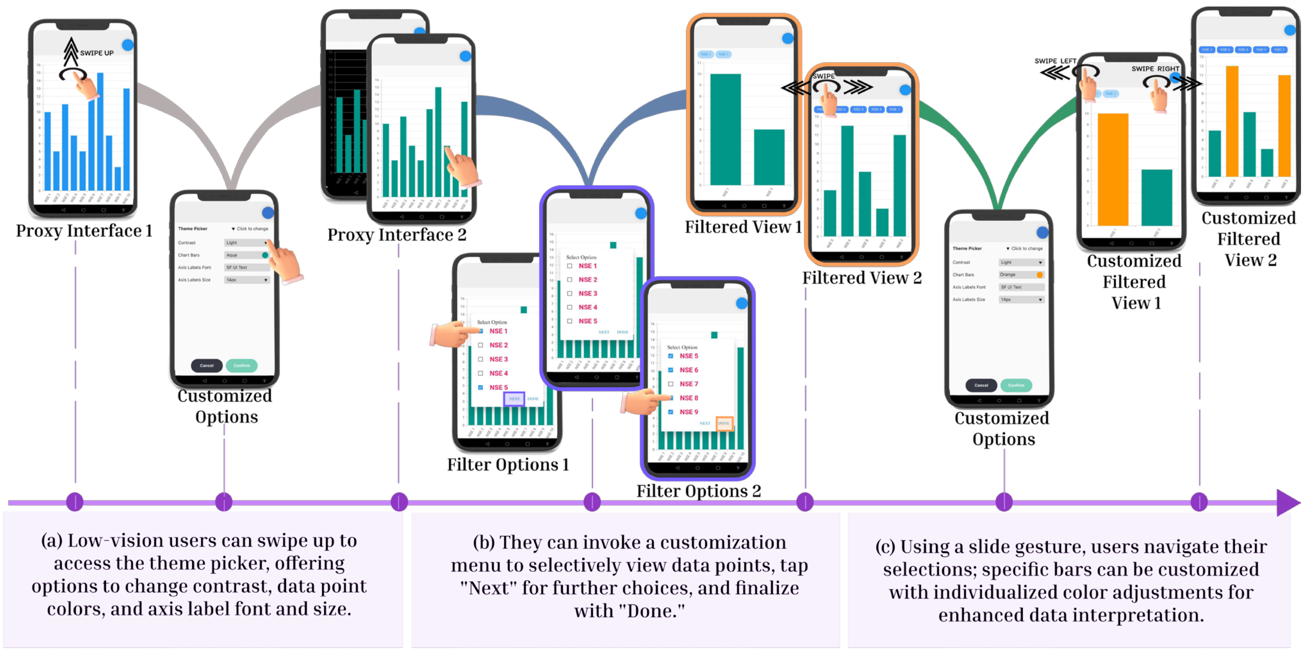
Towards Enhancing Low Vision Usability of Data Charts on Smartphones

Yash Prakash, Pathan Aseef Khan, Akshay Kolgar Nayak, Sampath Jayarathna, Hae-Na Lee, and Vikas Ashok



(a) Low-vision users can swipe up to access the theme picker, offering options to change contrast, data point colors, and axis label font and size.

(b) They can invoke a customization menu to selectively view data points, tap "Next" for further choices, and finalize with "Done."

(c) Using a slide gesture, users navigate their selections; specific bars can be customized with individualized color adjustments for enhanced data interpretation.

Fig. 1: Use scenario for GraphLite: (a) selecting themes, (b) making data choices, and (c) personalizing data appearance.

Abstract—The importance of data charts is self-evident, given their ability to express complex data in a simple format that facilitates quick and easy comparisons, analysis, and consumption. However, the inherent visual nature of the charts creates barriers for people with visual impairments to reap the associated benefits to the same extent as their sighted peers. While extant research has predominantly focused on understanding and addressing these barriers for blind screen reader users, the needs of low-vision screen magnifier users have been largely overlooked. In an interview study, almost all low-vision participants stated that it was challenging to interact with data charts on small screen devices such as smartphones and tablets, even though they could technically "see" the chart content. They ascribed these challenges mainly to the magnification-induced loss of visual context that connected data points with each other and also with chart annotations, e.g., axis values. In this paper, we present a method that addresses this problem by automatically transforming charts that are typically non-interactive images into personalizable interactive charts which allow selective viewing of desired data points and preserve visual context as much as possible under screen enlargement. We evaluated our method in a usability study with 26 low-vision participants, who all performed a set of representative chart-related tasks under different study conditions. In the study, we observed that our method significantly improved the usability of charts over both the status quo screen magnifier and a state-of-the-art space compaction-based solution.

Index Terms—Low vision, Graph usability, Screen magnifier, Graph perception, Accessibility

1 Introduction

Data charts have become commonplace online and in academic and professional settings, given their ability to condense and express complex data in a format that facilitates quick and easy comparisons, trend analysis, and consumption [30]. Popular data charts such as bar and line graphs are used for visualizing stock values, census, product sales, customer ratings, currency exchange rates, and hospitalization rates. Given the widespread use of data charts, they must be usable for people of all abilities, including those with visual impairments. However, data charts are inherently visual; therefore, there is a need to adapt the charts to make them usable for people with visual disabilities who rely on assistive technologies such as screen readers (e.g., JAWS, NVDA) or screen magnifiers (e.g., ZoomText, Apple Zoom).

While there exist a few works that have studied chart accessibility as well as proposed solutions to address the chart usability problems for blind screen reader users [42, 53, 68, 81], the visualization needs of low vision screen magnifier users concerning charts are still an uncharted research territory. Low vision refers to impairments in one or both eyes that cannot be rectified with glasses, contact lenses, medication, or surgery. A salient aspect of low vision condition is poor visual acuity

• *Yash Prakash, Pathan Aseef Khan, Akshay Kolgar Nayak, Sampath Jayarathna, and Vikas Ashok are with Old Dominion University. E-mail: yprak001, pkhan002, anaya001@odu.edu and sampath,vganjigu@cs.odu.edu*
• *Hae-Na Lee is with Michigan State University. E-mail: leehaena@msu.edu.*

Received 31 March 2024; revised 1 July 2024; accepted 15 July 2024.
Date of publication 20 September 2024; date of current version 29 November 2024.
This article has supplementary downloadable material available at https://doi.org/10.1109/TVCG.2024.3456348, provided by the authors.
Digital Object Identifier no. 10.1109/TVCG.2024.3456348

(less than 20/70), so people with low vision typically depend on a screen magnifier to access content via enlargement [16, 62]. However, content enlargement comes with a price – low vision users can only view a small portion of the chart content at any instant, as screen size is limited, especially on smartphones. Low vision users have to, therefore, move their magnifier lens over the content to view the occluded portions, an activity referred to in the literature as *panning*. Thus, interaction with enlarged charts or graphs can be discomforting for low vision users on smartphones even though they can technically "see" the charts since they often have to pan to-and-fro between different portions of the enlarged charts to make data comparisons and comprehend overall trends in data (e.g., see Figure 2).

To uncover and understand the chart usability issues of low-vision smartphone users, we conducted an interview study with 14 low-vision participants with different eye conditions. Despite the heterogeneity of the participant pool, all participants stated that it was arduous and tedious to interact with charts on smartphones, mainly due to two reasons. First, the participants mentioned that they could not easily associate axis labels with data points (e.g., 'y' coordinate value of a bar in a bar chart). Second, the participants specified that it was mentally taxing to compare data points, especially if these points were distant from each other in the chart. A common aspect in both these reasons specified by the participants was that they could not *see* the desired information simultaneously (e.g., axis labels and a bar in a bar graph, desired subset of data points in a bar graph). Therefore, they had to memorize individual pieces of information as they moved the magnifier lens to and fro over different portions of a chart.

To address these chart usability issues on smartphones, we present GraphLite, a mobile assistive technology that enables low-vision users to customize charts and seamlessly navigate through different 'views' of the chart, showcasing select data points within the magnifier window. In essence, GraphLite allows screen-magnifier users to use *selective attention* to visualize various relevant information in charts, e.g., selectively view desired data points for quick comparison and trend analysis across various portions of bar and line charts (see Figure 1), thereby enabling users to perform quick-and-easy visual comparison between desired data points mentally, and consequently reducing the significant amount of panning and cognitive effort. Moreover, GraphLite employs space compaction methods to further improve interaction usability by decreasing horizontal panning and providing a simple one-finger tap gesture-based interface to reduce the dependency on the default two or three-finger zoom-and-pan gestures.

In a user study with 26 low vision participants, we observed that GraphLite reduced the time to perform representative chart tasks compared to the baseline methods – default screen magnifier and a state-of-the-art method that converted charts to tables [13]. The subjective feedback for GraphLite was also significantly more positive than the baselines. All study participants stated that GraphLite significantly reduced their mental burden while interacting with data charts by enabling them to view desired content within the viewport despite magnification.

In sum, our contributions are: (i) The findings of an interview study uncovering the interaction challenges that low-vision users face while interacting with data charts using screen magnifiers; (ii) An assistive technology application that provides an alternative interactive mode for charts, enabling users to customize chart data and selectively view desired data points next to each other; and (iii) The findings of a user study with 26 low-vision participants evaluating the efficacy of GraphLite against state-of-the-art solutions.

2 Related Work

2.1 Low Vision Interaction with Smartphones

Extant research concerning smartphone interaction needs and behavior of people with visual disabilities has predominantly focused on screen reader users [32, 38]. In contrast, prior research on the needs of low vision screen magnifier users is still in its infancy [15, 73]. Szpiro et al. [73] conducted a study to understand the interaction behavior of low-vision screen-magnifier users on touch devices such as smartphones. Their study uncovered multiple accessibility and usability challenges: (i) Many participants struggled to pan back and forth after content enlargement; (ii) The participants had to remember and make use of different multiple-finger gestures to use the screen magnifier accessibility features; and (iii) Uniform content enlargement aspect of screen magnifiers made it challenging to navigate and understand the application content. While this research sheds light on general obstacles faced by low-vision users on smartphones, the extent to which these issues impact low-vision users' experience with charts remains unknown, which will be covered in our interview study (Section 3).

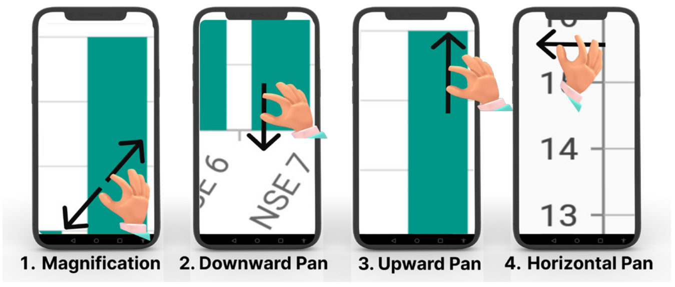


Fig. 2: Low-vision chart interaction using screen magnifier.

Preliminary works exist that propose solutions to improve usability for low vision screen magnifier users [47, 57]. Almost all of these works have based their solution ideas on the concept of context preservation after content enlargement, using some form of space compaction [7, 31].

While the research mentioned above primarily focused on generic aspects of low-vision interaction with smartphones and desktops, they did not directly address the unique low-vision needs associated with visualizations such as data charts. However, the main ideas, such as space compaction and context-preservation, are still helpful, and therefore, we adopted these ideas while designing GraphLite.

2.2 Accessibility and Usability of Data Charts

A few research studies have focused on improving the user experience of blind people while interacting with data charts. These solutions include automatic generation of textual alternatives [34, 37], sonification-based interfaces [2, 29], alternative tactile and multi-modal interfaces [20, 21], question and answer systems [36, 54], and tabular representations of charts [13, 24].

These solutions for improving interaction with data charts primarily target blind screen-reader users. While low-vision users can also use these solutions, they are not specifically tailored for low-vision interaction. Moreover, graphical visual decoding-based issues faced by low-vision users have yet to be thoroughly explored, and there is no present solution inherently designed for low-vision users. We address this gap in this paper by introducing a novel system GraphLite that strives to improve the usability of charts specifically for low-vision screen-magnifier users.

2.3 Data Extraction from Charts

Chart reverse-engineering refers to the process of analyzing and deconstructing a visual chart or graph to understand its underlying data, structure, and the methods used to create it [59]. Plenty of rule-based [23, 35, 59, 66], automated tool-based [12, 35, 55, 56], and deep learning-based [13, 48, 52] extraction techniques currently exist to extract all necessary information from charts.

While the tool-based approaches are effective for data extraction, the need for user engagement in automated tools can cause cognitive overload for low-vision users due to the excessive screen magnifier interaction with the user interface. On the other hand, rule-based algorithms are not scalable for real-world scenarios and have longer processing times. Therefore, for GraphLite, we chose an extraction algorithm using deep learning. Specifically, we employed the ChartOCR method [52], a deep hybrid framework that combines the strengths of deep learning and rule-based methods for data extraction. ChartOCR has demonstrated state-of-the-art performance on bar, line, and pie charts within the custom benchmark dataset ExcelChart400K. This performance surpasses classical rule-based models such as Revision [66], and other deep learning-based models such as Vis [13], ResNet+Faster-RCNN [13, 48], ResNet+Rotation RNN [48], and even commercial products such as Think Cell [75]. Note that GraphLite is not tied to ChartOCR per se; any chart data extraction technique can be used in place of ChartOCR.

2.4 Responsive Visualizations

Responsive visualization design involves creating multiple versions of data visualization in order to accommodate different screen sizes and device types [28, 39]. A plethora of prior research works have explored various ways in which visualizations can be adapted for 'smaller screens', primarily focusing on visual elements and structure [18,41,77], as well as interaction methods [40, 43, 70].

Hoffswell et al. [28] proposed a system that provides immediate cross-device previews, enabling designers to see the impact of their edits across multiple devices in real-time, and supports the propagation of successful edits to other views, ensuring a cohesive user experience.

While this approach provides chart designers with hands-on experience in designing multiple responsive versions, it can be tedious and often requires multiple design iterations. To mitigate these challenges, automated tools such as MobileVisFixer by Wu et al. [77] have been proposed. MobileVisFixer utilizes a Markov Decision Process model to automatically redesign SVG-based visualizations, improving their readability and usability on smaller screens.

While the ability to automate certain edits, like repositioning legends, may be widely useful, authors may still need to manually edit visualizations in ways that are difficult to automate, such as rewriting text annotations. To address this, Kim et al. [40] developed Dupo, a 'mixed-initiative authoring tool' designed to streamline the creation of responsive visualizations across different screen sizes. The Dupo interface integrates manual editing tools with automated design suggestions, which allows users to customize designs, manage edit history, and explore responsive suggestions while also providing additional controls for fine-tuning and quick edits.

These responsive visualization solutions primarily focus on adaptability for sighted users, ensuring that visualization charts remain informative and legible across various device platforms, such as mobile and desktop. While low-vision users can benefit from these responsive visualizations, their needs are not fully addressed. They require additional forms of responsive visualization tailored to their specific needs. In this research, we explore various usability issues faced by low-vision users, formulate design requirements, and build GraphLite to cater to the unique needs of low-vision users.

3 Low Vision Usability Issues with Data Charts

We conducted an IRB-approved interview study with 14 low-vision screen magnifier users to uncover their interaction issues with data charts on smartphones. We specifically gathered information on participants' experiences with data charts, such as their frequency of encounters, common settings of interaction, and the chart types they usually engage with in daily life. Examples of seed questions included: *What problems do you face when interpreting data charts?*, *In what type of charts do you face the most problems?*, and *How do you work around these issues?*. The collected interview feedback was then qualitatively analyzed using an open coding technique [65], where we iteratively reviewed the user responses and identified key insights, pain points, and themes that reoccurred in the data.

3.1 Findings

In the interviews, 6 participants frequently reported encountering bar charts, while 5 participants mentioned line charts. These charts were predominantly observed in news articles (by 5 participants), social media (by 4 participants), and blogs (by 4 participants). Given their familiarity and frequent exposure to these charts, participants' feedback and responses were tailored around bar and line charts. Therefore, in this paper, we focus primarily on bar and line charts.

(a) Associating bar chart data points with corresponding axis labels under magnified view is arduous. A majority (12) of the participants stated that it was often tedious and cumbersome to find the label values of data points in bar charts and line graphs. This can be explained as follows: for sighted users, the perceptual effort to associate label l with bar b on the primary key axis of graph g is computed as 230 units for the saccadic movement from the bar to the label [19]; 150 units for the discrimination of the label [49]; and 300 units for word recognition [33]. This cognitive burden is amplified for low-vision individuals because they have shorter and more frequent saccades and fixations [76]. However, other considerations also impact the total effort required, such as the loss of visual context when panning and the issues arising from high zoom levels. Specifically, a slight misalignment of the magnification window during panning can cause the disappearance of contextual information, as noted by (7) participants. They elaborated that precise navigation of the magnifier lens in a straight line from the data point to the axes was necessary for accurately estimating the data point's values. Any disruption in focus during this process usually meant restarting the estimation process entirely.

(b) Visually comparing data points that are far apart in bar/line charts is too difficult. Many participants (8) mentioned that visually comparing data points demanded slow, precise, and concentrated manual movement of the magnifier lens, which was mentally and visually tiring. All participants mentioned that making comparisons between data points that were far apart from each other was extremely difficult, mainly due to the magnification-induced loss of spatial/visual relationships between the data points and the increasing difficulty of comparing non-adjacent data points – a phenomenon known as separation effect loss [74]. For bar charts, 10 participants stated that attention predominantly gravitated towards taller bars, sometimes resulting in the oversight or complete neglect of shorter ones. Additionally, taller bars often cast a visual "smudge" in the space around them. When the user tries to identify the height of a tall bar, the smudge on top of the bar is misinterpreted as part of the bar, causing participants to overestimate its height. Regarding line charts (9), participants stated that issues like smudging and blurring persisted, complicating the analysis of trends. While participants could grasp a general sense of the chart, pinpointing the precise slope of the lines was arduous.

(c) Need to individually memorize data values for making comparisons between data points. One of the strategies mentioned by 6 participants to overcome data-comprehension difficulties in a bar/line chart was to determine and memorize the (axes) values of data points one at a time and then compare them mentally to comprehend the differences. Four participants explained that this strategy was always required for charts with many data points. Two participants even felt that if the number of data points was high, there was little benefit gained by visualizing this data as charts since they had to remember and mentally compare raw values.

The findings from the interview study clearly demonstrate the need for a tailored solution that addresses the specific challenges faced by low-vision users when interacting with data charts.

4 GraphLite Architecture and Interface Design

4.1 *Design Considerations and Requirements*

The design of GraphLite was informed by insights gathered from interviews with low-vision users and a review of prior research.

Visual Memory and Selective Focus. When sighted individuals encounter visual data, the transfer of information to working memory from the sensory memory is influenced by two main factors: the distinct features of the visualization and the viewer's deliberate focus. The completion of processing in working memory allows for the storage of information in long-term memory. The act of selectively focusing on specific information during the working memory phase is what determines the content stored in long-term memory. This focused attention is characterized by an effort to disregard certain stimuli or aspects deemed irrelevant while concentrating on those considered important [51].

For people with low vision, capturing information in sensory memory is different; they cannot observe distinct features of the chart due to impairments that cause smudging or blurring (see section 3.1(b)). Moreover, their focus is diverted to easily noticeable features, like taller bars in bar charts, or it might be divided due to the need to separately examine different components, such as labels and axis values (see section 3.1(b)). To supply working memory with accurate information, they depend on two methods: (1) visual aids like magnification tools, which introduce several challenges specifically due to loss of spatial/visual context (refer section 3.1(a)), and (2) adaptive techniques, including auditory feedback, such as alternative text or data presented in table format, which essentially bypasses the visual advantages of graphical representations. For low-vision users, the information retained in the long-term memory might be detached from the actual representation of the chart (refer section 3.1(c)).

Optimizing the Locus of Selection. To effectively address the unique challenges faced by low-vision users in processing visual information, we focused on creating a design approach that optimizes the "locus of selection"-—the critical stage at which specific information is chosen for deeper cognitive processing [45], drawing inferences from "perceptual load theory" principles, which involves breaking down visual tasks into smaller, manageable segments and integrating multimodal feedback to substantially improve selective attention [44].

Customizable Visual Attributes for Early/Late Selection. Customization options for visual attributes such as color, contrast, font style, and size are necessary for low-vision users. This initial customization serves as an effective cognitive filter for "early selection," which involves the screening of sensory input at the onset before detailed processing [45]. This allows users to minimize visual disturbance from the very beginning. This is backed by prior research, e.g., by Wurm et al. [78], who found that although acuity and color are independent of each other, color improved object recognition for low-vision users and it also improved the overall interaction experience with faster reaction times (353ms color advantage [78]).

Moreover, graphical comprehension involves interpolating data in charts and understanding various underlying relationships within data; this can also be referred to as "read between data" (i.e., integrating and interpreting) [64]. While existing solutions for low-vision users generally "read beyond data" (i.e., generating and predicting), the flavor of preserving chart semantics is typically lost in the process (refer section 2). To bring back the ability to "read between data" for low-vision users, GraphLite allows them to selectively view and trim down charts. Therefore, support should also be provided for "late selection", which involves diving deeper after an initial overview [45], e.g., by enabling selective focus on crucial data points. Furthermore, focusing solely on specific data points through single selective attention is insufficient for comprehensive analysis; a multi-attention mechanism plays a critical role in enhancing interaction with complex visualizations. This can be achieved via multiple 'views' of a single data chart, empowering users to undertake several rounds of "late selection". GraphLite was designed to accommodate the above requirements as explained next.

4.2 GraphLite Overview

Figure 3 presents an architectural schematic illustrating the workflow of GraphLite prototype. When the user loads a webpage in GraphLite, it leverages a custom-trained Inception-V3 model [79] to proactively identify the charts on the webpage as well as their types (e.g., bar chart, line chart). After recognition, GraphLite automatically extracts all information (e.g., labels, legends, data values, etc.) from the charts using an extended ChartOCR model [52]. Next, when the user selects a data chart with a single tap gesture, GraphLite generates an accessible proxy interface specifically designed for low-vision users.

A one-finger long-press gesture [5] on the proxy interface automatically opens up selection options, which can used to pick and view only a few data points of interest. Users can create multiple such 'views' if desired, by tapping on the 'NEXT' option. When the user finally selects the 'DONE' button, the first view is presented to the user by default. The user can navigate to the other self-created views using simple left/right swipe gestures. In addition, the proxy interface also provides an assortment of customization options to set the color, background, font, etc., to further improve usability. GraphLite also applies space compaction while rendering views of charts in the proxy interface to maximize the utilization of screen space, thereby reducing the user's

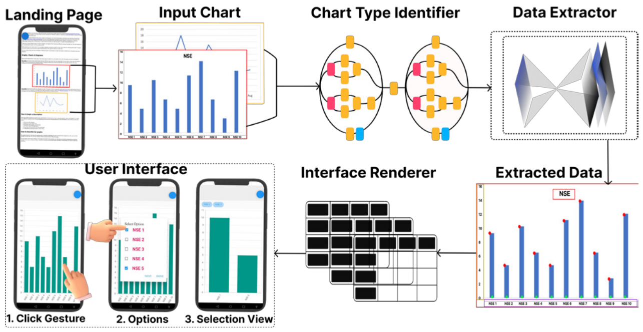


Fig. 3: Architectural schematic of GraphLite.

panning effort. The full implementation details, including front-end and backend components, along with the corresponding code, are all available on GitHub[1].

4.3 Chart Data Extraction

We used the robust ChartOCR method [52], a deep hybrid framework that takes advantage of both deep learning and rule-based methods for chart data extraction. We leveraged the CornerNet architecture [46], incorporating a 104-layer HourGlass network [58] as the backbone for ChartOCR to efficiently extract key points in bar and line charts, respectively. For example, the key points for the bar chart are the top-left and bottom-right corners of each bar. GraphLite then groups each of the top-left and bottom-right key points sequentially to obtain bounding boxes of bars. It also computes the height of each bar using these top-left and bottom-right key points. Similarly, for line charts, the key points are the pivot points along the lines. GraphLite then uses a hierarchical clustering algorithm to group these key points into their respective lines. Once grouped, the key points are correctly associated with their lines, enabling the reconstruction of the lines on the chart. GraphLite then leverages the OCR engine AWS-Rekognition (AWS-Rekognition DetectText API) [4] to extract relevant features from charts (e.g. title, axis labels, legends, scale, etc.). GraphLite then stores all the data values and their corresponding x-axis labels, as well as other relevant features from the chart, in a JSON format.

Training. We trained ChartOCR [52] for bar and line charts on Nvidia V100 GPU with 128GB memory per node, where the learning rate was set to 0.0025, we further decreased the learning rate by a factor of 10 for the last 5000 batches. The overall batch size was set at 27 and the total step size was set to 450000. For optimization, we made use of the well-known Adam optimizer [80]. We employed an early-stopping strategy during the model training process to optimize performance and prevent overfitting.

4.4 Proxy Interface Design

We designed the GraphLite's proxy interface by adhering to the accessibility guidelines proposed by Alcaraz et al. [1]. As illustrated in Figure 1, we designed the user interface of GraphLite to be navigable with simple one-finger gestures in contrast to the status-quo two-finger slide gestures offered by in-built OS accessibility services. To access the proxy interface, the user simply needs to tap on a chart in the current webpage. In the proxy interface, the user can execute a simple upward swipe gesture to invoke the 'Theme picker' to customize the appearance of the chart according to their preferences. A long press-and-hold gesture on the proxy interface pops up a check box interface with all the chart's x-axis labels. To navigate the list of options, the user can simply swipe down the pop-up interface.

To select any x-axis labels of choice, the user must tap on the corresponding check box. Following a selection, pressing the 'NEXT' button triggers the interface to refresh, allowing users to make additional selections as needed. This iterative selection process continues until the user is satisfied and decides to conclude by pressing the 'DONE' button. Upon finalizing their selections, GraphLite integrates a space compaction algorithm[1] for both bar and line charts, optimizing element

[1]https://github.com/accessodu/GraphLite.git

spacing under magnification. This algorithm adjusts the spacing and scaling of bars and line segments to maximize screen space usage, presenting a curated visualization that incorporates only the selections specified by the user. These customized views can be navigated using swipe gestures, enabling users to peruse through various chart visualizations that reflect their chosen options via the proxy interface, as shown in Figure 1.

Moreover, this interface extends its customization capabilities by allowing users to directly interact with individual data elements, such as bars in a bar chart or lines in a line chart. After tapping on these elements, users can invoke the 'Theme Picker' once again to apply distinct color changes to selected data points. To close the GraphLite proxy interface, the user can simply tap the blue button on the interface. Note that GraphLite does not block access to the original chart visualization; this is by design to ensure that the user always has a backup option in case of errors, e.g., incorrect data extraction by the GraphLite's data extraction algorithm. The implementation and technical details of GraphLite are provided in the appendix, and a video demonstrating GraphLite in action and explaining the interaction workflow is available on GitHub[1].

4.5 Implementation details

We implemented GraphLite as an Android mobile browser application developed using Flutter open source framework [22]. When the user loads a webpage, GraphLite leverages in-built Dart functions [17] to extract the DOM of the webpage and send it to the backend server via a POST request. Beautiful soup [61] Python package was used to extract all images in the DOM, which were then labeled with positional IDs. The images were then sent to a custom-trained Inception-V3 model [25]. Following this, images were annotated with flags (True, False) based on whether they were data charts or not. All chart images were sent to the ChartOCR [69] to extract data attributes. The respective chart IDs, flags, and attributes were packaged into a JSON object and sent to the Flutter module. When the user taps on a chart, the syncfusion flutter charts [72] package uses the chart data in the JSON object to recreate a new chart in the proxy interface. Additional functionalities, including typography adjustments, data point selection via checkboxes, swiping through various views, and customizing the color of individual data points, were implemented using Flutter's built-in features. The flutter inappwebview package [50] was utilized to integrate web content and enable interactions with charts within the app. To establish a communication channel between the Flutter app modules and the backend server modules, we used the Flask REST API [60].

5 Evaluation

We conducted an IRB-approved user study with low-vision screen magnifier users to assess the efficacy of GraphLite. We managed to recruit 26 low-vision participants (16 female, 10 male) for the study[2]. Full participant demographic details are available on GitHub[1].

5.1 Design

In a within-subject experimental setup, the participants were asked to perform representative chart tasks under the following conditions:

- Screen Magnifier (*SM*) – The participants used the status quo screen magnification accessibility features to do the tasks.
- Tabular Representation (*TBL*) – The participants could interact with tabular representations of the charts to do the tasks. This condition was chosen to represent extant solutions that convert charts to tables for better accessibility, as seen in example [13].
- GraphLite with only Space Compaction (*SC*) – The participants could leverage only the space compaction feature of GraphLite.
- GraphLite with Space Compaction and Configuration (*SCC*) – The participants could leverage both space compaction and customization (color, contrast, font) features of GraphLite.
- GraphLite with Space Compaction, Configuration and Feature selection (*SCCF*) – The participants could leverage all features.

[2]The typical size of low-vision user studies is between 12 to 20. We enrolled slightly more participants due to the relatively higher number (5) of conditions in our study.

In our study, participants engaged in a series of tasks tailored to assess their interaction with bar and line charts. For designing the tasks, we consulted prior work in information visualization [63] that has introduced various taxonomies categorizing tasks that connect visualization techniques with user cognitive processes. The insights from our earlier interview study helped identify the relevant taxonomies for this study and directly informed our choice of tasks. Specifically, we found Amar et al.'s taxonomy [3] of low-level tasks to be most fitting for our study. This taxonomy includes a series of tasks designed to require minimal reasoning about the data, thereby making it ideal for our study's focus. The chosen tasks were:

- **Task 1: Pairwise Comparison** required participants to compare predetermined data points.
  - The first subtask (**SBC**) involved comparing a single pair, such as sales from Wednesday to Saturday.
  - The second subtask (**MBC**) involved comparing multiple pairs, like sales from Thursday to Saturday, Monday to Friday, and Tuesday to Sunday.
- **Task 2: Selective Filtering** focused on data filtration, where participants identified data entries meeting specific criteria.
  - The first subtask (**SBF**) involved pinpointing days when sales fell below 1 million.
  - The second subtask (**MBF**) required filtering a range of sales figures to identify multiple data points, such as finding days when sales were between 1 and 2 million.
- **Task 3: Trend Prediction** was adapted to emphasize trend prediction (**LTP**), asking participants to predict sales trends of a product across the following month based on prior monthly data.
- **Task 4: Trend Comparison** was introduced, focusing on trend identification and comparison (**LTC**). For example, participants were tasked with identifying the overall trend in stock prices over a defined range of months and comparing the trend with another range of months.

To reduce the impact of confounding variables such as the learning effect, we used different bar and line charts for the five study conditions, i.e., for Task 1 and Task 2, we curated ten bar charts, five for each task. We ensured that all ten bar charts were similar, with each chart containing 20 data points (i.e., bars) and having identical initial formatting in terms of color, contrast, font, and spacing. For Task 1, we pre-selected data points such that the first data point was randomly picked from the leftmost three data points in the bar chart, and the second data point was randomly picked from the rightmost three data points. This selection strategy ensured participants interacted with the majority of the chart data while maintaining a consistent separation effect across tasks, thereby mitigating potential confounds as noted in Talbot et al. [74]. A similar approach was followed for Task 2. For the line charts, we employed the same methodology in chart creation, ensuring each chart contained 20 data points. For Task 3, participants were required to predict the trend following the 20th data point. In Task 4, we selected trends from different parts of the charts for comparison, ensuring that users traversed through most of the charts. The trends were chosen based on identifying the best possible correlations between line segments, thereby facilitating a comprehensive analysis and interaction with the chart data. We also ensured that the chart data extraction was accurate for all these graphs. The assignment of charts to conditions and the ordering of conditions were counterbalanced to the best extent possible using the Latin square method [8].

5.2 Procedure

The user study was conducted over a month to accommodate 26 participants. Three sessions were available each day from Monday to Friday, with each session lasting between 2 to 2.5 hours. Two experimenters were available per session, each accommodating one participant. Participants had the flexibility to choose any of these sessions and were required to attend three sessions to complete the study. The experimenter began by obtaining formal consent from each participant and briefly explaining the study's goals. Participants were then introduced to the TBL and GraphLite interfaces and took part in practice sessions. In the study, each subtask—SBC, MBC, SBF, MBF, LTP, and LTC—was allotted 10 minutes completion time. After completing each subtask, participants filled out the SUS and NASA-TLX questionnaires to capture their usability and workload perceptions. The experimenter also noted user-interaction behaviors and collected qualitative feedback in exit interviews. All study activities and data were recorded with consent, and participants received an Amazon gift card. Further details on the participant's demographics, apparatus, methodology, Charts used in the study, and the schedule of the study are available on GitHub[1].

5.3 Data Analysis

From the study data, we computed the following metrics for each study condition: (i) Task completion times, (ii) Task completion rates, (iii) Task performance accuracies, (iv) SUS usability scores, and (v) NASA-TLX workload scores. We then used these metrics to compare the different study conditions using standard statistical tests and determine if there was a significant positive impact of GraphLite on the overall performance and user experience of the participants. For analyzing the subjective exit-interview feedback and experimenter notes, we adopted a qualitative analysis method, specifically an open coding technique followed by axial coding [65], where we iteratively went over the user data and identified recurring observations and insights.

5.4 Abbreviations

For convenience, Table 1 lists all the abbreviations and placeholders that will be used in the rest of the paper to present the results.

Table 1: List of Abbreviations and Placeholders.

SM : Screen Magnifier	MBC : Multi-bar Comparison
TBL : Table of Content	MBF : Multi-bar Filtering
SC : Space Compaction	LTC : Line Trend Comparison
SCC : SC + Customization	TLX | SUS : Usability Scores
SCCF : SCC + Feature Selection	Task-1 : Pairwise Comparison
SBC : Simple Bar Comparison	Task-2 : Selective Filtering
SBF : Simple Bar Filtering	Task-3 : Trend Prediction
LTP : Line Trend Prediction	Task-4 : Trend Comparison

5.5 Results

5.5.1 Task Completion Times

We measured task completion time as the time (in seconds) a participant took to do a task under a given condition. If a participant failed to complete a task, then the maximum allotted time (i.e., 10 minutes) was considered as the completion time. The results are presented in Figure 4. Overall, there was a significant impact of the study condition on the task completion times (Friedman test, $\chi^2 = 74.6$, $p < 0.001$).

Bar Charts: Task 1 For the SBC subtask, we observed that the SCCF condition had the best performance (Mean: 235.1s, Median: 239.5s, Min: 122s, Max: 355s) whereas the SM condition exhibited the poorest performance (Mean: 531.34s, Median: 552.3s, Min: 423.5s, Max: 600s) among all the study conditions. Pairwise tests between conditions showed that the SCCF condition yielded significantly better results compared to all other study conditions except the TBL condition (Post-hoc Conover's test with Benjaminyi-Hochberg FDR adjustment, SCCF vs. TBL: $p = 0.07$). We also observed that the TBL condition significantly outperformed the SC and SCC conditions (TBL vs. SC: $p = 0.001$, TBL vs. SCC: $p = 0.002$).

Similar observations were made for the MBC subtask; however, in contrast to the SBC subtask, we noticed a significant difference between the SCCF condition (Mean: 326.4s, Median: 317.9s, Min: 188.7s, Max: 442.7s) and the TBL condition (Mean: 421.7s, Median: 429.8s, Min: 355s, Max: 535.5s).

This observed trend suggests that as tasks grow in complexity with an increased number of comparisons and variables, the effectiveness of the TBL condition starts to diminish, whereas the SCCF condition exhibits a noticeable performance improvement. This observation highlights the potential of the SCCF approach in handling complex analytical tasks within graphical data interpretation, especially as the demands of data analysis become more intricate.

Bar Charts: Task 2 For the second task, average task completion times revealed that when participants were asked to engage with smaller ranges of data for selection, for example, if the participant was to identify stocks in the range of 3 million to 5 million revenue, a simple filtration task, the SCCF condition (Mean: 422s, Median: 422.6s, Min: 341.9s, Max: 501s) and the TBL (Mean: 446.6s, Median: 452.2s, Min: 376s, Max: 512.3s) condition showed nearly similar performance. However, as the scope of the data range for filtration expanded, for example, if the participant was asked to filter stocks in the range of 3 million to 8 million revenue, a notable enhancement in performance under the SCCF condition (Mean: 377.6.4s, Median: 348.2s, Min: 265.7s, Max: 532s) was observed Vs. TBL (Mean: 476.9.4s, Median: 489.6s, Min: 377.8s, Max: 579.8s) (Post-hoc Conover's test with Benjaminyi-Hochberg FDR adjustment, SCCF vs. TBL: $p < 0.001$).

Summary: A closer inspection of the recorded data revealed that while performing the tasks in SM, SC, and SCC conditions, the participants often had either concentration lapses or accidentally executed incorrect panning gestures, so they had to repeat the process multiple times to complete the task. In the TBL and SCCF settings, horizontal panning issues were markedly reduced, streamlining the task execution process.

Nonetheless, the TBL condition exposed a notable drawback when participants performed both tasks with increased complexity. Here, participants faced difficulties in retaining and recalling specific data values, often necessitating backtracking for verification. This requirement for frequent memory recall and verification introduced additional cognitive load, leading to uncertainty and inefficiency among participants as they cross-checked and reassured themselves of the data points in question for both tasks, thereby increasing overall completion times.

Line Charts: Task 3 For the third task, the SCCF condition demonstrated the shortest average completion times (Mean: 243s, Median: 267s, Min: 166.5s, Max: 310s), outperforming the SM condition, which had the longest times (Mean: 528.9.6s, Median: 544s, Min: 435.5s, Max: 588s). Statistical analysis revealed a significant advantage of the SCCF condition over all others, according to post-hoc Conover's test with Benjamini-Hochberg FDR adjustment.

Line Charts: Task 4 For the fourth task, SCCF maintained its superior performance with the best average task completion times, distinctly better than the SM condition and the others in comparative analysis (SCCF versus SM: $p < 0.001$, SCCF versus SC: $p < 0.001$, SCCF versus SCC: $p < 0.001$, SCCF versus TBL: $p = 0.03$). These findings consistently indicate that SCCF is the most effective condition for conducting trend analysis. Additionally, the TBL condition showed improvement in the second task, narrowing the performance gap with the SCCF condition compared to the first task by 67%.

Summary: In the SM, SC, and SCC conditions, participants were required to continuously track trends with the aid of a magnifier, moving them in a non-linear fashion. Any deviation in the path not only increased the task completion time but also significantly compromised the accuracy of their analysis.

The TBL condition naturally made it challenging for users to discern and predict trends due to the increased cognitive load. For instance, predicting a trend over 12 months necessitated participants to recall trends month by month, mentally collating these values into a cohesive trend. This process often had to be repeated multiple times to ensure accuracy and confidence in their judgments. However, in the SCCF condition, they could swiftly navigate through multiple views by bucketing selective data points in groups across 12 months. A simple swipe gesture allowed them to observe trends without the need to meticulously

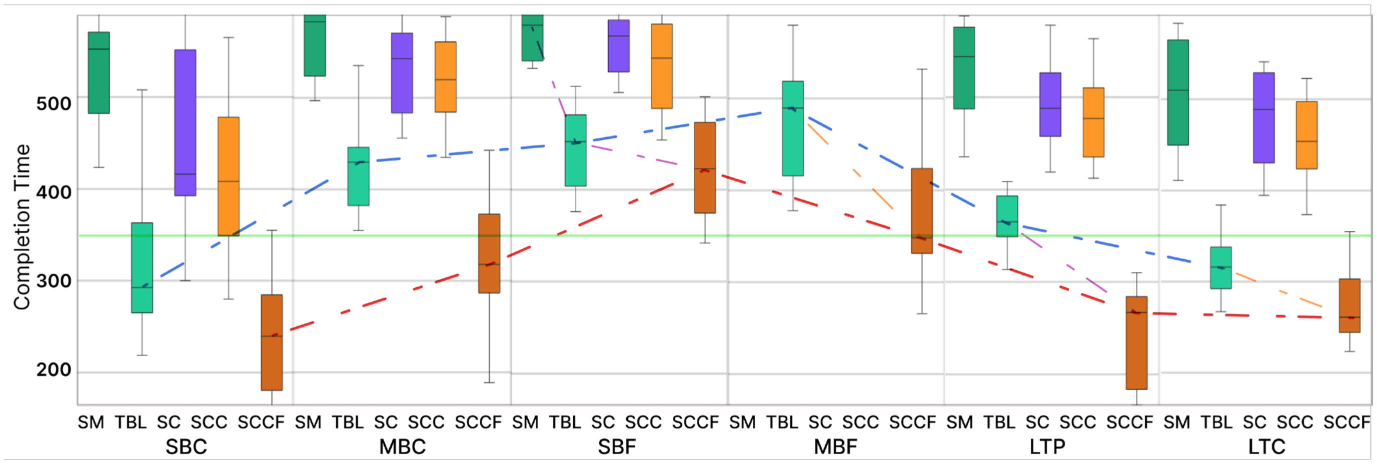


Fig. 4: Task completion time statistics, in seconds, for all the tasks and study conditions. Note: For the MBF task, only the TBL and SCCF conditions are included, as participants were unable to complete this task within the time limit under the SM, SC, and SCC conditions.

consider each underlying value, thus simplifying predictions.

Compared to the SCCF condition, the TBL condition showed improved performance in tasks involving the comparison of multiple trends rather than predicting trends. This improvement can be attributed to the ability of users in the TBL condition to categorize and compare values within specific ranges without needing to assess the entire trend from start to finish. However, for the analysis of trends in line charts, the SCCF condition still proved to be superior. This is because it allowed for a more streamlined approach to trend analysis, enabling users to focus on broader trend patterns rather than getting bogged down by the need to remember and map each data point individually. This strategic division of attention, facilitated by selective focus in the SCCF condition, significantly reduced the cognitive load of trend analysis.

5.5.2 Task Completion Rate and Accuracy

The overall task completion rate for a condition was computed as the percentage of tasks completed by the participants in that condition. Overall, the SCCF and TBL study conditions had the best completion rates (100%), whereas the baseline screen magnifier (SM) condition had the lowest average completion rate(61.5%). These differences in completion rates between the study conditions were found to be statistically significant (Friedman test[3], $\chi^2 = 28.68$, $p < 0.001$).

A closer analysis of experimenter notes and manual inspection of screen-captured videos revealed that the task completion failures were mainly due to two reasons: (i) The participants were unable to complete the task within the stipulated time limit of 10 minutes, and (ii) The participants complained about *stress* and *fatigue* while giving up in the middle of the task. The latter reason was more common in the SM (4 participants) and SC (3 participants) conditions. In these conditions, the participants frequently mentioned that "there were too many bars" and that it was difficult to keep track of and remember the pre-specified task bars and their y-axis values. The completion rate was slightly better in the SCC condition, where some of the participants mentioned that it was "easier on their eyes" after changing bar colors and contrast. In the TBL and SCCF conditions, all participants completed the tasks.

To evaluate the precision of participant responses across the four distinct tasks—comprising two for bar charts and two for line charts, we adopted specific accuracy metrics tailored to the nature of each task. For bar charts, Task 1's accuracy involved comparing participants' data point comparisons to actual values, while Task 2 evaluated the accuracy of identifying points that fit given criteria. Line charts Task 3 measured accuracy by matching participants' trend predictions to real data. Task 4's accuracy was binary, based on correctly identifying and comparing trends. These tasks assessed participants' analytical skills across both chart types. Note: The accuracy was noted without the inclusion of the complexity of tasks.

[3]We used this non-parametric test as the data was not normally distributed.

Bar Charts: We noticed a significant difference in the participants' estimation errors between the different study conditions across both the tasks (Friedman test[4], $\chi^2 = 52.88$, $p < 0.001$), with the TBL condition having the lowest average error of 4.8% (and therefore the best performance) for task 1 comparison and nearly the same average error as SCCF condition of 9.7% for task 2 filtration and the SM condition had the highest average error(18.4% in Task 1 and 19.3% in Task 2) for both the tasks. The TBL condition saw a reduction in performance during task 2 particularly when increasing the complexity of the tasks as participants had to pan through the table multiple times and often mixed the rows mainly due to magnification misalignment. In the remaining three conditions (SM, SC, and SCC), we observed that on many occasions, the participants could not view both the y-axis labels and a taskbar at the same time due to enlargement, and therefore, they had to pan right-to-left along a straight line while mentally visualizing the bar height, to estimate the bar's value or to filter out bars within a range. We believe most errors were introduced during this process, possibly due to imperfect horizontal panning and/or slight lapses in concentration during panning.

Line Charts: In our examination of line charts across various experimental setups and two distinct tasks centered on trend analysis, we noted differences in estimation errors that deviated from those seen with bar charts. The SCCF condition had the lowest average error of 7.1% (and therefore the best performance), and the SM condition had the highest average error of 21.1% (worst performance) for the task. The superior performance observed in the SCCF condition is largely due to participants adopting a strategy that involved grouping their selections into particular value ranges for analysis. This approach allowed them to quickly evaluate the types of trends by observing shifts in trends across different sections of the charts efficiently. The higher average error observed in the TBL condition, as compared to the SCCF, can be attributed to the inherent demands of tabular data interpretation. The TBL condition necessitated a sequential and meticulous approach from users to extract information, leading to the identification of only a select few trends, with many potentially relevant patterns overlooked. On the other hand, the SCCF condition, by leveraging visual representations, offered a more holistic and efficient means for trend analysis. Visual charts support the brain's ability to process information in parallel, allowing for a more comprehensive and effective recognition of trends without overloading cognitive capacities.

In the other conditions (SM, SC, and SCC), we observed a pattern of errors similar to those identified in bar charts. This consistency suggests that the underlying factors contributing to inaccuracies in these conditions are not unique to the type of chart but rather indicative

[4]We used this non-parametric test because the raw data was not normally distributed as per Shapiro-Wilk test.

of broader challenges in data interpretation and analysis using screen magnifiers that affect both bar and line charts similarly.

5.5.3 Perceived Usability and Task Workload

We used the standard SUS questionnaire [9] to measure perceived usability. This questionnaire consists of 10 alternative positive and negative Likert statements where the participant has to provide a rating between 1 (strongly disagree) and 5 (strongly agree) for each statement. The ratings from the 10 statements are then assimilated based on a formula to generate an overall score between 0 and 100, with higher scores indicating better usability. The SCCF received the best SUS scores (Average = 85.8, Median = 86, Min = 65, Max = 100) followed by the TBL condition (Average = 68.7, Median = 64, Min = 38, Max = 100). The SM condition received the least SUS scores (Average = 51.3, Median = 53, Min = 8, Max = 69). The differences in SUS scores between the study conditions were statistically significant, according to a Friedman test[5] ($\chi^2 = 57.1$, $p < 0.001$).

In the exit interviews, almost all participants attributed their high usability ratings for the SSCF condition to the *selection criteria* aspect of GraphLite that they believed saved them a lot of panning effort which otherwise would have been necessary to navigate and compare data points in charts. The participants also stated that the configuration options were extremely important, which explains the relatively higher scores for the SCC condition compared to the baseline SM condition (post-hoc Conover's test, SCC vs. SM: $p = 0.03$). However, we did not notice a significant difference in usability ratings between the SC and SCC conditions (post-hoc Conover's test, SC vs. SCC: $p = 0.06$). This may be due to the fact that space compaction was a background *passive* feature of GraphLite that the users could not explicitly control via the interface; indeed, a few participants (all of whom need less than 5X zoom) mentioned in the exit interviews that they were so focused on customizing the charts that they did not even realize the presence of automatic space compaction. The highest variance observed in the TBL condition can be attributed to the lack of a uniform consensus among participants: while many found the TBL condition less stressful and easier to navigate, others experienced difficulty in analyzing trends under the same condition. This divergence in experiences contributed to the notable variability in the data gathered from the TBL condition.

We administered the standard NASA-TLX questionnaire [26] to asses the perceived workload while the participants did the tasks. NASA-TLX also generates a score between 0 and 100; however, unlike SUS, lower TLX values indicate lesser workloads and, therefore, better performance. Overall, we observed a significant effect of the study conditions on the NASA-TLX scores (Friedman test, $\chi^2 = 92.32$, $p < 0.001$). The GraphLite's SCCF condition received the lowest (i.e., best) TLX scores (Average = 22.7, Median = 22, Min = 14, Max = 32) condition, whereas the SM condition received the highest (i.e., worst) TLX scores (Average = 70.1, Median = 70, Min = 57, Max = 79). The TLX scores for the TBL condition were the second best (Average = 31.7, Median = 31, Min = 24, Max = 45), presumably due to the reduced horizontal panning effort needed to complete the tasks compared to that needed in the SM, SC, and SCC conditions.

A closer inspection of the ratings the participants gave to individual subscales (*Mental Demand*, *Physical Demand*, *Temporal Demand*, *Performance*, *Effort*, *Frustration*) of the TLX questionnaire revealed that poor ratings to the *Effort*, *Mental Demand*, and *Frustration* subscales contributed the most towards the higher workload scores in the SM, Sc, and SCC conditions. For the SCCF condition, the ratings for all the sub-scales were the lowest and, moreover, uniform across all the subscales. For the TBL condition, the ratings for the *Effort* and *Frustration* were considerably higher than those for the SCCF condition, which explains the significant difference in overall TLX scores between SCCF and TBL conditions.

[5]We used this test because the data for some of the conditions were not normally distributed.

5.5.4 Qualitative Feedback

Analysis of subjective feedback from participants in exit interviews revealed shared insights and observations across multiple participants. A few notable ones are presented next.

Customization options are essential. All participants explicitly stated that customization options are essential for conveniently interacting with graphs. More than half (16) of the participants mentioned that they preferred dark themes while interacting with content and that they often experienced issues while interacting with images, including charts under dark themes. These participants further explained that re-rendering charts in dark themes were often not ideal, increasing comprehension difficulty. To counter this issue, 6 of these 16 participants also mentioned that they relied on OS-level color invert accessibility features as an alternative to dark themes. However, they mentioned this option too often, resulting in chart renderings that were not legible and also uncomfortable to view. These participants stated that the fine-grained customization offered by GraphLite quickly addressed this issue and reduced eye strain.

Need to reduce vertical panning. Ten participants expressed a desire for a feature that could reduce the vertical panning/scrolling in GraphLite. Four of these participants suggested automatic readjustment of unit dimensions so that the overall height of a chart is reduced. A couple of participants also suggested adding tooltips that displayed the x/y axis labels/values on demand, e.g., a tap on a bar.

Auto-focus on important chart regions. Nearly a quarter of 6 of the participants suggested providing some auto-panning feature that shifted focus to different parts of a chart on demand, e.g., with a gesture. For instance, P8 wanted gestures that automatically bring the x or y axis to the forefront of the magnifier viewport. Similarly, P17 wished that GraphLite had a gesture that would automatically move the magnifier focus to the top of the current bar in focus.

Filtering in tabular chart representations. More than half (15) of the participants indicated that the usability of the TBL condition (i.e., a tabular representation of chart data) could have been significantly improved by providing a similar filtering feature as that in the GraphLite SCCF condition. These participants further explained that filtering would make tables a more palatable interface due to reduced vertical panning. However, a few (4) participants did mention that tables were not a suitable representation to perceive "trends" in data quickly; these participants explained that their cognitive effort would significantly increase with the increase in the number of data points to be compared for capturing trends.

Following participant feedback, we integrated a tooltip feature into GraphLite. We then evaluated this new feature, focusing on its effectiveness across bar and line charts. In typical screen magnification (SM) + tooltip scenarios, we noted that low-vision users often had to randomly select bars in bar charts and points in line charts to understand their corresponding values. This method proved challenging, especially with charts containing more than 10 data points, as participants tended to forget which bars or points they had already checked, leading to frequent re-use of the tooltip feature. However, integrating the tooltip feature with GraphLite greatly enhanced the efficiency of this process. Users could proactively choose the x-axis values they wanted to focus on for comparison or analysis. Employing tooltips in conjunction with GraphLite, particularly after data filtration, significantly aided users in maintaining awareness of the data points on the charts. This combination of tools effectively streamlined the data interpretation process for low-vision users, making it easier for them to interact with and understand complex data visualizations.

6 Discussion

The user study findings demonstrated that converting non-interactive charts into interactive customizable charts significantly improves usability and graphical comprehension for low vision screen magnifier users on smartphones. However, our work was not without limitations and the study also illuminated some of the shortcomings of our approach, thereby exposing the avenues for future research. A few notable limitations and future work directions are discussed next.

Limitations. GraphLite is dependent on ChartOCR [52] trained on the comprehensive ChartEx dataset [11]. This dataset encompasses a wide range of bar (e.g., grouped, stacked, normalized, categorical vs. ordinal dimensions, horizontal vs. vertical, sorted vs. unsorted) and line chart variants (e.g., single series vs. multiple series values in line segments), which can be seamlessly supported by GraphLite with minimal modifications to its proxy interface. However, GraphLite currently supports only simple bar and line charts, and our evaluation was restricted to these types, where data extraction was accurate to both avoid confounds and make the tasks comparable. However, data extraction is not always accurate, especially if the charts are complex or the quality of chart images is not good. Therefore, the performance of GraphLite must be evaluated "in the wild" on arbitrary charts where the data extraction is not 100% accurate. For this purpose, a separate user study is needed to assess GraphLite usability. Furthermore, given the tremendous advancements in computer vision and image processing techniques in recent years, we anticipate better chart data extraction algorithms will be available in the next few years. Given the modular implementation of GraphLite, we can easily replace the current extraction algorithm with these new, improved algorithms, thereby enhancing the reliability of GraphLite.

Moreover, GraphLite transforms static charts into interactive charts without considering the underlying label annotations present in the static chart, which may reduce the viewer's ability to grasp certain intended takeaways. To address this, we integrated the tooltip feature within GraphLite based on qualitative feedback, allowing users to see the value of data points on the charts. However, GraphLite does not adapt to other forms of annotations, for example, color annotations, i.e., if an author uses specific colors for certain bars in a bar chart to convey a particular message, GraphLite does not retain those specific colors when the chart is re-rendered, leading to a potential loss of information that the chart designer wants to convey.

Chart designers, particularly those creating visualizations for dissemination on news articles or social media, could possibly face challenges when adapting their static charts for low-vision users through GraphLite's transformations. These adaptations can be likened to the responsive design adjustments made when transitioning charts from desktop to mobile viewing environments, as discussed by Kim et al. [39]. With GraphLite's features, such as the integration of dynamic theme pickers, tooltip enhancements, and selectively view data points, designers are equipped to tailor visualizations for improved chart usability. However, these transformations require subjectively prioritizing select data elements while potentially altering and completely missing others. The challenge lies in maintaining the core message and ensuring that key insights remain clear and interpretable for low-vision users.

To address this, we plan to develop a novel algorithm capable of not only extracting data points but also identifying underlying relationships between data points based on the design choices made by the designer. This approach will highlight key elements and their relationships in charts and then present them to low vision users. This approach seeks to balance the trade-offs between preserving information richness and enhancing accessibility for low-vision users.

Also, in the study, the participants used the experimenter-provided smartphone to do the tasks, not their smartphones. While we did our best to accommodate iOS interaction gestures in our evaluation app and conducted an extended practice session, it still needs to replicate the participants' comfort while using their phones. Porting GraphLite to the iOS platform is more of an engineering effort, and we plan to accomplish this in the near future. Subsequent user studies will be conducted to further validate our findings by letting participants do the study tasks using their smartphones.

Lastly, in our work, we did not fully uncover the exact effects of zooming and panning user actions on the perceptual effort and chart comprehension of low-vision users. Uncovering these correlations will require a more extensive study, similar to those in prior works on graph perception for sighted people [14, 27, 74], covering various chart-interaction scenarios. In future work, we plan to conduct this study and build computational interaction models quantifying the chart perceptual effort of low-vision users.

Enhancing data comprehension with Details-on-Demand interactions. Details-on-demand (DoD) interactions in data visualization systems allow users to access additional information about specific elements as needed, keeping the main display uncluttered. This is typically achieved through user actions like hovering, clicking, or performing gestures to display detailed data. Current DoD approaches include selection-based methods [67, 71], where users select visual objects to retrieve details, and zoom-based methods [10], where visualizations transform to reveal more information. Future research could explore how these interactions reduce the need for panning or auto-focus. We envision a controlled study that simulates all existing DoD features to examine the function of each feature individually, helping to better understand the impact of each on visualization comprehension for low-vision users.

Predictive magnification for chart interaction. Prior work [6, 7, 57] has shown that automating screen magnifier lens movement to salient portions of the content can significantly improve usability for low-vision users. Some of the participants in the user study also expressed a desire for "automatic panning" while interacting with charts to reduce the number of input gestures. One method to devise an auto-panning algorithm for low-vision chart interaction is by understanding the corresponding data saliency (e.g., [6]), i.e., the portions of the chart that low-vision users typically assign higher priority and attention compared to other portions. For instance, the units and labels on axes and the top of bars in a bar chart have a higher saliency compared to the empty whitespace regions of the chart. The full saliency heat map can be obtained from gaze tracking studies [19], and this saliency can be used to automate panning between different portions of a data chart, e.g., automatically pan to the top of a given bar in a bar chart based on eye movement. Devising such auto-panning algorithms along with corresponding user interfaces can significantly improve low-vision interaction with data charts.

7 Conclusion

With the increasing use of smartphones among people with low vision, there is a need to address the limitations of conventional screen magnifier accessibility features: loss of context and slow navigation time. This paper addressed these limitations for a specific case scenario of interaction with data charts, specifically the popular bar and line charts. An interview study with 14 low-vision smartphone users revealed that they frequently encountered data charts in their daily smartphone browsing activities, but they also stated that the interaction with charts was challenging due to the magnification-induced loss of visual relationships, e.g., between different data points, and between data points and chart annotations. As an initial effort towards improving low-vision data chart interaction smartphones, we designed and developed GraphLite - a manifestation of the idea of automatically transforming data charts that are usually non-interactive images into customizable interactive graphs that enable users to selectively view multiple data points close to each other, thereby preserving visual context as much as possible under screen enlargement. In a user study with 26 low vision participants, the GraphLite was found to significantly improve the usability of data charts over both the status quo screen magnifier and a state-of-the-art method while doing typical chart interaction tasks. The subjective feedback from the participants also provided future directions for further improving GraphLite, e.g., predictive auto-panning.

Supplemental Materials

All supplementary materials are available on GitHub at `https://github.com/accessodu/GraphLite.git`. These include: (1) Excel files with data from the user study, (2) Details on participant demographics, task assignments, and charts used in the study, (3) Demonstration videos related to the study, and (4) The GraphLite codebase.

Acknowledgment

We would like to express our sincere gratitude to Amanda Kelly Dcosta and Aditya Vishal for their invaluable assistance in system development and user study setup. We also thank the anonymous reviewers for their invaluable feedback that helped improve the paper.

References

[1] R. Alcaraz Martínez, M. Ribera, and T. Granollers Saltiveri. A summary of the article: Methodology for heuristic evaluation of the accessibility of statistical charts for people with low vision and color vision deficiency. In *Comunicació a: Congreso Internacional de Interacción Persona-Ordenador (21º: 2021: Málaga). Interacción 2020/2021. 2021.*, 2021. doi: 10.1007/s10209-021-00816-0 4

[2] S. Ali, L. Muralidharan, F. Alfieri, M. Agrawal, and J. Jorgensen. Sonify: making visual graphs accessible. In *International Conference on Human Interaction and Emerging Technologies*, pp. 454–459. Springer, 2019. doi: 10.1007/978-3-030-25629-6_70 2

[3] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pp. 111–117. IEEE, 2005. doi: 10.1109/INFVIS.2005.1532136 5

[4] Amazon Web Services. Amazon rekognition: Image recognition software, ml image & video analysis. https://aws.amazon.com/rekognition/, 2024. 4

[5] Apple Developer Documentation. Handling long-press gestures. https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures/handling_long-press_gestures, 2024. 4

[6] A. S. Aydin, S. Feiz, V. Ashok, and I. Ramakrishnan. Towards making videos accessible for low vision screen magnifier users. In *Proceedings of the 25th international conference on intelligent user interfaces*, pp. 10–21, 2020. doi: 10.1145/3377325.3377494 9

[7] S. M. Billah, V. Ashok, D. E. Porter, and I. Ramakrishnan. Steeringwheel: a locality-preserving magnification interface for low vision web browsing. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pp. 1–13, 2018. doi: 10.1145/3173574.3173594 2, 9

[8] J. V. Bradley. Complete counterbalancing of immediate sequential effects in a latin square design. *Journal of the American Statistical Association*, 53(282):525–528, 1958. doi: 10.1080/01621459.1958.10501456 5

[9] J. Brooke. Sus: a "quick and dirty'usability. *Usability evaluation in industry*, 189(3), 1996. https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale. 8

[10] T. Büring, J. Gerken, and H. Reiterer. User interaction with scatterplots on small screens-a comparative evaluation of geometric-semantic zoom and fisheye distortion. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):829–836, 2006. doi: 10.1109/TVCG.2006.187 9

[11] ChartOCR. Deepruledataset. https://huggingface.co/datasets/asbljy/DeepRuleDataset/tree/main, 2023. 9

[12] 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*, 2023. doi: 10.1109/TVCG.2023.3327354 2

[13] J. Choi, S. Jung, D. G. Park, J. Choo, and N. Elmqvist. Visualizing for the non-visual: Enabling the visually impaired to use visualization. In *Computer Graphics Forum*, vol. 38, pp. 249–260. Wiley Online Library, 2019. doi: 10.1111/cgf.13686 2, 3, 5

[14] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554, 1984. doi: 10.1080/01621459.1984.10478080 9

[15] W. W. W. Consortium et al. Accessibility requirements for people with low vision. https://www.w3.org/TR/low-vision-needs/, 2016. 2

[16] M. D. Crossland, R. S. Silva, and A. F. Macedo. Smartphone, tablet computer and e-reader use by people with vision impairment. *Ophthalmic and Physiological Optics*, 34(5):552–557, 2014. doi: 10.1111/opo.12136 2

[17] Dart Team. Dart programming language. https://dart.dev/, 2024. 5

[18] 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 3

[19] S. Elzer, N. Green, S. Carberry, and J. Hoffman. A model of perceptual task effort for bar charts and its role in recognizing intention. *User Modeling and User-Adapted Interaction*, 16(1):1–30, 2006. doi: 10.1007/s11257-006-9002-9 3, 9

[20] C. Engel, E. F. Müller, and G. Weber. Svgplott: an accessible tool to generate highly adaptable, accessible audio-tactile charts for and from blind and visually impaired people. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pp. 186–195, 2019. doi: 10.1145/3316782.3316793 2

[21] C. Engel and G. Weber. A user study to evaluate tactile charts with blind and visually impaired people. In *International Conference on Computers Helping People with Special Needs*, pp. 177–184. Springer, 2018. doi: 10.1007/978-3-319-94274-2_24 2

[22] Flutter. Write your first flutter app. https://docs.flutter.dev/get-started/codelab, 2024. 5

[23] J. Fu, B. Zhu, W. Cui, S. Ge, Y. Wang, H. Zhang, H. Huang, Y. Tang, D. Zhang, and X. Ma. Chartem: reviving chart images with data embedding. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):337–346, 2020. doi: 10.1109/TVCG.2020.3030351 2

[24] C. Goncu, K. Marriott, and J. Hurst. Usability of accessible bar charts. In *International Conference on Theory and Application of Diagrams*, pp. 167–181. Springer, 2010. doi: 10.1007/978-3-642-14600-8_17 2

[25] Google Cloud Team. Advanced guide to inception v3 on cloud tpu. https://cloud.google.com/tpu/docs/inception-v3-advanced, 2024. 5

[26] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, vol. 52, pp. 139–183. Elsevier, 1988. doi: 10.1016/S0166-4115(08)62386-9 8

[27] J. Heer and M. Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 203–212, 2010. doi: 10.1145/1753326.1753357 9

[28] J. Hoffswell, W. Li, and Z. Liu. Techniques for flexible responsive visualization design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2020. doi: 10.1145/3313831.3376777 3

[29] M. N. Hoque, M. Ehtesham-Ul-Haque, N. Elmqvist, and S. M. Billah. Accessible data representation with natural sound. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–19, 2023. doi: 10.1145/3544548.3581087 2

[30] J. In and S. Lee. Statistical data presentation. *Korean journal of anesthesiology*, 70(3):267–276, 2017. doi: 10.4097/kjae.2017.70.3.267 1

[31] M. T. Islam and S. M. Billah. Spacex mag: An automatic, scalable, and rapid space compactor for optimizing smartphone app interfaces for low-vision users. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 7(2):1–36, 2023. doi: 10.1145/3596253 2

[32] M. Jain, N. Diwakar, and M. Swaminathan. Smartphone usage by expert blind users. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2021. doi: 10.1145/3411764.3445074 2

[33] B. E. John and A. Newell. Toward an engineering model of stimulus-response compatibility. In *Advances in psychology*, vol. 65, pp. 427–479. Elsevier, 1990. doi: 10.1016/S0166-4115(08)61233-9 3

[34] I. A. Joshi. Unblind the charts: Towards making interactive charts accessible in android applications. *arXiv preprint arXiv:2109.12442*, 2021. doi: 10.48550/arXiv.2109.12442 2

[35] 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*, pp. 6706–6717, 2017. doi: 10.1145/3025453.3025957 2

[36] K. Kafle, B. Price, S. Cohen, and C. Kanan. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2018. doi: 10.48550/arXiv.1801.08163 2

[37] S. Kanthara, R. T. K. Leong, X. Lin, A. Masry, M. Thakkar, E. Hoque, and S. Joty. Chart-to-text: A large-scale benchmark for chart summarization. *arXiv preprint arXiv:2203.06486*, 2022. doi: 10.48550/arXiv.2203.06486 2

[38] A. Khan and S. Khusro. Blind-friendly user interfaces–a pilot study on improving the accessibility of touchscreen interfaces. *Multimedia Tools and Applications*, 78(13):17495–17519, 2019. doi: 10.1007/s11042-018-7094-y 2

[39] H. Kim, D. Moritz, and J. Hullman. Design patterns and trade-offs in responsive visualization for communication. In *Computer Graphics Forum*, vol. 40, pp. 459–470. Wiley Online Library, 2021. doi: 10.1111/cgf.14321 3, 9

[40] H. Kim, R. Rossi, J. Hullman, and J. Hoffswell. Dupo: A mixed-initiative authoring tool for responsive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2023. doi: 10.48550/arXiv.2308.05136 3

[41] H. Kim, R. Rossi, A. Sarma, D. Moritz, and J. Hullman. An automated approach to reasoning about task-oriented insights in responsive visu-

alization. *IEEE transactions on visualization and computer graphics*, 28(1):129–139, 2021. doi: 10.48550/arXiv.2107.08141 3

[42] J. Kim, A. Srinivasan, N. W. Kim, and Y.-S. Kim. Exploring chart question answering for blind and low vision users. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, article no. 828, 15 pages. Association for Computing Machinery, New York, NY, USA, 2023. doi: 10.1145/3544548.3581532 1

[43] Y.-H. Kim, B. Lee, A. Srinivasan, and E. K. Choe. Data@ hand: Fostering visual exploration of personal data on smartphones leveraging speech and touch interaction. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–17, 2021. doi: 10.1145/3411764.3445421 3

[44] N. Lavie, Z. Lin, N. Zokaei, and V. Thoma. The role of perceptual load in object recognition. *Journal of Experimental Psychology: Human Perception and Performance*, 35(5):1346, 2009. doi: 10.1037/a0016454 4

[45] N. Lavie and Y. Tsal. Perceptual load as a major determinant of the locus of selection in visual attention. *Perception & psychophysics*, 56:183–197, 1994. doi: 10.3758/BF03213897 4

[46] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 734–750, 2018. doi: 10.48550/arXiv.1808.01244 4

[47] H.-N. Lee, Y. Prakash, M. Sunkara, I. Ramakrishnan, and V. Ashok. Enabling convenient online collaborative writing for low vision screen magnifier users. In *Proceedings of the 33rd ACM Conference on Hypertext and Social Media*, pp. 143–153, 2022. doi: 10.1145/3511095.3531274 2

[48] X. Liu, D. Klabjan, and P. NBless. Data extraction from charts via single deep neural network. *arXiv preprint arXiv:1906.11906*, 2019. doi: 10.48550/arXiv.1906.11906 2, 3

[49] G. L. Lohse. A cognitive model for understanding graphical perception. *Human-Computer Interaction*, 8(4):353–388, 1993. doi: 10.1207/s15327051hci0804_3 3

[50] Lorenzo Pichilli. Flutter inappwebview. `https://pub.dev/packages/flutter_inappwebview`, 2024. 5

[51] S. J. Luck and M. A. Ford. On the role of selective attention in visual perception. *Proceedings of the National Academy of Sciences*, 95(3):825–830, 1998. doi: 10.1073/pnas.95.3.825 3

[52] J. Luo, Z. Li, J. Wang, and C.-Y. Lin. Chartocr: data extraction from charts images via a deep hybrid framework. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 1917–1925, 2021. doi: 10.1109/WACV48630.2021.00196 2, 4, 9

[53] R. A. Martínez, M. R. Turró, and T. G. Saltiveri. Accessible statistical charts for people with low vision and colour vision deficiency. In *Proceedings of the XX International Conference on Human Computer Interaction*, pp. 1–2, 2019. doi: 10.1145/3335595.3335618 1

[54] A. Masry, D. X. Long, J. Q. Tan, S. Joty, and E. Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022. doi: 10.18653/v1/2022.findings-acl.177 2

[55] 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*, pp. 1–17, 2023. doi: 10.1145/3544548.3581113 2

[56] G. G. Méndez, M. A. Nacenta, and S. Vandenheste. ivolver: Interactive visual language for visualization extraction and reconstruction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 4073–4085, 2016. doi: 10.1145/2858036.2858435 2

[57] F. Momotaz and S. M. Billah. Tilt-explore: Making tilt gestures usable for low-vision smartphone users. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pp. 1154–1168, 2021. doi: 10.1145/3472749.3474813 2, 9

[58] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pp. 483–499. Springer, 2016. doi: 10.48550/arXiv.1603.06937 4

[59] J. Poco and J. Heer. Reverse-engineering visualizations: Recovering visual encodings from chart images. In *Computer graphics forum*, vol. 36, pp. 353–363. Wiley Online Library, 2017. doi: 10.1111/cgf.13193 2

[60] Python Basics. Flask rest api tutorial. `https://pythonbasics.org/flask-rest-api/`, 2021. 5

[61] L. Richardson. Beautiful soup documentation. `https://beautiful-soup-4.readthedocs.io/en/latest/`, 2015. 5

[62] J. L. Robinson, V. Braimah Avery, R. Chun, G. Pusateri, and W. M. Jay. Usage of accessibility options for the iphone and ipad in a visually impaired population. In *Seminars in Ophthalmology*, vol. 32, pp. 163–171. Taylor & Francis, 2017. doi: 10.3109/08820538.2015.1045151 2

[63] S. F. Roth and J. Mattis. Data characterization for intelligent graphics presentation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 193–200, 1990. doi: 10.1145/97243.97273 5

[64] E. Russell-Minda, J. W. Jutai, J. G. Strong, K. A. Campbell, D. Gold, L. Pretty, and L. Wilmot. The legibility of typefaces for readers with low vision: A research review. *Journal of Visual Impairment & Blindness*, 101(7):402–415, 2007. doi: 10.1177/0145482X0710100703 4

[65] J. Saldaña. Coding techniques for quantitative and mixed data. `https://www.torrossa.com/en/resources/an/5018667`, 2021. 3, 6

[66] 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, 2011. doi: 10.1145/2047196.2047247 2

[67] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *IEEE transactions on visualization and computer graphics*, 16(6):1139–1148, 2010. doi: 10.1109/TVCG.2010.179 9

[68] A. Sharif, S. S. Chintalapati, J. O. Wobbrock, and K. Reinecke. Understanding screen-reader users' experiences with online data visualizations. In *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '21, article no. 14, 16 pages. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3441852.3471202 1

[69] soap117. Deeprule. `https://github.com/soap117/DeepRule`, 2023. 5

[70] A. Srinivasan, B. Lee, N. Henry Riche, S. M. Drucker, and K. Hinckley. Inchorus: Designing consistent multimodal interactions for data visualization on tablet devices. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pp. 1–13, 2020. doi: 10.1145/3313831.3376782 3

[71] H. Subramonyam and E. Adar. Smartcues: a multitouch query approach for details-on-demand through dynamically computed overlays. *IEEE transactions on visualization and computer graphics*, 25(1):597–607, 2018. doi: 10.1109/TVCG.2018.2865231 9

[72] Syncfusion. Syncfusion flutter charts. `https://pub.dev/packages/syncfusion_flutter_charts`, 2024. 5

[73] S. F. A. Szpiro, S. Hashash, Y. Zhao, and S. Azenkot. How people with low vision access computing devices: Understanding challenges and opportunities. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*, pp. 171–180, 2016. doi: 10.1145/2982142.2982168 2

[74] J. Talbot, V. Setlur, and A. Anand. Four experiments on the perception of bar charts. *IEEE transactions on visualization and computer graphics*, 20(12):2152–2160, 2014. doi: 10.1109/TVCG.2014.2346320 3, 5, 9

[75] Think-Cell. Think-cell: Powerpoint charts | waterfall, gantt, mekko, process flow and agenda. `https://www.think-cell.com/en`, 2024. 3

[76] R. Wang, L. Zeng, X. Zhang, S. Mondal, and Y. Zhao. Understanding how low vision people read using eye tracking. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–17, 2023. doi: 10.1145/3544548.3581213 3

[77] A. Wu, W. Tong, T. Dwyer, B. Lee, P. Isenberg, and H. Qu. Mobilevisfixer: Tailoring web visualizations for mobile phones leveraging an explainable reinforcement learning framework. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):464–474, 2020. doi: 10.48550/arXiv.2008.06678 3

[78] L. H. Wurm, G. E. Legge, L. M. Isenberg, and A. Luebker. Color improves object recognition in normal and low vision. *Journal of Experimental Psychology: Human perception and performance*, 19(4):899, 1993. doi: 10.1037//0096-1523.19.4.899 4

[79] X. Xia, C. Xu, and B. Nan. Inception-v3 for flower classification. In *2017 2nd international conference on image, vision and computing (ICIVC)*, pp. 783–787. IEEE, 2017. doi: 10.1109/ICIVC.2017.7984661 4

[80] Z. Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*, pp. 1–2. Ieee, 2018. doi: 10.1109/IWQoS.2018.8624183 4

[81] H. Zou and J. Treviranus. Chartmaster: A tool for interacting with stock market charts using a screen reader. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, ASSETS '15, 10 pages, p. 107–116. Association for Computing Machinery, New York, NY, USA, 2015. doi: 10.1145/2700648.2809862 1