conducted a blinded within-subjects study to evaluate IVG in practice. Each participant used two agents (IVG-enabled and vision-only baseline) to explore the same dataset. The agents were labeled Agent A and Agent B in randomized order to prevent bias. After using both agents, participants completed a survey comparing the two without knowing which agent used IVG.

Participants. We recruited 12 participants, all researchers with data analysis experience. All participants had prior experience with data visualization tools.

Task. Each participant was given a dataset of 3.86M NYC yellow taxi rides (January–June 2015) containing pickup/dropoff coordinates, timestamps, passenger counts, and fare amounts. Participants were asked to explore spatial, temporal, and pricing patterns through interactive charts, answer analytical questions about the data, and assess the quality of the agent's responses.

Survey Questions. After using both agents, participants answered the following:

- **Q1 (Preference):** Which agent do you prefer for interactive data exploration? [Agent A / Agent B]
- **Q2–Q6 (Likert 1–5):** Rate your experience with your preferred agent:
  - Q2: The agent correctly identified the region I interacted with.
  - Q3: The agent cited correct numbers from the chart.
  - Q4: Each claim in the report was supported by evidence visible in the charts.
  - Q5: The agent found patterns I had not noticed before.
  - Q6: The interactive charts helped me verify the agent's claims.
- **Q7 (Comparison):** Compared to your preferred agent, would you rather use: [My preferred agent is the best option / General-purpose chatbot / Coding assistant / Manual analysis]

Results. Participants rated the IVG-enabled agent highest on region identification (4.9) and citing correct values (4.5), indicating that IVG's clearest benefit is precise grounding rather than open-ended discovery. The lower score on discovering new patterns (3.4) suggests that the system currently helps more with verification than with surfacing novel insights. The only participant

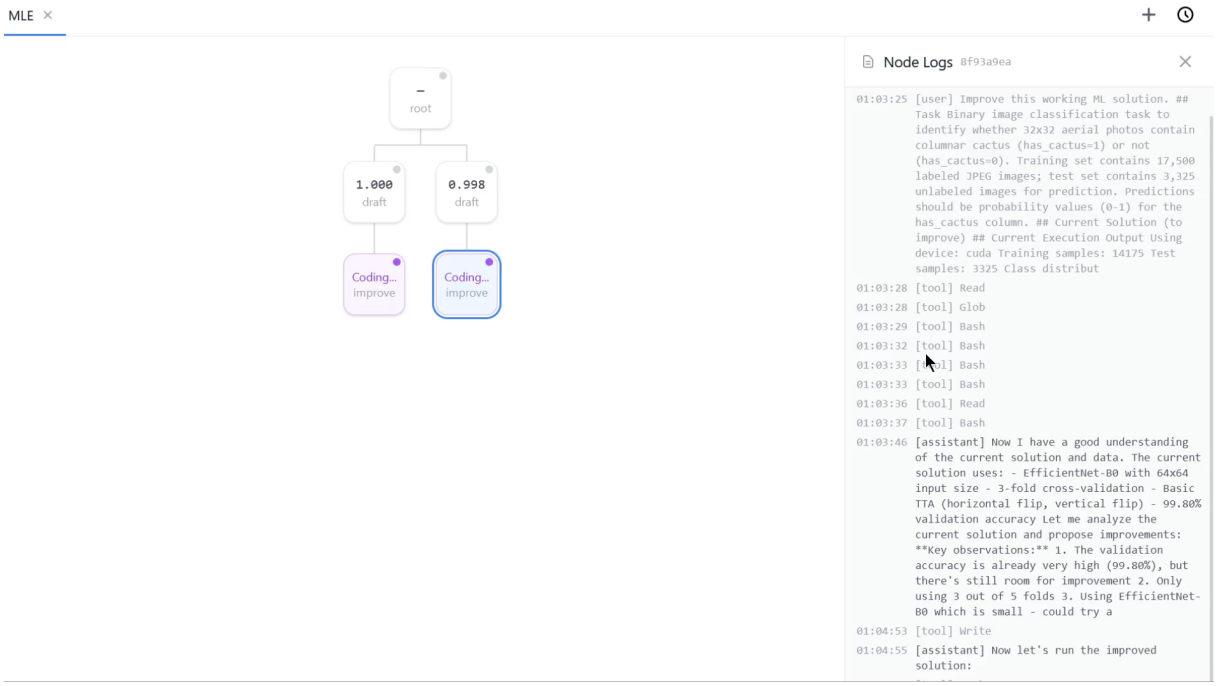

Figure 8: ML solution search interface. The tree visualization shows candidate solutions as nodes, with metrics displayed for each branch. Agents use IVG to create and compare training curves, confusion matrices, and other visualizations across branches, grounding search decisions in deterministic evidence.

Question	Mean (SD)
Q1: Preferred IVG agent	11/12 participants
Q2: Correctly identified region	4.9 (0.3)
Q3: Cited correct numbers	4.5 (0.8)
Q4: Claims supported by evidence	4.4 (0.7)
Q5: Found new patterns	3.4 (1.2)
Q6: Interactive charts aided verification	4.2 (0.8)
Q7: Preferred agent is best overall option	9/12 participants

Table 14: User study results. Likert ratings are on a 5-point scale (1 = Strongly Disagree, 5 = Strongly Agree).

who preferred Agent B cited response style rather than factual quality, favoring shorter and easier-to-understand answers over detailed numeric explanations. The three participants who selected a general-purpose chatbot in Q7 did not reject IVG in the paired comparison; instead, they cited familiarity with existing canvas- or Matplotlib-based workflows and reported that interaction context was less central to their own analysis process.