arXiv:2410.12268v2 [cs.HC] 3 Apr 2025

VisAnatomy: An SVG Chart Corpus with Fine-Grained Semantic Labels

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, Zhicheng Liu*

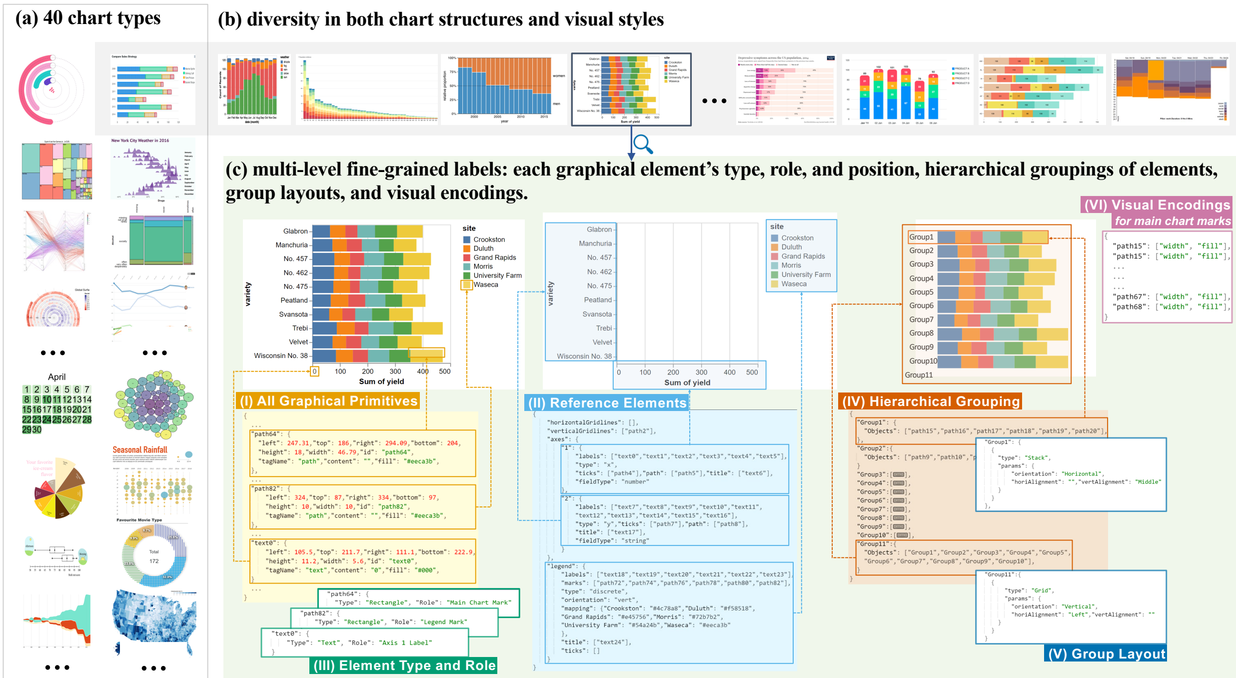


Fig. 1: VisAnatomy is an SVG chart corpus that (a) consists of 942 charts across 40 chart types, (b) promotes diversity within each chart type, featuring rich design variations in terms of visual structures and styles, and (c) distinguishes from existing chart corpora in its multi-granular chart semantic labels on more than 383K graphical elements, including each element's type, role, and position, hierarchical groupings of elements, group layouts, and visual encodings.

Abstract— Chart corpora, which comprise data visualizations and their semantic labels, are crucial for advancing visualization research. However, the labels in most existing corpora are high-level (e.g., chart types), hindering their utility for broader applications in the era of AI. In this paper, we contribute VisAnatomy, a chart corpus containing 942 real-world SVG charts produced by over 50 tools, encompassing 40 chart types and featuring structural and stylistic design variations. The underlying data tables are also included if available. Each chart is augmented with multi-level fine-grained labels on its semantic components, including each graphical element's type, role, and position, hierarchical groupings of elements, group layouts, and visual encodings. In total, VisAnatomy provides labels for more than 383k graphical elements. We demonstrate the richness of the semantic labels by comparing VisAnatomy with existing corpora. We illustrate its usefulness through four applications: shape recognition for SVG elements, chart semantic decomposition, chart type classification, and content navigation for accessibility. Finally, we discuss our plan to improve VisAnatomy and research opportunities VisAnatomy presents.

Index Terms—Chart, SVG, data visualization, corpus, dataset, multilevel fine-grained semantic labels

1 Introduction

*All authors are with the Department of Computer Science, University of Maryland, College Park, MD, USA.

†Emails: {cchen24, hbako, peihong}@umd.edu, {jhooker, jjoyal, scwang00, skim1270, jwu36, ading1, lsandeep, achen131, csinha}@terpmail.umd.edu, {leozcliu}@umd.edu

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

Visualization researchers have been curating chart corpora to advance the state of the art in chart creation and generation [27, 20], classification [37, 63], retrieval [45, 35], decomposition [13, 58], and editing [13, 19]. The availability of fine-grained semantic labels such as element properties and data encodings is vital for a range of visualization downstream tasks. For example, the shapes and roles of visual elements as well as their visual properties are required to *develop (semi-)automated data visualization reuse pipelines* [13, 63]; the correspondence between visual elements (or groups) and axis/legend labels is necessary for *developing chart reader experiences for visually impaired people* [75]; the grouping structure of visual elements can be utilized to develop graph neural network models [45].

However, the semantic labels in existing chart corpora are often in-sufficient and sometimes unreliable for supporting various visualization tasks [13, 14]. According to two recent surveys [14, 23], many corpora are not publicly available. The remaining ones typically have only high-level labels (e.g., chart types [7, 63] and chart area bounding boxes [26]). Although a few existing corpora do offer fine-grained labels, they often exhibit limited diversity in terms of the variety of chart-authoring tools and chart designs. This limited diversity hinders the generalizability of models built upon those corpora: they can easily fail when handling "out-of-distribution (OOD)" charts produced by other tools with inconsistent usage of SVG elements [45] and grouping structures [13]. In general, existing chart corpora are inadequate to support the development of robust visualization applications.

In this paper, we seek to address these limitations and contribute a diverse SVG chart corpus, VISANATOMY, with multi-level fine-grained semantic labels. VISANATOMY includes 942 real-world SVG charts and their corresponding multi-level semantic labels. The underlying data tables are also included if available (329 out of 942). The charts are collected through a manual process with careful inspection. For each chart, multiple independent expert annotators use a semi-automated annotation tool to obtain the semantic labels, and the quality of the labels is controlled through consensus among them.

VISANATOMY makes two key extensions over prior chart corpora. First, regarding corpus diversity, VISANATOMY encompasses 40 chart types (synthesized from three visualization typologies [12, 60, 30]) produced by over 50 tools from hundreds of public online sources, featuring structural and stylistic design variations (Section 2). Second, and more importantly, we identified a set of core components through a survey on existing visualization scene models [46, 61, 62, 65] to augment each chart in VISANATOMY with comprehensive semantic labels, including each visual element's shape (e.g., rectangle, pie, polyline), role (e.g., main chart mark, axis path, legend tick, annotation), and bounding box, the hierarchical grouping of elements, the layout for each group, and visual encodings (Section 3.1). In total, VISANATOMY provides labels for $383,459$ visual elements.

We evaluate the richness of semantic labels in VISANATOMY and the corpus diversity through a qualitative comparison between VISANATOMY and existing corpora (Section 3.4). The usefulness of VISANATOMY is illustrated through four applications: shape recognition for SVG elements, chart semantic decomposition, chart type classification, and content navigation for accessibility (Section 4). Finally, we discuss the current limitations of VISANATOMY and outline our future work including enlarging the size of VISANATOMY, obtaining labels of relational constraints, extracting data tables, and exploring visualization applications in greater depth and breadth (Section 5). The VISANATOMY corpus is available at https://VisAnatomy.github.io/.

2 VISANATOMY: CHART COLLECTION

In this section, we introduce the process to construct VISANATOMY. We first describe how the team decided on the desired chart types and collected charts following a standardized procedure. We then give an overview of the 942 charts in VISANATOMY, showing the distributions of the chart types, charting tools, and source domains.

2.1 Manual Chart Collection

Before the collection process started, we decided to focus on charts in the SVG (Scalable Vector Graphics) format. In recent years, SVG has emerged as a popular choice for curating corpora with fine-grained semantic labels in diverse visualization applications [13, 45, 51, 58, 65]. Compared to raster images, SVG directly includes low-level details such as element types and visual styles in its XML structure [14], removing the need for error-prone image segmentation [54] and element extraction [53]. Compared to code, where a label extraction approach is not easily generalizable to different visualization languages [52], SVG is supported as the output format by a wide array of languages and tools.

We manually searched for and sampled real-world SVG charts online to form the chart collection in VISANATOMY. There are other approaches to collecting charts, such as web crawling, that could lead to larger corpora. However, charts collected through automatic crawling cannot guarantee a balanced distribution of charts or diversity in terms of chart types and design variations [14]. Also, since SVGs can be embedded in web pages in various ways (e.g., inline SVG, object tag, iframes), consistent extraction of SVG charts in practice is more challenging compared to raster images, which are mostly directly embedded. We therefore decided to follow a manual collection process to ensure the quality and diversity of the SVG charts.

The first step in our chart collection process was to compile a set of targeted chart types by browsing three visualization typologies: the Chartmaker Directory [12], the Data Viz Project [30], and the Data Visualisation Catalogue [60], each of which contains a detailed categorization of chart types. We cross-checked the named chart types in these three collections and focused on visualizations that consist of basic geometric shapes including `rectangle`, `circle`, `ellipse`, `pie`, `arc`, `line`, `polyline`, `area`, `polygon`, `geo-polygon`, and `text`. For the underlying data in the visualizations, Munzner [55] defined four types of datasets: tables (items & attributes), trees and networks (nodes & links), fields, and geometry. We decided to focus on visualizations of tables, thus excluding node-link diagrams and scientific visualizations. Applying these criteria, we narrowed down to 31, 37, and 37 chart types from the three typologies, respectively. Finally, we unified the names of these chart types and achieved a final set of 40 chart types (Figure 2).

We then started collecting SVG charts for each chart type by (1) browsing online charting tool galleries (e.g., D3.js [10], Vega-Lite [62], Mascot.js [47]), (2) browsing online communities where visualizations are shared by chart makers (e.g., Observable [57], bl.ocks.org [9], Spotfire [66]), and (3) searching for certain chart types using engines such as Google Advanced Image Search [1] (with "SVG" specified as the target file type) and Bing Visual Search [3]. It is important to note that when collecting charts, *we prioritized diversity over quantity because promoting rich design variations and supporting broader visualization applications are our research goals*. Therefore, for each chart type, we focused on including designs from different visualization galleries and websites, showcasing diverse visual styles. We aimed to include at least 20 designs for each chart type. However, we kept adding new designs as long as they introduced unique visual styles that had not yet been observed and did not distort the overall distribution of chart types (i.e., an approximately uniform distribution). During this process, we also examined the details of SVG files to discard invalid ones, e.g., those using `<image>` elements with hyperlinks to render the whole chart. For each SVG chart collected, we obtained its image in the PNG format with a Python script.

The project team consisting of six authors held weekly meetings to inspect the collected SVG charts, removed unqualified charts (e.g., repeated or highly similar charts), and enforced consistent selection criteria. Charts that do not fall into one of the 40 types but still embody interesting design ideas were moved to the "Others" category where we store bespoke chart designs. This process was repeated until we had the expected number of valid designs (i.e., 20) for each chart type. This iterative process ensures that the whole team has at least one pass on every collected chart and that issues with collected charts are resolved consistently. After a 28-week chart collection effort, we reach the final set of charts in both the vector and raster image formats in VISANATOMY, containing a total number of 942 charts with an approximately even distribution across the 40 chart types plus an "Others" category, encompassing more than 50 charting tools and hundreds of web domains (chart sources).

2.2 VISANATOMY Promotes Chart Diversity

Chart Types. In its current state, VISANATOMY contains 942 real-world SVG charts. Each chart has its corresponding label file in the JSON format, and is also available in the PNG (Portable Network Graphics) format. Figure 2 shows the overall distribution of the chart types categorized by the mark types. Within VISANATOMY, there are 40 named chart types together with an "Others" category containing custom designs such as composite or superimposed charts that do not fall appropriately into a specific chart type. Line graph, area chart, bar chart, grouped bar chart, and the "Others" category constitute higher proportions in the corpus, and the number of charts in each remaining type ranges between 20 to 26. This distribution makes VisAnatomy a balanced corpus with a wide variety of chart types.

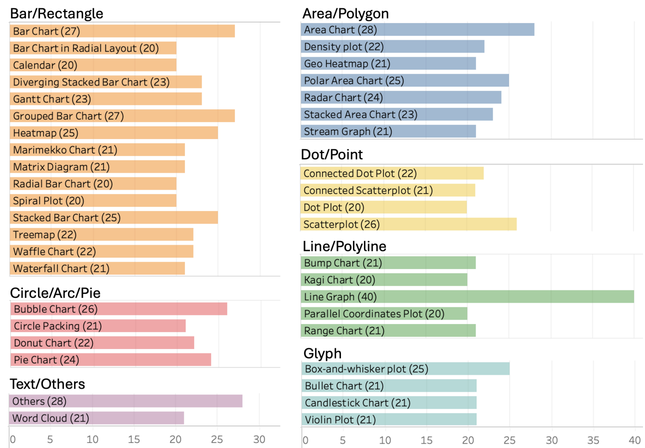


Fig. 2: Chart type distribution in VisAnatomy categorized by primary mark types. Within each category the chart types are sorted alphabetically.

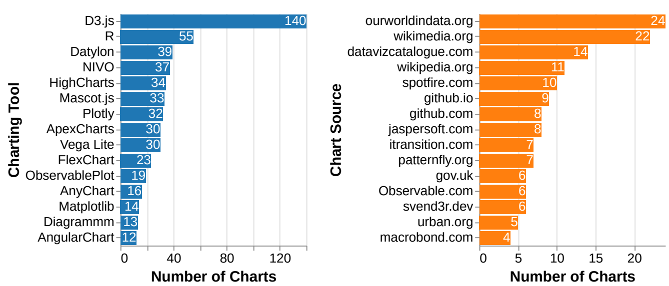


Fig. 3: Distributions of chart types, charting tools (top-15), and source domains (top-15) in VisAnatomy.

Charting Tools and Chart Sources. For each chart, we record information on the charting tool (if it is explicitly revealed in the source website) and/or the public website domain (if it is not coming from a charting tool's gallery). In total, VisAnatomy collects charts created using 58 charting tools and more than 100 different online domains. In Figure 3 we show the top 15 charting tools and chart sources. D3.js [10] contributes the largest portion as it is the most expressive tool for creating interactive web-based SVG visualizations [7]. Several other visualization grammars and tools also provide a decent amount of charts to VisAnatomy, including R [72], Datylon [22], NIVO [56], Highcharts [34], Mascot.js [47], Plotly [2], Apexcharts.js [5], and Vega-Lite [62]. The number of charts is more evenly distributed across chart sources, as most domains only occur fewer than five times.

Design Variations. In VisAnatomy, we also strive for chart design diversity within each chart type, featuring rich design variations in terms of both chart structures and visual styles. Figure 1(b) shows nine exemplary design variations for the stacked bar chart type. In VisAnatomy, design variations include but are not limited to:

- different mark types used to create charts of the same type (e.g., bump charts composed of area marks or polylines with dots);
- different orientations of marks (e.g., connected dot plots containing horizontal or vertical glyphs);
- layering of marks (e.g., superimposed area charts);
- nested structures (e.g., small-multiple waffle charts and grouped box and whisker plots);
- different positions, orientations, and visual styles of reference elements such as axes, legends, and gridlines;
- different styles of annotations and embellishments.

We include detailed information of the chart types, tools, and sources in VisAnatomy in the supplementary materials[1].

3 VisAnatomy: Multilevel Fine-grained Semantic Labels

We took an iterative in-house labeling approach to obtain high-quality semantic labels for a set of core components synthesized from literature [46, 61, 62, 65]: mark elements reference elements, hierarchical grouping, visual encodings, and group layout. We decided not to obtain labels through crowdsourcing to ensure the label quality in VisAnatomy. Crowdsourced labels often require significant time to inspect and correct [42, 39, 74]. Moreover, our desired labels require visualization expertise from the annotators, which is hard to guarantee through crowdsourcing platforms. Three experts in the team, each with at least four years of visualization research experience, participated in labeling the semantic components of the collected SVG charts. In this section, we describe the semantic labels that are associated with the charts in VisAnatomy and the labeling process.

3.1 Fine-grained Labels of Multilevel Scene Components

To support a broad set of visualization applications, we need detailed semantic labels beyond just chart types [14]. Specifically, we need a comprehensive understanding of the structure of a chart at multiple levels of granularity, from its global-level chart type down to the properties of individual elements. To this end, we surveyed related literature to compare existing visualization abstractions [46, 61, 62, 65]. We finally decided to focus on the labels for the following components commonly shared across visualization grammars and abstractions: mark elements, reference elements, hierarchical grouping, visual encodings, and group layout. The supplemental materials contain detailed descriptions of the components we have surveyed.

Using the stacked bar chart presented in Figure 1(c) as an example, we show its scene structure (gray nodes and edges) together with the correspondence to the semantic labels recorded in VisAnatomy in Figure 4. Specifically, Reference Elements specify properties and involved elements for `gridlines`, `axes`, and `legend`; All Graphic Primitives and Element Type and Role contain detailed information about all graphical elements in the `scene`, including `rectangle` marks and Reference Elements; Hierarchical Grouping and Group Layout correspond to the spatial clusters and relationships along the rightmost branch (the `collection` subtree) of the scene graph, and Visual Encodings record encoded channels for `rectangle` marks. Note that in other types of visualizations, such correspondences remain similar. We next give a detailed explanation of the six semantic labels by walking through the example stacked bar chart (in Figure 1(c)).

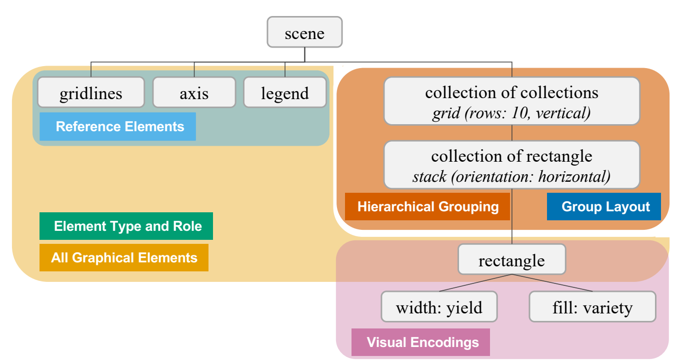


Fig. 4: The components in the stacked bar chart from Figure 1(c) and its correspondences to the labels in VisAnatomy.

[1] see detailed information of VisAnatomy's chart collection.

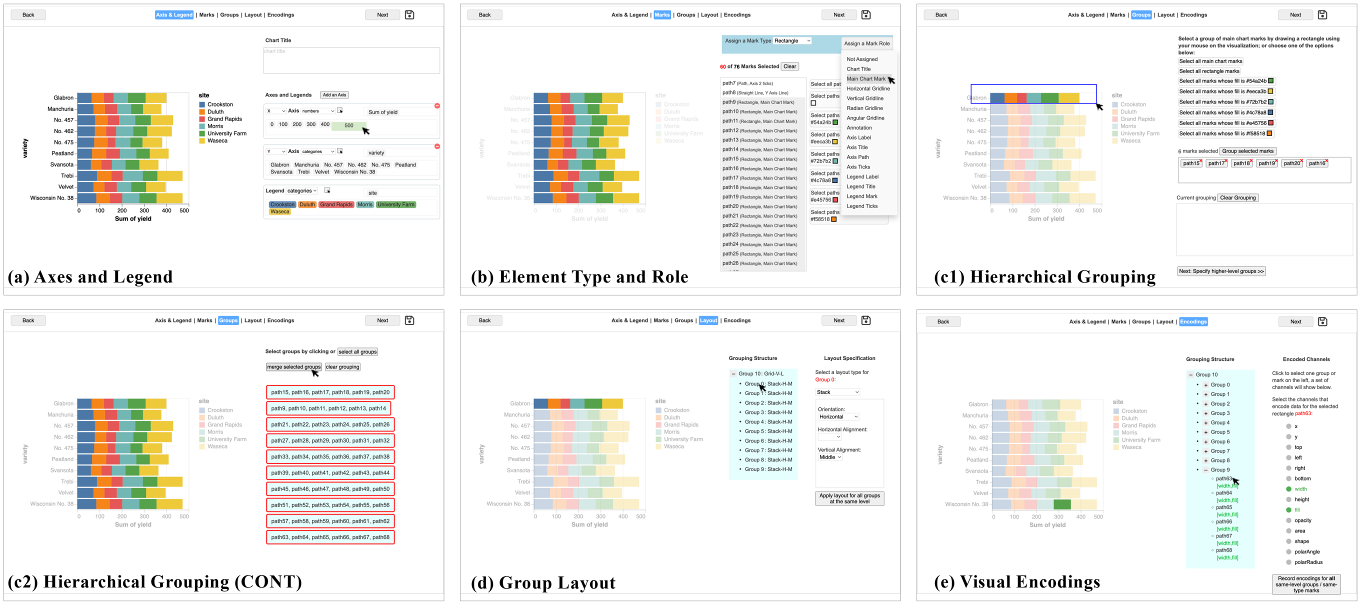


Fig. 5: The labeling tool we have developed to produce the semantics labels in VISANATOMY divides the labeling process into five stages: **Axis&Legend** (a), **Marks** (b), **Groups** (c1, c2), **Layout** (d), and **Encodings** (e) and provides necessary interactions to accelerate the labeling process.

All Graphic Primitives include every geometric shape and text element in an SVG chart (leaf nodes in the SVG hierarchy). Each element has several general properties, including element ID, element tag name, text content (if any), and fill color. Each element's bounding box is expressed in absolute coordinates. Figure 1(c-I) presents three example primitives and their properties in the stacked bar chart.

Reference Elements are titles, axes, legends, and gridlines in a chart. Titles and gridlines are specified with the IDs of the corresponding SVG elements in the label file. Axes and legends contain information about the type (e.g., x, y, angle, radius) and orientation (e.g., horizontal, vertical), and they are further broken down into lower-level components such as labels and ticks that are also specified with IDs of their corresponding elements. Figure 1(c-II) shows the semantic labels for the x-axis, y-axis, and the color legend in the chart.

Element Type and Role record the shape type and semantic role of each SVG element. In SVG files, the tag name of an element does not always match the geometric shape. VISANATOMY resolves this ambiguity through the Element Type label. For example, in Figure 1(c-III), "`path64`", the bar representing "Winsconsin No. 38" in Waseca, has the tag name "path", while its Element Type is labeled as `Rectangle`. In addition, the same type of elements can play different roles in a chart. For example, elements "path64" and "path82" are both rectangles in Figure 1(c-III), but the former is labeled with the Element Role `Main Chart Mark` and the latter is labeled as a `Legend Mark`.

Hierarchical Grouping reflects the multi-level semantic clustering of main chart marks (elements whose Element Role is `Main Chart Mark`, i.e., the 60 colored bars from the main chart area). The example stacked bar chart has 10 mark groups ("`Group1`" to "`Group10`"), each corresponding to one of the 10 varieties of barley (i.e., "Glabron" to "Winsconsin No. 38" along the y-axis), and they further form a higher-level group, "`Group11`", that encapsulates all the 10 lower-level groups. This hierarchy is recorded in the label file as shown in Figure 1(c-IV).

Group Layout indicates the spatial relationship between visual objects within one group (e.g., grid, stack, packing, radial) with orientation (e.g., horizontal, vertical, angular) and alignment parameters (e.g., bottom-aligned, left-aligned). This information is present for all the groups at all levels. For example, in Figure 1(c-V), "`Group1`" is labeled with a middle-aligned horizontal stack layout, and the same layout applies across "`Group2`" to "`Group10`"; the higher-level "`Group11`" is labeled as a left-aligned vertical grid layout.

Visual Encodings records the mapping of visual channels to data for main chart marks and mark groups. Figure 1(c-VI) shows labels for visual encodings in the stacked bar chart, indicating that the "`width`" and "`fill`" of the bars are representing values from the underlying data table. In some charts, the visual channels of a group can encode data as well. For example, in a small-multiple design of grouped bar charts, the position of each bar group can represent the approximate geographic location of the corresponding U.S. state. VISANATOMY thus organizes visual encoding labels by each visual object's ID.

To summarize, VISANATOMY provides rich fine-grained semantic labels of each chart's components, including each graphical element's type (e.g., rectangle, pie, polyline), role (e.g., mark, axis path, legend tick, annotation), properties (e.g., ID, fill color, content) and bounding box, the hierarchical grouping of graphical elements, the layout for each element group, and visual encodings on main chart elements.

3.2 Labeling with A Semi-Automated Tool

These semantic labels are created using a semi-automated chart labeling tool we have developed. According to Chen and Liu [14], obtaining high-quality chart labels is expensive and time-consuming, especially for complex labels that require careful examination of charts. Moreover, SVG charts have diverse hierarchical structures and utilize SVG elements in various ways, especially when they come from different tools and sources. To address these challenges, we have developed a mixed-initiative multi-stage labeling tool to facilitate the process and mitigate laborious inspection.

The annotation tool divides the labeling process into five stages: **Axis&Legend**, **Marks**, **Groups**, **Layout**, and **Encodings**. Figure 5 shows the system user interface and an example labeling workflow using the stacked bar chart (in Figure 1(c)). Once the SVG chart is loaded, the system will first pre-process the SVG file through its hierarchy to obtain All Graphic Primitives. Then the system leverages the heuristics-based methods reported in the Mystique system [13] to detect the chart title, axes, and legend, and displays the results in the **Axis&Legend** UI accordingly (Figure 5(a)). The annotator can correct the results through drag-and-drop or lasso selection over SVG texts to revise Reference Elements; for example, in Figure 5(a), the annotator is dragging the text "500" from the chart into the x-axis label box. The annotator can also add more axes if more than two axes exist (e.g., in parallel coordinates).

Once the annotator finishes inspecting the results in this stage, they can click the "Next" button to go to the **Marks** stage, where the full list of graphical elements is displayed (Figure 5(b)). The annotator can click to select a single mark, batch-select multiple marks through the generalized selection options [32] (e.g., selecting the same-type or same-color marks), or shift-click to select consecutive marks in the mark list. When a set of marks is selected, they will be highlighted in full opacity in the chart, while all other elements will be partially transparent. The annotator can label the Element Type and Role of the selected marks through the corresponding drop-down menus. In Figure 5(b), the selected paths are assigned Element Type `Rectangle` and Element Role `Main Chart Mark`. At this stage, the annotator would also need to specify gridlines and low-level components for axes and legend (e.g., ticks, paths), so that the Reference Elements label is complete.

In the next stage, **Groups**, the annotator specifies Hierarchical Grouping on all the main chart marks. To select marks for grouping, the annotator can click to select individual elements, make a lasso selection (Figure 5(c1)), or choose from the generalized selection options. The selected marks can then be grouped by clicking the "Group selected marks" button. After that, the system will recommend a list of inferred "other mark groups", and the annotator can either agree to finish the lowest-level grouping or disagree to continue grouping manually. Once all lowest-level mark groups are specified, the annotator clicks the "Specify higher-level groups" button to go to the second phase of the **Groups** UI (Figure 5(c2)), where they work on higher-level grouping, e.g., in Figure 5(c2) the annotator has selected all 10 mark groups and are merging them into one final group.

After Hierarchical Grouping is finished, the annotator goes to the **Layout** stage. Here the hierarchical groups will be displayed as a collapsible nested list with clickable items. The annotator can click on an individual group and label its layout type and parameters, as shown in Figure 5(d). In the final **Encodings** stage, the system adds mark items to the nested list and lets the annotator specify their encoded visual channels (Figure 5(e)). The visual channel list will be updated accordingly based on Element Type of the selected element. In both the **Layout** and **Encodings** stages, the annotator can apply the label of one group or mark to its peers, i.e., same-level and same-type objects, to accelerate the process of assigning Group Layout and Visual Encodings.

At any time during the labeling process, the annotator can click on the "save" button in the upper-right corner of the interface to save a local copy of all semantic labels created so far in the JSON format. When the same chart is loaded next time, the system will automatically retrieve the corresponding label file (if it exists), and load the semantic labels in the UI accordingly to allow the annotator to inspect existing labels and continue unfinished work. The semi-automated labeling system and a document recording the options for Element Type, Element Role, types and parameters of Group Layout, and channels of Visual Encodings, are included in the supplementary materials[2].

3.3 Iterative Annotation and Quality Control

Using the system to produce high-quality semantic labels requires sufficient knowledge and expertise in the visualization field. Thus, we adopt an iterative approach to obtain the labels from three experts who are familiar with visualization abstraction papers in the team. The first author, a Ph.D. student who has published peer-reviewed papers in visualization-related conferences (e.g., VIS, EuroVis), performed the first-round labeling. The time required to finish labeling one chart ranges between 10 to 20 minutes. After that, the second author, a Ph.D. student who has published at visualization-related conferences such as VIS and IUI, and the last author, a researcher who has been contributing to the visualization community for more than 15 years, performed the second-round inspection. Each chart has been reviewed by at least two experts on the team. Whenever disagreements over certain semantic labels arose, the three authors discussed the cases to reach a consensus on the correct labeling approach, and all the charts with features related to the issues would be re-labeled. These discussions not only improved the consistency and accuracy of the labels but also enhanced the labeling system.

3.4 Comparing VisAnatomy with Related Corpora

In this section, we compare VisAnatomy with nine existing chart corpora: Beagle [7], YOLoT++ [28], REV [58], MASSIVE [8], MVV [17], VisImages [26], Chart-LLM [43], VisText [68], and VisEval [16] (results are shown in Table 1). These corpora were selected based on the following criteria: the corpus should be publicly available, contain chart images, and have been published at VIS or HCI venues. Based on the criteria, we didn't include the Visually29k corpus [48] as it focuses on infographics and the VIS30K corpus [15] which contains images of data tables.

The comparison shows that VisAnatomy distinguishes from other chart corpora in two major aspects: the rich, multi-granular chart semantic labels, and its diversity regarding chart types, designs, and sources. In terms of the number of charts, VisAnatomy is comparable to datasets that specifically emphasize label quality, such as

[2] see the code and details for the labeling system.

Table 1: A comparison between VisAnatomy and nine related chart corpora in their current states. ✓ indicates the existence of a certain property in the corresponding corpus, ◐ indicates partial existence, ○ means a property is absent in the current state but can be obtained with some effort, and - indicates that a property is unavailable.

		VisAnatomy	MVV [17]	VisEval [16]	YOLaT++ [28]	REV [58]	VisText[68]	Beagle [7]	MASSIVE [8]	VisImages [26]	Chart-LLM [43]
Primary Collection Method		Manual Curation	Manual Curation	Transforming An Existing Corpus	Computer-Aided Generation	Computer-Aided Generation	Computer-Aided Generation	Web Crawling	Web Crawling	Web Crawling	Web Crawling
# Charts		946	360	1,150	15,197	5,125	12,441	~**41,000**	2,070	12,267	1,981
# Types		**40**	14	7	11	4	3	24	12	34	10
# Tools		**58**	-	1	2	-	1	5	-	-	1
Format	SVG	✓	-	-	✓	◐	○	✓	-	-	○
	Bitmap	✓	✓	-	○	✓	✓	✓	✓	✓	○
	Program	-	-	✓	✓	-	✓	-	-	-	✓
Label	Chart Type	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Chart BBox	✓	✓	-	-	-	-	-	-	✓	-
	Element BBox	✓	-	-	✓	◐	◐	-	-	-	-
	Legend Elements	✓	-	-	✓	◐	-	-	-	-	-
	Axis Elements	✓	-	-	✓	◐	✓	-	-	-	-
	Element Shape	✓	-	-	✓	-	◐	-	-	-	-
	Element Role	✓	-	-	✓	◐	◐	-	-	-	-
	Mark Grouping	✓	-	-	-	-	-	-	-	-	-
	Group Layout	✓	-	-	-	-	-	-	-	-	-
	Visual Encoding	✓	-	-	-	-	-	-	-	-	○
	(NL, VIS) pairs	-	-	✓	-	-	✓	-	-	-	✓

VisEval [16] and Chart-LLM [43], which typically contain thousands of samples. While corpora such as Beagle [7] collect many more charts through web crawling, they tend to include duplicated designs with imbalanced chart distributions [13]. Moreover, they lack fine-grained component-level labels. On the other hand, corpora created through computer-aided generation (e.g., YOLaT++[28]) do provide fine-grained component labels, but they are highly restricted in terms of chart and tool diversity. VISANATOMY balances between promoting diversity for real-world charts and maintaining quality control for fine-grained labels. The labels for over 380K visual elements in VISANATOMY adequately support applications requiring extensive component-level annotations.

4 USE CASES

In this section, we demonstrate four use cases of VISANATOMY, each focusing on a specific downstream visualization application:

- *Recognizing shapes of SVG path elements* with Large Language Models (LLMs): we evaluate three open-srouce LLMs to examine their capabilities in shape recognition with text inputs.
- *Decomposing* rectangle-based *SVG charts for layout reuse*: we use VISANATOMY to obtain a validation set to evaluate the performance of an existing system Mystique [13].
- *Classifying chart types* utilizing Graph Neural Networks (GNNs) [73]: we present a new approach called GNN4SVG and report our preliminary findings on the performance of GNNs using VISANATOMY as a benchmark corpus.
- *Supporting accessible navigation* of chart content for visually impaired people: we demonstrate how VISANATOMY can enable the replication of the rich screen reader experiences through keyboard navigation as described in the study by Zong et al. [75].

The four use cases spans different requirements on the semantic labels, from low-level mark details (e.g., attributes of SVG elements) to high-level chart information (e.g., chart type). Through these applications, we showcase the utility of VISANATOMY. We demonstrate its role in introducing new paradigms for AI in visualizations. In addition, we illustrate how the semantic labels in VISANATOMY ease the constraints on input formats such as the necessity for charts to be created with specific tools, supporting a wider range of input charts. The supplementary materials[3] include our implementations, detailed results, and demo videos for the use cases.

4.1 Shape Recognition with LLMs

Mark shape recognition is a classic task in automated visualization understanding. For instance, REV [63] utilizes vision models to detect the mark type used in a bitmap visualization. More recently, researchers have found that (1) text-based chart specifications improve the performance of language models (LMs) on chart-reading tasks compared to vision-based approaches [11], and (2) the existence of semantic shape information for SVG *path* elements ("Primal Visual Description (PVD)") boost the performance of LMs for vector graphics reasoning [71]. Therefore, shape recognition plays an important role in AI-based visualization comprehension and reasoning. In this use case, we generate a large-scale dataset of SVG *path* elements with ground-truth mark types based on VISANATOMY, and evaluate the shape recognition capability of three open-source LLMs: Llama-3.1-8B-Instruct [29], Mistral-7B-Instruct-v0.3 [4], and DeepSeek-V2.5 [25].

Dataset of SVG *path* elements. We first select all the SVG elements with the tag "path" and their corresponding shape labels (i.e., Element Type), forming an initial dataset of 175,065 path elements. However, this dataset is highly unbalanced because *rect* marks account for more than 100k of them. We therefore randomly sampled up to 1,000 elements per shape type (or all available elements if fewer than 1,000 existed) to form the final dataset which includes 9,914 *path* elements. This dataset encompasses ten shape types: Straight Line, Polyline, Rectangle, Polygon, geoPolygon, Circle, Pie, Arc, Area, and Text. Among these, Circle is the only shape type with fewer than 1,000 elements (914).

Prompt. The prompt contains the role specification ("*expert SVG shape classifier*"), the task ("*analyze the commands and coordinates in the 'd' attribute to classify the shape*"), explanations on the ten shape types and two examples per shape type (examples are *path* elements in VISANATOMY that were not sampled, except the two for Circle), and example input and output pairs at the end. The 'd' attribute of each *path* element in the dataset is appended to the prompt, and together they are fed into the LLMs. The supplementary materials contain the detailed prompt used.

Experimental Environments. We conducted inference using different hardware configurations based on model size. For Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3, inference was performed locally on a single NVIDIA RTX A5000 GPU. For DeepSeek-V2.5, due to its significantly larger model size, we utilized its cloud API for inference.

Results. Figures 6, 7, 8 show the per-class shape recognition accuracy of the three LLMs, where the rows represent the ground-truth shapes and the columns represent predicted shapes plus an "wrong answer" class. We observe significant variations in the three LLMs' capabilities. DeepSeek-V2.5 demonstrates superior performance with consistently higher accuracy across most shape categories compared to the other two models. The top-performing categories are Text (99.3%), Polyline (96.5%), Circle (88.7%), and Polygon (80.6%). In contrast, Llama-3.1-8B-Instruct shows moderate performance with notable strengths in specific categories like Circle (75.4%) and Text (72.0%); Mistral-7B-Instruct-v0.3 exhibits the lowest overall perfor-

[3] see our implementations, results, and demo videos for the use cases.

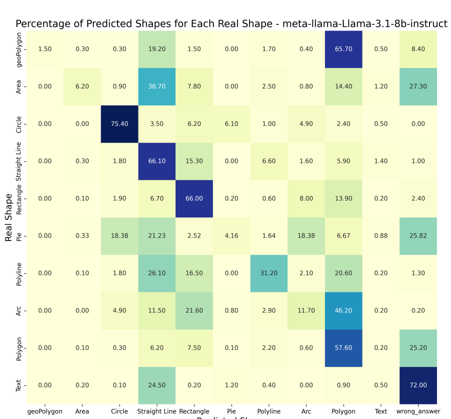


Fig. 6: Per-shape Accuracy of LLaMa-3.1.

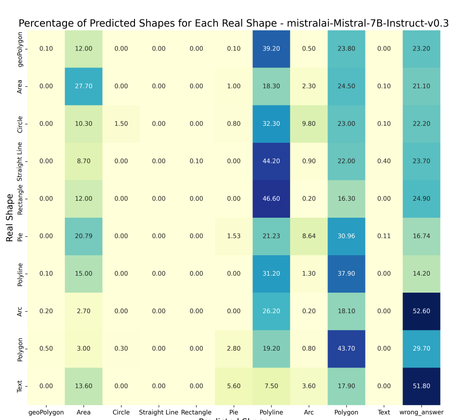


Fig. 7: Per-shape Accuracy of Mistral-7B.

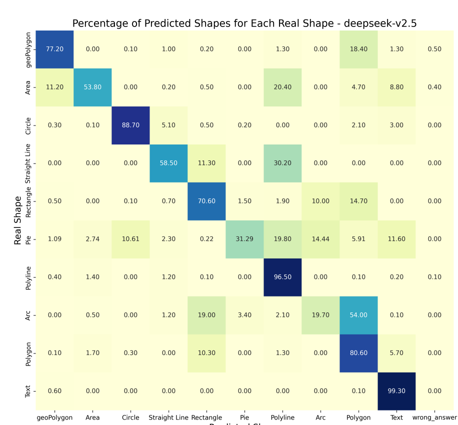


Fig. 8: Per-shape Accuracy of DeepSeek-V2.5.

mance with higher misclassification rates.

Each model exhibits distinct error patterns in their recognition behavior. Mistral-7B-Instruct-v0.3 frequently produces wrong answers, and tends to assign higher probabilities to Area, Polyline, and Polygon. While Llama-3.1-8B-Instruct outputs much fewer wrong answers, it shows confusion between geometrically similar shapes like Polygon and geoPolygon. Although DeepSeek-V2.5 demonstrates more robust performance overall, it still struggles with certain shape distinctions, particularly for Arc and Pie. The strong correlation between model size and performance in this shape recognition task indicates that larger models possess better capabilities in capturing and distinguishing subtle geometric features from textual descriptions.

The performance comparison highlights a critical trade-off in LLM deployment for visualization applications. While DeepSeek-V2.5 achieves superior accuracy, its size may require cloud deployment with substantial costs. Medium-sized models like Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3, though locally deployable on a single GPU, exhibit error-prone behaviors. Given this dilemma, fine-tuning medium-sized models on specialized datasets emerges as a promising direction, potentially offering a balance between computational efficiency and model accuracy for visualization applications.

4.2 Chart Layout Deconstruction

Unlike previous corpora that lack semantic information on low-level components, VisAnatomy specifies the mark grouping, layout, and data-channel encoding information, which play a vital role in the task of chart decomposition and reuse. Most approaches that focus on this task, such as D3 Deconstructor [31] and ChartReuse [19], require the input charts to be created using a specific tool. A more recent work, Mystique [13], designed a tool-agnostic decomposition and reuse pipeline for general SVG charts composed of rectangle-shape marks based on a diverse 150-chart corpus. What lies at the core of Mystique is a bottom-up hierarchical clustering algorithm that determines nested mark groups and their internal spacial relationships presented in a chart. In this section, we use VisAnatomy to form a validation set to evaluate the generalizability of the hierarchical clustering algorithm from Mystique on unseen charts.

To prepare the validation set, we first filter the charts based on the Element Type labels, and only keep those having the `Rectangle` type for all `main chart marks`, resulting in 306 charts. Given that Mystique does not consider radial and spiral layouts, we exclude spiral plots and bar charts in the radial layout. We further remove charts that were already included in Mystique's corpus. The final validation set consists of 248 charts encompassing 15 chart types.

We then run Mystique's hierarchical clustering algorithm using the `main chart marks` of each chart in this validation set as input, and the results are compared with Hierarchical Grouping and Group Layout labels in VisAnatomy. For the error cases, we perform another round of manual inspection to avoid false negatives as some charts have multiple reasonable grouping structures (e.g., a diverging bar chart) [13].

Results. 217 charts (out of 248) within the validation set are decomposed into their correct grouping structures and corresponding spatial relationships, making an approximate 87.5% accuracy (8% lower compared to the test accuracy reported in Mystique [13]). Examining the 31 error cases, we report three kinds of failure cases that were not discovered by the authors of Mystique [13]:

1. A slice-and-dice treemap (Figure 9(a)), whose overall packing layout was wrongly recognized as a vertical stack layout by Mystique's decomposition algorithm, leading to incorrect mark groups;
2. Glyph-based charts where the marks within a glyph do not always overlap, e.g., in Figure 9(b), the three lighter gray bars are stacked within each row without overlapping. The decomposition algorithm uses overlapping relationships to detect glyphs and fails in such cases;
3. A bespoke Gantt-calendar chart (Figure 9(c)) where the bar groups are positioned based on data and the bars within each group follow a horizontal stack layout. The algorithm fails to decompose this chart correctly, giving a lowest-level packing layout instead.

In summary, the generalizability of Mystique's hierarchical clustering algorithm is satisfactory, which corroborates the authors' claim on

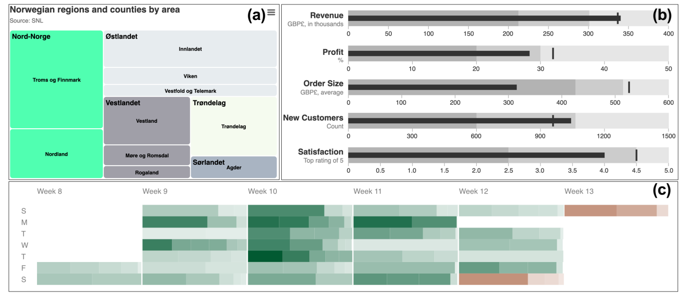


Fig. 9: Three new kinds of failure cases have been observed when evaluating the hierarchical clustering algorithm in Mystique [13] on a validation chart set generated from VisAnatomy: (a) a treemap visualization, (b) a bullet chart, and (c) a bespoke Gantt-calendar chart.

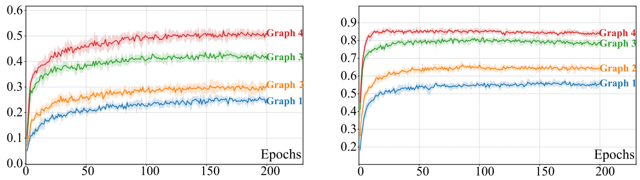
the diversity of their corpus. Nevertheless, the new error cases and Mystique's lack of support for non-rectangular marks and additional layouts call for more intelligent decomposition and mixed-initiative reuse approaches. The supplementary materials include details on the validation set and the performance statistics.

4.3 Chart Type Classification

Chart type classification is a typical visualization task, serving as the first step in many end-to-end systems such as Revision [63], ChartSense [37], and REV [58]. Various models, including Support Vector Machines (SVMs) [18] and Convolutional Neural Networks (CNNs) [44], have been explored. However, existing work mostly assumes bitmap images as inputs, and little is known about the implications of the SVG format on chart classification tasks. In text-based XML representations, SVG charts cannot be directly processed by vision models. Considering that SVG is a highly structured data format, Graph Neural Networks [64, 73] could be a better fit [14]. In a recent work focusing on chart retrieval [45], GNN is employed to obtain semantic embeddings to calculate pair-wise similarities between charts and is shown to be a more effective approach compared to image-feature-only models. To our knowledge, no previous research has conducted SVG chart-type classification with GNNs. In this case study, we introduce how we prepare the graph data based on the SVG charts and present our preliminary results on employing GNNs to recognize chart types on vector graphics charts. The main goal of this case study is to understand the influence of different types of semantic labels from VisAnatomy on the classification results.

Node Feature Extraction and Inclusion Criteria. If we focus on the raw SVG representations, we can designate basic shape elements such as `<line>` and `<circle>` as nodes in the graph. These elements correspond to All Graphic Primitives in VisAnatomy, and contain `main chart marks`, Reference Elements, and background noise (e.g., background rectangles, offscreen tooltips). To extract node-level features, we consider the following common features shared by different SVG element types for simplicity: (1) node-type, which is a one-hot encoding over the SVG element types, (2) node-position, which is a four-dimensional vector indicating the node's bounding box in `top`, `right`, `bottom`, and `left` coordinates, and (3) node-style, a three-dimensional binary vector indicating the existence of the `fill`, `stroke`, and `stroke-width` style properties in the SVG file. We can scale the node-position feature using the width and height of the source SVG chart so that all values are within the range [0, 1]. In addition to shape elements, we also designate SVG container elements like `<g>` and `<SVG>` as graph nodes, which only have the node-type feature with other feature dimensions filled with 0.

With the semantic labels in VisAnatomy, we can also restrict graph nodes to elements with `Main Chart Mark` as their Element Role, so we only focus on the chart content and remove potential noise. In addition, the node-type feature from the raw SVG file can be inaccurate, as it is normal for SVGs to represent different shapes (rectangles, circles, etc.) with `<path>` elements [14]. Thus, a more accurate ver-



(a) Top-1 test accuracy (b) Top-5 test accuracy

Fig. 10: Learning curves from the four models (introduced in Section 4.3) regarding both the top-1 (a) and top-5 (b) test accuracy.

sion of the node-type feature can be the one-hot encoding over the ground-truth mark types from Element Type.

Edges Definitions. Once the graph nodes are processed, we consider two potential ways to define edges. First, we can add edges based on the hierarchical organization between SVG elements and their parent containers in the raw SVG file. Alternatively, we can add the groups from Hierarchical Grouping (instead of the SVG container elements) as nodes to the graph, and construct edges based on the relation of subordination in Hierarchical Grouping.

Graph Construction. Combining the above variations of node definition, feature extraction, and edge definition, we present the following four graph representations with increasing amounts of semantic labels:

- SVG-Only (Graph 1): SVG shape and container elements as nodes, and parent-child relationships from the SVG hierarchy as edges;
- SVG-MainChart (Graph 2): main chart marks based on Element Role and SVG container elements as nodes, and parent-child relationships from the SVG hierarchy as edges;
- SVG-MainChart-MarkType (Graph 3): main chart marks based on Element Role and SVG container elements as nodes, with ground-truth one-hot encoding of Element Type as the node-type feature, and parent-child relationships from the SVG hierarchy as edges;
- SVG-MainChart-MarkType-Grouping (Graph 4): main chart marks based on Element Role and groups from Hierarchical Grouping as nodes, with ground-truth one-hot encoding of Element Type as the node-type feature, and group-child relationships from Hierarchical Grouping as edges.

Model Training. Next, we present the experimental details of training GNN models on the chart graphs. First, to prepare the dataset, we get all 40 named types of charts, in total 914 charts, and split them into the training and test sets randomly using a $6:4$ ratio. Then, we construct the four kinds of graph representations for each chart. We use a Graph Convolutional Network (GCN) [41] with three GCN layers followed by a global mean pooling layer and a fully connected layer to output a probabilistic inference over the 40 types. For each model, we perform 5 independent runs to minimize noises; during training, we use the Adam optimizer [40] with a learning rate 0.01 and a dropout [67] probability 0.5 for 200 epochs. The supplementary materials include the code for graph data generation and model training.

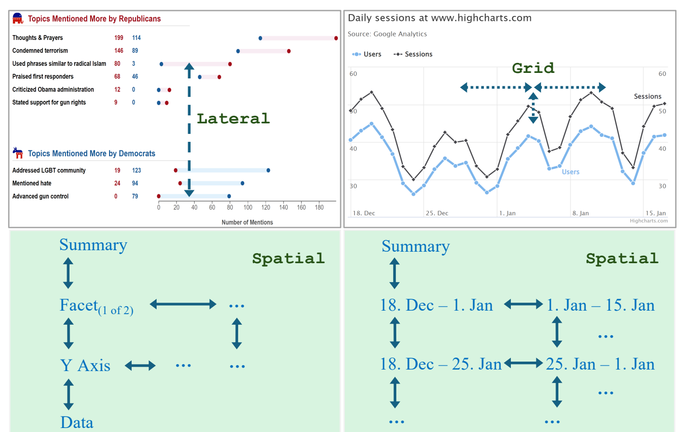
Results. Figure 10 presents the learning curves from the four models regarding the test accuracy. Generally, this 40-class classification task is challenging: with Graph 1 (SVG-Only), the top-1 accuracy is around 26% and the top-5 accuracy is around 55%. With more semantic information revealed to the models, the two metrics are enhanced by Graph 2 (30%, 65% respectively), Graph 3 (40%, 80% respectively), and Graph 4 (50%, 85% respectively), demonstrating the usefulness of Element Role, Element Type, and Hierarchical Grouping from VISANATOMY.

Summary. Our results demonstrate that more semantic labels lead to increased performance of GNNs, indicating the possibility of developing effective mixed-initiative applications powered by GNNs. Because our contribution here is an exploratory study of how the semantic labels can help form graph data and which are useful (instead of providing well-trained models), there still exist many research questions to explore, such as how to embed more semantics into the nodes and what is the best training strategy. We include more discussions in Section 5.

4.4 Content Navigation for Accessibility

VISANATOMY not only serves as a benchmark dataset for evaluating algorithms and models, but also as a valuable resource for researchers to develop visualization applications that use SVG charts as inputs. We demonstrate its utility by replicating a project focused on chart accessibility, which involves designing chart reading experiences for people with visual impairments. Current visualization accessibility practices link textual descriptions of charts (usually provided by the authors through alt texts) to the underlying data tables so that assistive screen readers can communicate some high-level chart semantics to the user [69]. However, this paradigm does not offer visually impaired individuals the same level of chart exploration experiences that sighted people can access through interactive visualizations [75]. In response, Zong et al. [75] propose a chart accessibility tree as the underlying structure for traversal of a chart's scene graph: the user navigates along multi-level branches of the accessibility tree through a keyboard.

Considering that the demos provided by Zong et al. [75] are implemented with Vega-Lite charts, here we demonstrate that the semantic labels in VISANATOMY can support the construction of the accessibility trees for charts created using other tools. To re-create the accessible chart reading experiences, we focus on the five examples in the gallery of Zong et al.'s work (Figure 2 in [75]), and choose five charts from VISANATOMY that have very close visualization designs: a faceted connected dot plot [50] from Mascot.js, a multi-line chart [33] from HighCharts, a geographical heatmap [21] from D3.js, a stacked bar chart [6] from Apexcharts.js, and a bar chart with annotations [24] from *poetrybetweenpain.deb.is*. In our implementation, we focus on the following navigation patterns: `structural navigation` with the **Up**, **Down**, **Left**, **Right** arrow keys, `spatial navigation` with the **WASD** keys across grids, and `lateral navigation` across facets with the **Shift+Left** and **Shift+Right** key combinations. We next briefly introduce how we implemented the accessibility trees (illustrated in Figure 11) for the faceted connected dot plot and the multi-line chart example. We include the construction processes for the other three charts in the supplementary materials.



(a) Faceted Connected Dot Plot (b) Multi-Line Chart

Fig. 11: Navigation patterns re-created in (a) a faceted connected dot plot and (b) a multi-line chart with annotations using semantic labels from VISANATOMY.

Faceted Connected Dot Plot (Figure 11(a)): Starting with the whole SVG chart as the root, we introduce two branches corresponding to the two facets using Hierarchical Grouping. Each branch has several connected dot pairs as children, which are linked to their corresponding y-axis labels in Reference Elements based on their y coordinates obtained from All Graphic Primitives. Once the **Shift + Left/Right** combination is detected, the focus will be shifted to the other facet branch's dot pair of the same index to support the `lateral navigation`. The leaf nodes in the `structural navigation` tree are the individual dots.

Multi-Line Chart (Figure 11(b)): A binary accessibility tree similar to that in [75] is formed. The range of the x-axis labels from Reference Elements is split into two, each is attached to the SVG root node as a child and linked with marks whose x coordinates are in the range. This binary range partition continues until only one mark is inside; the whole binary accessibility tree is then used for the `structural navigation`. We further support `spatial navigation` with the **WASD** keys across grids: x and y coordinates of gridlines from Reference Elements are obtained to cut the coordinate system into 2-dimensional grids, which are regarded as children of the SVG root. The **S** key triggers the `spatial navigation` mode starting with the upper-left grid and the **Up** arrow key brings the user back to the root.

Overall, we find it straightforward to construct the chart accessibility tree using the semantic labels from VISANATOMY: once the tree structure is decided, the mappings between the tree nodes and the graphical objects in the chart can be retrieved with the help of Hierarchical Grouping, Element Type and Role, position and color properties in All Graphic Primitives, and axis and legend information from Reference Elements. The supplementary materials include demo videos and a web application for re-created navigation patterns on the five charts. However, we have also observed a few places where semantics beyond VISANATOMY are required. For example, for the geographical heatmap example, the underlying CSV data from the source website is needed to obtain names of provinces and cities represented by marks; for the annotated bar chart example, the affixation of annotation texts onto the main chart bars needs to be determined in advance. We include more discussion on these issues in Section 5.

5 Discussion and Future Work

Dealing with Real-World SVG Charts. SVG charts found in the wild exhibit considerable noise and heterogeneity, even among charts of the same type. The semantic labels in VISANATOMY significantly reduce the noise, but there are still edge cases we cannot handle properly. For instance, some box-and-whisker plots draw the entire box glyph using a single `<path>` element. Due to the heterogeneity in the SVG structure, the semantic labels for the same type of charts can also vary. For instance, in some bullet charts, the rectangles in the same glyph are overlapping and aligned to one side (e.g., left or bottom), while others stack the rectangles without overlapping (Figure 9(b)). We label the former as a group with a glyph layout, and the latter as a group with a stack layout. The implications of these cases on the downstream applications remain to be explored and better understood.

Extracting Data Tables. In its current state, only 392 charts (out of 942) in VISANATOMY have the underlying data tables available. However, data tables offer important information such as data schema and attribute values that can be useful for applications like visualization redesign and recommendation [36]. In future work, we plan to augment charts that lack underlying data tables by investigating methods to automatically reconstruct these tables using existing labels such as Reference Elements and Visual Encodings. If fully automated approaches are not possible, we plan to extend the current data extraction methods (e.g., ChartDetective [51]) and combine related functionalities with the semantic labels in VISANATOMY to add a **Data Table** stage in our labeling tool, allowing human-machine collaborative curation of the underlying data table from a chart.

Labeling Inter-Element Relationships as Constraints. Some chart scene abstraction frameworks, such as Charticulator [59] and MSC [46] which inspired the semantic labels in VISANATOMY (Section 3.1), have a `constraint` component that describes the spatial relationships between elements. Examples include `align` (e.g., customized alignments in stacked bar charts and glyphs and element ordering in connected scatterplots) and `affix` (e.g., the relative positioning between a mark and its associated annotations). VISANATOMY currently does not have labels on such constraints. The main challenge is that manually linking every pair of elements (e.g., a bar and its text annotation) and specifying their relationships as constraints can be time-consuming and error-prone. Automatic algorithms are needed to recognize and predict such constraints in batch. We plan to investigate such algorithms based on VISANATOMY, and develop novel interaction models to support generalizable labeling of the `constraint` components.

Enlarging the Corpus Size. Although the number of charts in VISANATOMY is comparable to state-of-the-art chart corpora curated using manual approaches (Section 3.4), and the scale of fine-grained labels can support various applications (Section 4), VISANATOMY can be enlarged further with more chart designs and intra-type variations. We will open-source VISANATOMY and our labeling tool to encourage contributions from the visualization community. A significant portion of our time on this project was dedicated to developing and refining the labeling system. Given its current maturity and the fact that labeling one chart now takes 10 to 15 minutes on average, we anticipate that expanding the corpus size will not be overly time-consuming. We have also experimented using synthetic data to augment the size of VISANATOMY. To do this, we (1) converted charts in VISANATOMY to scene templates in Mascot.js [46], (2) generated compatible synthetic datasets, and (3) used Mascot.js to infuse the synthetic datasets with the templates. Example charts are included in the supplementary materials. We decide not to include these generated charts in VISANATOMY: the distribution of the synthetically generated data is best determined according to the specific machine-learning tasks or interactive application goals; it is better that VISANATOMY only contains the original real-world charts, which can be augmented in different ways depending on the use case. More research effort is needed to develop approaches to promoting diversity in the underlying data and the visual styles at the same time, to enhance the underlying visualization engine used for generating charts on synthetic data, and to achieve a desired balance between quantity and diversity. Finally, we intend to investigate methods for learning-based data augmentation. Specifically, we would like to explore if GNN can model the chart layout specified together by Group Layout and Visual Encodings as templates, and transform a given SVG chart into a new one with similar visual designs but representing a new data table. We also plan to experiment with large language models (e.g., fine-tuning LLMs using VISANATOMY) to see if LLMs are capable of helping accelerate the labeling process by regarding the labeling task as a code generation task.

Enhancing the Breadth and Depth of Applications. We plan to explore more visualization applications in greater breadth and depth to unlock the full capacity of VISANATOMY. For example, the GNN models tested in Section 4.3 are homogeneous, having the same feature length for all the nodes. However, given that SVGs are composed of a variety of different elements that have distinct sets of visual properties, heterogeneous GNNs [70], which accept different types of nodes and support node features of unequal lengths, could be a better model candidate. Second, semantic labels from VISANATOMY have the potential to support the development of rigorous approaches to computing pair-wise chart similarities, which can serve as a quantitative measure to evaluate corpus diversity [14], guiding the future collection of chart corpora. Third, there is a thread of research, combining Visualization and Natural Language (NL), to solve tasks such as chart QA [38] and chart captioning [49]. The diverse set of charts in VISANATOMY and the associated semantic labels can be utilized to synthesize datasets that contain high-quality (VIS, NL) pairs, enhancing the generalizability of trained models.

Supporting Interaction and Animation Labeling. Currently VISANATOMY focuses on semantic labels in static charts. A future direction is to annotate interaction and animation. The semantic labels of static components are based on existing visualization abstraction models; we envision that the annotation of interaction and animation also requires solid theoretical foundations on dynamic visualizations. We will investigate how interaction grammars like Vega-Lite [62] may be used and explore additional abstractions.

6 Conclusion

In the paper, we contribute VISANATOMY, a diverse SVG chart corpus encompassing 40 chart types produced by over 50 tools from hundreds of public online sources. Each chart in VISANATOMY is augmented with rich, multi-granular semantic labels including graphical elements' types, roles, and bounding boxes, the hierarchical grouping of elements, the layouts of groups, and visual encodings. We compare VISANATOMY with related corpora to demonstrate its diversity and the richness of the semantic labels. The usefulness of VISANATOMY is evaluated through four applications: shape recognition for SVG elements, chart semantic decomposition, chart type classification, and content navigation for accessibility. Finally, we outline research challenges and opportunities for future work.

ACKNOWLEDGMENTS

The authors wish to thank A, B, and C. This work was supported in part by a grant from XYZ (# 12345-67890).

REFERENCES

[1] Google Images, the most comprehensive image search on the web., 2023. https://www.google.com/advanced_image_search. 2

[2] Plotly: the front end for ML and data science models, 2023. https://plotly.com/. 3

[3] See it, search it — Bing Visual Search, 2023. https://www.bing.com/visualsearch. 2

[4] Mistral-7b-instruct-v0.3, 2024. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3. 6

[5] ApexCharts. Apexcharts.js - open source javascript charts for your website, 2024. https://apexcharts.com/. 3

[6] ApexCharts.js. A Stacked Bar Chart. https://apexcharts.com/javascript-chart-demos/column-charts/stacked/. 8

[7] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, and M. Stonebraker. Beagle: Automated Extraction and Interpretation of Visualizations from the Web. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, article no. 594, pp. 594:1–594:8. ACM, Montreal, 2018. doi: 10.1145/3173574.3174168 2, 3, 5, 6

[8] M. A. Borkin, A. A. Vo, Z. Bylinskii, P. Isola, S. Sunkavalli, A. Oliva, and H. Pfister. What makes a visualization memorable? *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2306–2315, 2013. doi: 10.1109/TVCG.2013.234 5

[9] M. Bostock. bl.ocks.org, 2021. https://bl.ocks.org/. 2

[10] M. Bostock, V. Ogievetsky, and J. Heer. D³ Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011. doi: 10.1109/TVCG.2011.185 2, 3

[11] V. S. Bursztyn, J. Hoffswell, S. Guo, and E. Koh. Representing charts as text for language models: An in-depth study of question answering for bar charts. In *IEEE VIS Short Papers*, 2024. 6

[12] Chartmaker. The chartmaker directory, 2021. http://chartmaker.visualisingdata.com/. 2

[13] C. Chen, B. Lee, Y. Wang, Y. Chang, and Z. Liu. Mystique: Deconstructing svg charts for layout reuse. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):447–457, 2024. doi: 10.1109/TVCG.2023.3327354 1, 2, 4, 6, 7

[14] C. Chen and Z. Liu. The State of the Art in Creating Visualization Corpora for Automated Chart Analysis. *Computer Graphics Forum*, 42(3):449–470, 2023. doi: 10.1111/cgf.14855 2, 3, 4, 7, 9

[15] J. Chen, M. Ling, R. Li, P. Isenberg, T. Isenberg, M. Sedlmair, T. Möller, R. S. Laramee, H.-W. Shen, K. Wünsche, and Q. Wang. Vis30k: A collection of figures and tables from ieee visualization conference publications. *IEEE Transactions on Visualization and Computer Graphics*, 27(9):3826–3833, 2021. doi: 10.1109/TVCG.2021.3054916 5

[16] N. Chen, Y. Zhang, J. Xu, K. Ren, and Y. Yang. Viseval: A benchmark for data visualization in the era of large language models. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1301–1311, 11 pages, Jan. 2025. doi: 10.1109/TVCG.2024.3456320 5, 6

[17] X. Chen, W. Zeng, Y. Lin, H. M. AI-maneea, J. Roberts, and R. Chang. Composition and configuration patterns in multiple-view visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1514–1524, 2021. doi: 10.1109/TVCG.2020.3030338 5

[18] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995. 7

[19] W. Cui, J. Wang, H. Huang, Y. Wang, C.-Y. Lin, H. Zhang, and D. 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 1, 7

[20] W. Cui, X. Zhang, Y. Wang, H. Huang, B. Chen, L. Fang, H. Zhang, J.-G. Lou, and D. Zhang. Text-to-viz: Automatic generation of infographics from proportion-related natural language statements. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):906–916, 2019. doi: 10.1109/TVCG.2019.2934785 1

[21] D3.js. A Geographical Heatmap. https://observablehq.com/@z-richard/choropleth-map-of-the-covid-19-cases-in-china. 8

[22] Datylon. Datylon: Design, automate & share beautiful, on-brand reports, 2024. https://www.datylon.com/. 3

[23] K. Davila, S. Setlur, D. Doermann, B. U. Kota, and V. Govindaraju. Chart mining: A survey of methods for automated chart analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3799–3819, 2020. doi: 10.1109/TPAMI.2020.2992028 2

[24] D. Davis. An Annotated Bar Chart. https://poetrybetweenpain.deb.is/. 8

[25] DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. 6

[26] D. Deng, Y. Wu, X. Shu, J. Wu, S. Fu, W. Cui, and Y. Wu. VisImages: A fine-grained expert-annotated visualization dataset. *IEEE Transactions on Visualization and Computer Graphics*, 2022. Publisher: IEEE. 2, 5

[27] V. Dibia and Ç. Demiralp. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE Computer Graphics and Applications*, 39(5):33–46, 2019. doi: 10.1109/MCG.2019.2924636 1

[28] S. Dou, X. Jiang, L. Liu, L. Ying, C. Shan, Y. Shen, X. Dong, Y. Wang, D. Li, and C. Zhao. Hierarchical recognizing vector graphics and a new chart-based vector graphics dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 5, 6

[29] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 6

[30] Ferdio. The data viz project, 2021. https://datavizproject.com/. 2

[31] J. Harper and M. Agrawala. Deconstructing and restyling D3 visualizations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pp. 253–262. ACM, Honolulu, 2014. doi: 10.1145/2642918.2647411 7

[32] J. Heer, M. Agrawala, and W. Willett. Generalized selection via interactive query relaxation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, 10 pages, p. 959–968. Association for Computing Machinery, New York, NY, USA, 2008. doi: 10.1145/1357054.1357203 5

[33] Highcharts. A Multi-Line Chart. https://www.highcharts.com/demo/highcharts/line-ajax. 8

[34] Highcharts. Highcharts: Interactive charting library, 2024. https://www.highcharts.com/. 3

[35] K. Hu, M. A. Bakker, S. Li, T. Kraska, and C. Hidalgo. Vizml: A Machine Learning Approach to Visualization Recommendation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2019. doi: 10.1145/3290605.3300358 1

[36] K. Hu, S. N. S. Gaikwad, M. Hulsebos, M. A. Bakker, E. Zgraggen, C. Hidalgo, T. Kraska, G. Li, A. Satyanarayan, and C. Demiralp. VizNet: Towards A Large-Scale Visualization Learning and Benchmarking Repository. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–12. ACM, Glasgow Scotland Uk, May 2019. doi: 10.1145/3290605.3300892 9

[37] D. Jung, W. Kim, H. Song, J.-I. Hwang, B. Lee, B. Kim, and J. Seo. Chartsense: Interactive Data Extraction from Chart Images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 12 pages, pp. 6706–6717. ACM, Denver, 2017. doi: 10.1145/3025453.3025957 1, 7

[38] S. E. Kahou, A. Atkinson, V. Michalski, Ákos Kádár, A. Trischler, and Y. Bengio. FigureQA: An annotated figure dataset for visual reasoning, 2018. 9

[39] G. Kazai, J. Kamps, and N. Milic-Frayling. An analysis of human factors and label accuracy in crowdsourcing relevance judgments. *Information retrieval*, 16:138–178, 2013. 3

[40] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 8

[41] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 8

[42] A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Fac-*

tors in Computing Systems, CHI '08, 4 pages, p. 453–456. Association for Computing Machinery, New York, NY, USA, 2008. doi: 10.1145/1357054.1357127 3

[43] H.-K. Ko, H. Jeon, G. Park, D. H. Kim, N. W. Kim, J. Kim, and J. Seo. Natural language dataset generation framework for visualizations powered by large language models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, article no. 843, 22 pages. Association for Computing Machinery, New York, NY, USA, 2024. doi: 10.1145/3613904.3642943 5, 6

[44] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. 7

[45] H. Li, Y. Wang, A. Wu, H. Wei, and H. Qu. Structure-aware visualization retrieval. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, article no. 409, 14 pages. Association for Computing Machinery, New York, NY, USA, 2022. doi: 10.1145/3491102.3502048 1, 2, 7

[46] Z. Liu, C. Chen, and J. Hooker. Manipulable Semantic Components: A Computational Representation of Data Visualization Scenes. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):732 – 742, 2025. doi: 10.1109/TVCG.2024.3456296 2, 3, 9

[47] Z. Liu, C. Chen, F. Morales, and Y. Zhao. Atlas: Grammar-based Procedural Generation of Data Visualizations. In *2021 IEEE Visualization Conference (VIS)*, pp. 171–175. IEEE, New Orleans, 2021. doi: 10.1109/VIS49827.2021.9623315 2, 3

[48] S. Madan, Z. Bylinskii, M. Tancik, A. Recasens, K. Zhong, S. Alsheikh, H. Pfister, A. Oliva, and F. Durand. Synthetically trained icon proposals for parsing and summarizing infographics. *arXiv preprint arXiv:1807.10441*, 2018. 5

[49] A. Mahinpei, Z. Kostic, and C. Tanner. Linecap: Line charts for data visualization captioning models. In *2022 IEEE Visualization and Visual Analytics (VIS)*, pp. 35–39. IEEE, 2022. doi: 10.1109/VIS54862.2022.00016 9

[50] Mascot.js. A Connected Dot Plot. `https://mascot-vis.github.io/gallery#DumbbellChart`. 8

[51] D. Masson, S. Malacria, D. Vogel, E. Lank, and G. Casiez. ChartDetective: Easy and Accurate Interactive Data Extraction from Complex Vector Charts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, number 147, article no. 147, 17 pages. ACM, Hamburg, 2023. doi: 10.1145/3544548.3581113 2, 9

[52] A. M. McNutt. No Grammar to Rule Them All: A Survey of JSON-style DSLs for Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):160–170, 2023. doi: 10.1109/TVCG.2022.3209460 2

[53] Microsoft. Azure AI Vision with OCR and AI - Microsoft Azure. `https://azure.microsoft.com/en-us/products/ai-services/ai-vision`. 2

[54] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2022. doi: 10.1109/TPAMI.2021.3059968 2

[55] T. Munzner. *Visualization analysis and design*. CRC press, 2014. 2

[56] NIVO. Home — nivo, 2024. `https://nivo.rocks/`. 3

[57] Observable. Explore, analyze, and explain data as a team., 2021. `https://observablehq.com/`. 2

[58] J. Poco and J. Heer. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Computer Graphics Forum*, 36(3):353–363, 2017. doi: 10.1111/cgf.13193 1, 2, 5, 7

[59] D. Ren, B. Lee, and M. Brehmer. Charticulator: Interactive Construction of Bespoke Chart Layouts. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):789–799, 2018. doi: 10.1109/TVCG.2018.2865158 9

[60] S. Ribecca. The data visualisation catalogue, 2021. `https://datavizcatalogue.com/about.html`. 2

[61] A. Satyanarayan, B. Lee, D. Ren, J. Heer, J. Stasko, J. Thompson, M. Brehmer, and Z. Liu. Critical Reflections on Visualization Authoring Systems. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):461–471, 2019. doi: 10.1109/TVCG.2019.2934281 2, 3

[62] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2016. doi: 10.1109/TVCG.2016.2599030 2, 3, 9

[63] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer. Revision: Automated Classification, Analysis and Redesign of Chart Images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 393–402. ACM, Santa Barbara, 2011. doi: 10.1145/2047196.2047247 1, 2, 6, 7

[64] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605 7

[65] L. S. Snyder and J. Heer. DIVI: Dynamically Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):403 – 413, 2024. Publisher: IEEE. doi: 10.1109/TVCG.2023.3327172 2, 3

[66] Spotfire. Spotfire: Transforming Data into Real-Time Insights and Actionable Decisions. `https://www.spotfire.com/`. 2

[67] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 8

[68] B. J. Tang, A. Boggust, and A. Satyanarayan. VisText: A Benchmark for Semantically Rich Chart Captioning. In *The Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. 5

[69] W3C (2019). WAI Web Accessibility Tutorials: Complex Images., 2019. `https://www.w3.org/WAI/tutorials/images/complex/`. 8

[70] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, WWW '19, 11 pages, p. 2022–2032. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3308558.3313562 9

[71] Z. Wang, J. Hsu, X. Wang, K.-H. Huang, M. Li, J. Wu, and H. Ji. Text-based reasoning about vector graphics. *arXiv preprint arXiv:2404.06479*, 2024. 6

[72] L. Wilkinson. ggplot2: elegant graphics for data analysis by wickham, h., 2011. 3

[73] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. 6, 7

[74] K. T. Zinat, J. Yang, A. Gandhi, N. Mitra, and Z. Liu. A comparative evaluation of visual summarization techniques for event sequences. *Computer Graphics Forum*, 42(3):173–185, 2023. doi: 10.1111/cgf.14821 3

[75] J. Zong, C. Lee, A. Lundgard, J. Jang, D. Hajas, and A. Satyanarayan. Rich Screen Reader Experiences for Accessible Data Visualization. *Computer Graphics Forum (Proc. EuroVis)*, 2022. doi: 10.1111/cgf.14519 1, 6, 8, 9