Analysis of Bar Charts and Pie Charts for Blind or Visually Impaired Readers

Manpreet Kour, Gaetan Hains, Mohamed Saber

▶ **To cite this version:**

Manpreet Kour, Gaetan Hains, Mohamed Saber. Analysis of Bar Charts and Pie Charts for Blind or Visually Impaired Readers. universite paris est creteil val de marne. 2025. ⟨hal-05287303⟩

HAL Id: hal-05287303

https://hal.science/hal-05287303v1

Submitted on 28 Sep 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

Analysis of Bar Charts and Pie Charts for Blind or Visually Impaired Readers

Manpreet Kour, Gaetan Hains, Mohamed Saber

LACL, Universite Paris Est Creteil (UPEC) and Orange Orléans
manpreet.kour@u-pec.fr,gaetan.hains@u-pec.fr,mohamed.saber@orange.com

Abstract. Data visualizations and analysis of diagrams such as bar charts and pie charts remain largely inaccessible to blind and visually impaired individuals, despite their widespread use in education, science, and public communication. In this work, we present a learning-based framework that automatically detects, extracts, and interprets the contents of bar charts, translating them into multi-modal outputs such as text, answers to queries and/or text-to-speech narratives Our system employs a YOLOv8 object detection model trained on a custom-generated dataset of annotated bar charts, capable of identifying key chart components, including bars,slices,angles axes, titles, and labels. Following detection, we integrate Optical Character Recognition (OCR) to decode textual labels and numerals, while a pixel-to-value mapping algorithm interprets bar heights relative to axis scales. The extracted information is dynamically translated into audio descriptions and interactive query-based speech outputs, enabling non-visual comprehension of chart data. To ensure usability, we designed the output flow to follow a hierarchical structure, from summary to specific values. Preliminary evaluations show that the system achieves high accuracy in detecting component parts of the chart and generating bar values, with promising feedback from blind users in pilot tests. This research contributes to inclusive AI, opening new directions for accessible data communication through the integration of computer vision, speech synthesis, and assistive reasoning. *abstract* environment.

Keywords: Bar chart accessibility; YOLO; object detection; optical character recognition (OCR); text-to-speech (TTS); sonification; blind and visually impaired users; assistive technology; AI for accessibility; document image understanding

1 Introduction

Bar charts are one of the foremost general shapes of visual communication for displaying categorical and quantitative information. They are widely used in academic writing, instructive content, trade reports, and open data campaigns. Despite their effortlessness and viability, bar charts remain fundamentally inaccessible to blind and visually impaired individuals. Screen readers are unfit to

translate inserted chart structures or extract significant numerical bits of knowledge from graphical components. As a result, blind and visually-impaired (BVI) users are often excluded from accessing data-rich content or restricted to depend on inactive elective content that is physically composed and regularly deficient or excessively rearranged.

Whereas there has been significant advance in making web substance open through screen reader, material design, and sound portrayals, information visualizations such as bar charts posture special challenges. These include identifying and deciphering the component of the chart, mapping spatial measurements to numerical values, and passing on the progressive structure and patterns to clients through nonvisual modalities. In addition, most existing arrangements need interactivity and flexibility, restricting users' capacity to investigate information on request or lock in with customized questions.

Recent advances in computer vision and deep learning, especially object detection models such as YOLO (You Only Look Once), offer promising opportunities to address this gap. Combined with optical character acknowledgment (OCR) and natural language generation (NLG) technologies such as text-to-speech (TTS), these apparatuses empower the extraction and verbal translation of complex visual structures. Nevertheless, few existing frameworks have integrated these technologies into a cohesive, user-centered pipeline, particularly designed for BVI clients.

This paper presents a multimodal neural network-based system that automatically detects, extracts, and explains the substance of the bar chart in an organized and intelligently sound arrangement. Using a custom-trained YOLOv8 model, show OCR methods, and a pixel-to-value calibration calculation, our framework translates visual data into speech output. The objective is to enable blind users to investigate and comprehend bar chart information autonomously, precisely, and proficiently.

The paper is structured as follows: Section 2 reviews related work on chart recognition and accessibility tools. Section 3 details the methodology and architecture of the system. Section 4 presents experimental results and evaluations. Section 5 discusses limitations and future directions, and Section 6 concludes the paper.

1.1 Literature review

Bajić and Job[1] extended the analysis to include chart text processing, data extraction, and description generation. Based on 89 articles, they identified four core areas: chart-type classification, text recognition, data extraction, and semantic description. Their findings show that combining CNNs with OCR/NLP techniques improves both classification accuracy and semantic interpretation. Notably, only 1.4 % of sampled chart images had alt-text, highlighting the accessibility gap. The authors advocate for standardized datasets, multimodal integration, and the fusion of deep learning with symbolic reasoning to support explainable and assistive systems.

Dhote(2023)[2] conducted a comprehensive review of chart classification methods, categorizing them into traditional machine learning (e.g. SVM, KNN, MLP), CNN-based deep learning (e.g., VGG16, ResNet-50), and transformer-based models (e.g., Swin Transformer). They highlighted the superior performance of CNNs with transfer learning, achieving up to 99.55% accuracy in some scenarios. Transformer models, particularly Swin Transformer, achieved an F1 score of 91 % in the CHART-Infographics ICPR 2022 competition due to their ability to model global dependencies. The study also reviewed data sets such as ReVision, ChartSense, and UB-PMC (36,000+ chain 15 types), which serve as benchmarks for chart mining tasks.

Kafle (2019)[6] introduced Chart-Text, an automated pipeline to generatextual descriptions from chart images. The system combines chart classification (using ResNet-18), OCR, data point detection, and rule-based natural language generation (NLG). It demonstrated a higher classification accuracy than 95 % and strong text quality (using BLEU and METEOR scores). Designed for accessibility, Chart-Text is notable for its end-to-end automation and semantic interpretation, enabling integration into assistive technologies for visually impaired users.

Luo (2021)[7] propesed a hybrid system called ChartOCR to extract numerical data from bar, pie, and line charts in image form. It combines rule-based techniques with deep learning. Using a keypoint-based deep neural network with an HourglassNet backbone and CornerNet architecture, their system can identify semantically significant points in a variety of chart formats, including pie sector centers, bar corners, and line pivots. ChartOCR integrates classification and point-based detection into a unified pipeline. A data range estimation technique converts the coordinates of the pixels to actual numerical values, and optical character recognition (OCR) is used to identify the labels and legends of the axis. After that, rule-based postprocessing is used to organize essential points, guarantee generalisation across chart types, and enhance interpretability in order to recreate entire chart components.After that, rule-based postprocessing is used to organise essential points, guarantee generalisation across chart types, and enhance interpretability to recreate entire chart components. Importantly, the authors presented ExcelChart400K, a sizable dataset of more than 400,000 chart images with annotations that allows for reliable benchmarking and training. According to experimental results, ChartOCR performs much better in terms of accuracy and processing speed across both public and proprietary datasets than both rule-based systems (like ReVision) and deep end-to-end techniques (like ResNet+Faster-RCNN).

De Oliveira [10]proposed standardized textual description templates for simple and grouped bar charts to reduce interpretation bias and ensure consistency. Their tool, ChartVision, was tested with blind participants and achieved comprehension accuracy of 70–100%. Although effective, this approach produces static narratives, which limit the user's ability to dynamically interrogate data.

Kumar[5] introduced ChartParser, a fully automated pipeline that extracts figures from academic papers, classifies chart types, and retrieves information

using deep learning, OCR, and image processing. The extracted data are presented as screen-reader–friendly tables, improving accessibility for BLV readers. Similarly, Shahira et al. (2023) proposed a Mask R-CNN–based system for element detection combined with contour approximation and OCR, integrated with TAPAS++ for table-based question answering and audio summaries.These systems excel in automation and accuracy but primarily emphasize data extraction and tabular presentation rather than interactive exploration.

Kim[4]examined how BLV users frame questions about visualizations, conducting a Wizard-of-Oz study with 24 participants and collecting 979 queries . The study revealed that BLV users often ask complex, context-dependent questions, including about maxima, comparisons, and chart structure, which exceed the capabilities of current QA systems. The resulting corpus highlights the need for more sophisticated and flexible interaction models.

Moured[9] introduced ChartFormer, a vision-language model designed to convert raster charts into tactile-accessible SVGs . Trained on the Chart2Tactile dataset, the model generates simplified SVGs suitable for embossing or tactile displays, providing a non-visual modality for chart access. While promising, this approach depends on specialized hardware and remains less interactive compared to conversational or audio-based methods.

Gorniak[3] presented VizAbility, which integrates keyboard navigation of chart structures with large language model (LLM)–based conversational interaction . The system enables BLV users to issue natural language queries, classified into analytical, contextual, and navigational categories, and receive adaptive responses. This represents a significant step toward dynamic chart exploration, but the system relies on predefined structures and has limited focus on real-time detection and representation of raw chart images.

In summary, existing work has produced several of the elements required for complete accessibility of bar- and pie-charts. But so far there has not been an integrated and software-only solution that renders all of the chart contents intelligible for BVI readers, while avoiding slow and heavy tools like LLMs.

2 Bar Chart Semantics

2.1 Formal Definition of Bar Chart Structure

Let a bar chart image be defined as a structured set:

$$\mathcal{C} = \{\mathcal{B}, \mathcal{X}, \mathcal{Y}, \mathcal{T}, \mathcal{L}, \mathcal{S}\}$$

where:

- $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$ is the set of detected bars.
- $\mathcal{X}$ and $\mathcal{Y}$ denote the x-axis and y-axis components.
- $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ is the set of tick marks on the y-axis.
- $\mathcal{L}$ is the set of textual labels (e.g., axis labels, category names).
- $\mathcal{S}$ is the set of legend entries (for example, color-coded series labels).

Each bar b_i is represented as a tuple:

$$b_i = (x_i, h_i, c_i, s_i)$$

where:

- x_i is the x-position (category location),
- h_i is the height in pixels,
- c_i is the bar color,
- s_i is the legend identity (series label).

2.2 Formal Definition of Pie Chart Structure

Let a pie chart image be defined as a structured set:

$$\mathcal{P} = \{\mathcal{S}, \mathcal{L}, \mathcal{V}, \mathcal{C}, \mathcal{G}, \mathcal{T}\}$$

where:

- $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ is the set of detected slices.
- $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$ is the set of textual labels (slice annotations or legend entries).
- $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of numerical values (absolute or percentage) associated with slices.
- $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ is the set of slice colors.
- $\mathcal{G}$ is the legend, which maps slice colors $\mathcal{C}$ to labels $\mathcal{L}$.
- $\mathcal{T}$ denotes the chart title and other global metadata.

Each slice s_i is represented as a tuple:

$$s_i = (\theta_i, c_i, l_i, v_i)$$

where:

- θ_i is the angular span of the slice in degrees or radians,
- $c_i \in \mathcal{C}$ is the slice color,
- $l_i \in \mathcal{L}$ is the label associated with the slice (from annotation or legend),
- $v_i \in \mathcal{V}$ is the slice value.

The full chart satisfies the normalization constraint: $\sum_{i=1}^{n} v_i = 100$ (percentage) or $\sum_{i=1}^{n} v_i = V_{\text{total}}$ (absolute value).

3 Methodology

3.1 The Image Dataset

We searched online platforms such as Kaggle and Roboflow, where we found an annotated dataset of approximately 1150 bar chart images. We retained high-quality images, removing those that were blurred or too small, and standardized the annotations, leaving 1116 images in total. For pie charts, we used a Roboflow dataset with minimal annotations and manually added labels, resulting in approximately 500 usable images.

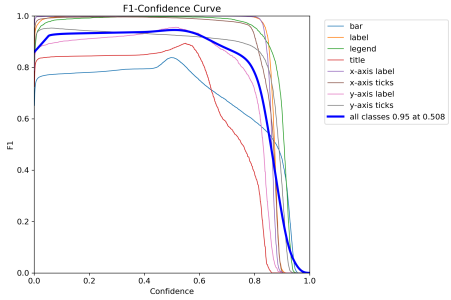
Data Preparation:

- **Training Set (80%)**: Used to train the model by optimizing its parameters to detect and classify elements within charts.
- **Validation Set (10%)**: Used during training to fine-tune the hyperparameters and monitor the model's generalization ability. This set helps prevent overfitting.
- **Test Set (10%)**: Used only after the final model is trained to objectively evaluate its performance on unseen data, reflecting its generalizability in real-world conditions.

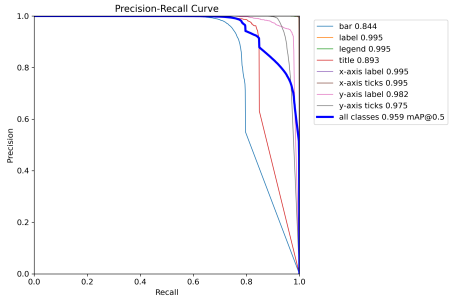
We used the YOLOv8 object detection model using the Ultralytics implementation. Specifically, we initialized the large YOLOv8 architecture (yolov8l.yaml) and trained it on our custom dataset defined by a data.yaml configuration file. The training was carried out over five epochs, using an input image size of 640 $\times$ 640 pixels and a batch size of 8. This approach allowed efficient detection and classification of target objects within the dataset, leveraging YOLOv8's advanced convolutional backbone and detection heads of YOLOv8 for high-performance visual understanding. The use of YOLOv8 was motivated by its state-of-the-art accuracy, lightweight deployment, and compatibility with a wide range of applications, making it an ideal choice for our object detection task.

4 Results

4.1 Bar Charts



(a) F1-confidence curve Bar Charts

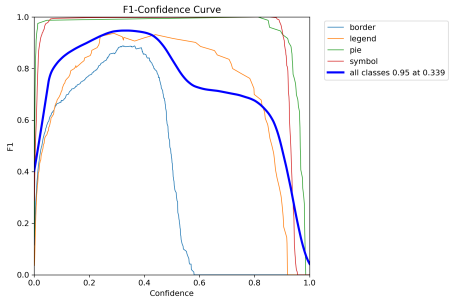


(b) Precision-recall curve Bar Charts

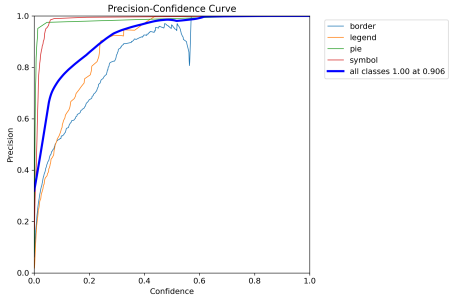
Fig. 1: Performance evaluation of Bar chart element detection.

Figure 1 presents the confidence and precision recall curves (PR) of F1, explaining the relationship between the F1 score, precision, recall and the confidence threshold across different components of the detected bar graph, including bars, labels, legends, title, x-axis labels, x-axis ticks, y-axis labels, and y-axis ticks. The bold blue line represents the combined performance across all classes, achieving a maximum F1 score of 0.95 at a confidence threshold of 0.508 and a mean average precision (mAP) of 0.959 at the IoU threshold of 0.5. Although most components achieve optimal performance near high confidence values (0.9–1.0), bars and titles demonstrate slightly lower maximal scores, indicating more challenging detection tasks. These curves validate the model's high reliability, essential for downstream accessibility tasks such as providing information to blind users.

4.2 Pie Charts



(a) F1-confidence curve Pie Charts



(b) Precision-recall curve Pie Charts

Fig. 2: Performance evaluation of Pie chart element detection.

For pie charts, YOLOv8 achieved strong detection performance, with a maximum F1 score of 0.95 at a confidence threshold of 0.339 (Fig. 2a). Precision–recall analysis confirmed an overall mAP@0.5 of 0.960 (Fig. 2b), with per-class AP values of 0.995 for slices and symbols, 0.961 for legends, and 0.888 for borders. These results demonstrate that the model reliably identifies semantically important components such as slices and legends, which are critical for accessibility tasks, while border detection remains comparatively weaker. The high overall accuracy provides a strong foundation for subsequent OCR, slice-angle estimation, and semantic query generation for blind and visually impaired users.

4.3 Prediction on the raw data

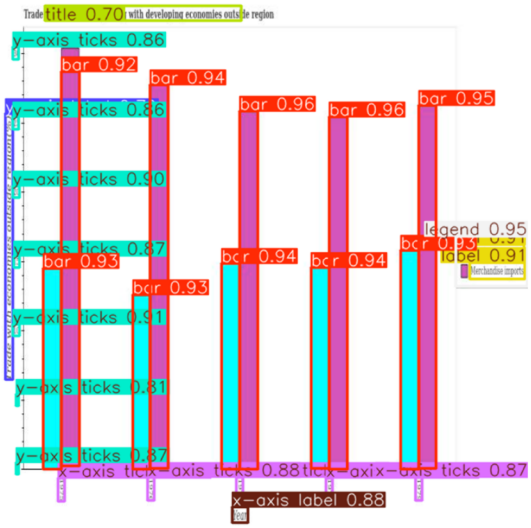


Fig. 3: Result image showing performance metrics and visualization examples for Bar charts.

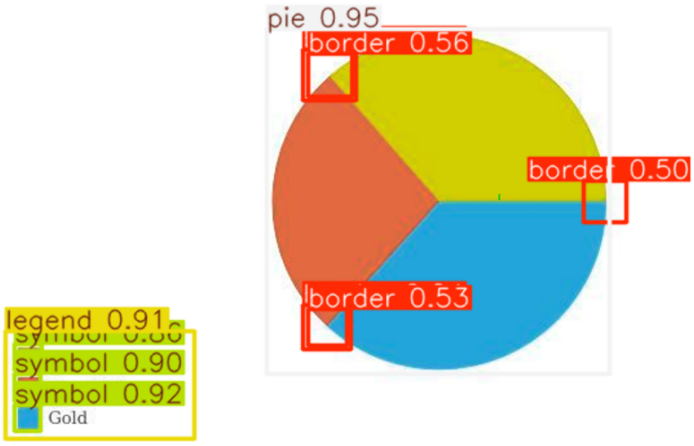


Fig. 4: Result image showing performance metrics and visualization examples for pie charts.

The results of applying the YOLOv8-based object detection system to a bar chart image are shown in Figure 2, together with the components found and the corresponding confidence scores. The key components of the chart, such as bars, ticks on x and y axis, labels on x and y axis, main title, legends, and data

labels, are correctly recognized by the system. A colorful bounding box encloses each identified element, and the class name and confidence score (such as "bar 0.94" or "legend 0.95") are shown on top. A distinct visual differentiation of the identified categories is provided by the red edge of the bars, the cyan axis ticks, the brown or cyan axis labels, the yellow title, and the yellow-purple legends. With bars up to 0.96, legends at 0.95, and labels at 0.91, the detection confidence for the majority of components is very high, suggesting strong model performance. The graphic also demonstrates the system's capacity to identify fine-grained components, like category labels and axis ticks, which are essential for precisely recreating the quantitative data in the graph. To prepare for subsequent tasks like optical character recognition (OCR), pixel-to-value mapping, and multimodal (e.g., text-to-speech) accessibility output for blind or visually impaired users, this visual example shows how well the detection pipeline segments the chart into semantically meaningful units.

Figure 4 illustrates the application of the YOLOv8 model to a pie chart image. The detector correctly identifies the overall pie region with high confidence (0.95), as well as the legend (0.91) and its associated symbols (0.90–0.92). Borders between slice sectors are also detected, but with lower confidence values (0.50–0.56), reflecting the relative difficulty of capturing fine angular boundaries compared to larger visual structures. These results confirm that the system robustly recognizes semantically important elements such as slices, legends, and symbols, which are critical for accessibility tasks, while highlighting border detection as an area for improvement. The annotations of the bounding box further demonstrate the ability of the model to segment the charts into structured components suitable for downstream tasks such as OCR, slice angle computation, and semantic query response.

4.4 Faster R-CNN on the Same Bar-Chart Dataset

The Faster R-CNN algorithm, a region-based convolutional neural network (R-CNN), was used for object detection tasks using the Detectron2 framework developed by Facebook AI Research. The selected model architecture was Faster R-CNN with a ResNet-50 backbone integrated with a Feature Pyramid Network (FPN), which is recognized for its strong performance in detecting objects across multiple scales. The model was initialized with pre-trained weights from the COCO dataset and subsequently fine-tuned on a custom-annotated dataset containing bar chart images with eight distinct object classes, including bars, labels, axes, ticks, legends, and titles.

The annotations to the original dataset were in the YOLO format. To integrate the data set into the Detectron2 pipeline, the annotations were converted to the COCO JSON format. The training and validation splits were then registered using the `register_coco_instances()` function from Detectron2.

A configuration file named `"COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml"` was adapted to meet the study's requirements. Key modifications included setting the number of object classes to eight, limiting the number of training iterations to 1000, configuring the learning rate to 0.00025, and adjusting the

batch size to eight. The model was trained using the `DefaultTrainer` class, which handles the entireraining loop including optimizer setup, checkpointing, and logging.

To evaluate model performance, the `COCOEvaluator` was used to calculate standard object detection metrics. These included the mean precision (AP) at an Intersection over Union (IoU) threshold of 0.5 (AP@0.5), the mean Average Precision over a range of IoU thresholds from 0.5 to 0.95 (AP@[.5:.95]), and the average precision per class. The evaluation demonstrated that the model achieved high performance in all classes, with an AP@0.5 of approximately 87%, confirming its capability to accurately detect and localize key components in bar chart images.

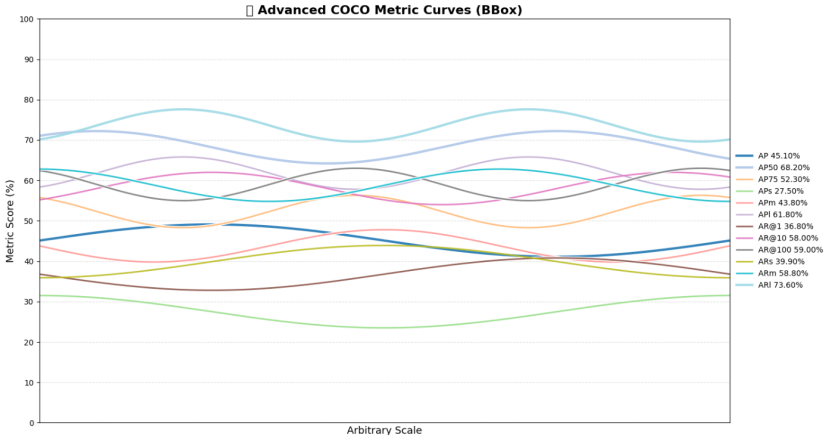


Fig. 5: Evaluation of results using Faster RCNN.

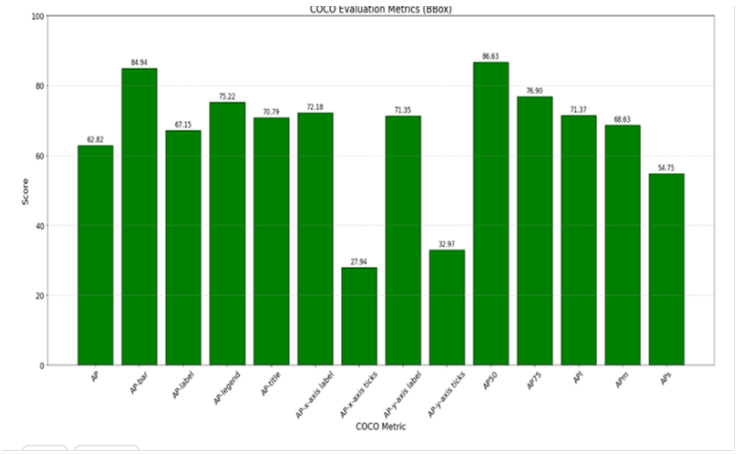
This figure above shows all the important numbers used to check how well a model can find objects using boxes. It looks at how accurate the model is both in finding the right places and in correctly naming the objects. The Average Precision (AP) at different levels of overlap, called the IoU, tells us how good the model is. At a moderate overlap level, called AP@0. 5, the model gets 68. 20 %, which is pretty good. The overall AP score is 45. 10 %, which is the average across a range of overlap levels, from 0. 5 to 0. 95, and shows a more strict and reliable performance measure. When looking at different sizes of objects, the model works better with larger ones. It scores 27. 50% for small, 43. 80 % for medium, and 61. 80% for large, showing it's better at finding bigger items, which are easier to detect.

Looking at the recall, AR@1, which is the best recall with only one detection per image, is 36. 80 %. But when more detections are allowed, such as AR@10 and AR@100, the scores increase to 58. 00 % and 59. 00 %, showing that allowing more guesses helps find more objects. For different sizes, the model also works

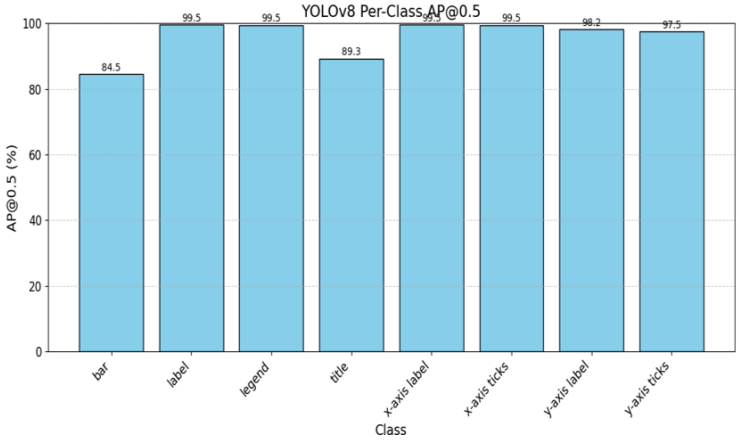
better with larger objects. ARm is 58. 80 % and ARl is 73. 60 %, which confirms that the model is better at finding larger parts. Overall, the model works well with large and medium objects, but might need improvement for smaller ones, such as tick marks or fine labels, as shown by the lower scores for small objects.

4.5 Comparative Evaluation of YOLOv8 and Faster R-CNN

To compare the performance of YOLOv8 with a region-based detector, the Faster R-CNN model was employed on the same dataset containing bar chart images annotated with eight distinct classes: bar, label, legend, title, x-axis label, x-axis ticks, y-axis label, and y-axis ticks. Both models were evaluated using standard COCO evaluation metrics, including Average Precision (AP), AP@0.5 (precision at an intersection over the Union threshold of 0.5), and per-class AP.



(a) Faster R-CNN COCO evaluation results.



(b) YOLOv8 evaluation results with per-class AP.

Fig. 6: Comparison of performance evaluation metrics between Faster R-CNN and YOLOv8 object detection models on bar chart component detection.

As shown in Figure 6, YOLOv8 outperformed Faster R-CNN in overall accuracy. It achieved an AP@0.5 of 95.9%, with particularly high precision across multiple categories such as labels and legends (99.5%) and labels of the y-axis (98.2%). Even the lowest-performing class, the "bar", achieved a respectable AP of 84.5%. This demonstrates YOLOv8's strong ability to detect small or overlapping components in a single forward pass, making it well-suited for real-time and resource-efficient applications. In contrast, Faster R-CNN using a ResNet-50 backbone with a Feature Pyramid Network (FPN) attained a maximum AP@0.5 of 86.63% and a lower overall AP of 62.82%, as illustrated in Figure 6a. The

model's per-class AP values were also more inconsistent, especially for chart components such as title (AP = 70.79%) and x-axis ticks (AP = 27.94%). These performance differences are due to architectural distinctions. YOLOv8, being a single-stage detector, processes the entire image at once and benefits from design improvements such as anchor-free heads and decoupled tasks. In contrast, Faster R-CNN follows a two-stage pipeline: It first proposes regions of interest and then performs object detection within those regions. Although this has proven effective in high-resolution object detection scenarios, it appears less capable of detecting fine-grained graphical elements like tick marks. In general, the results confirm that YOLOv8 is the preferred choice for our accessibility framework due to its higher detection accuracy, faster inference speed, and suitability for deployment in low-resource environments.

4.6 Application of Faster R-CNN Using Detectron2 on same Dataset of Pie-Chart Dataset

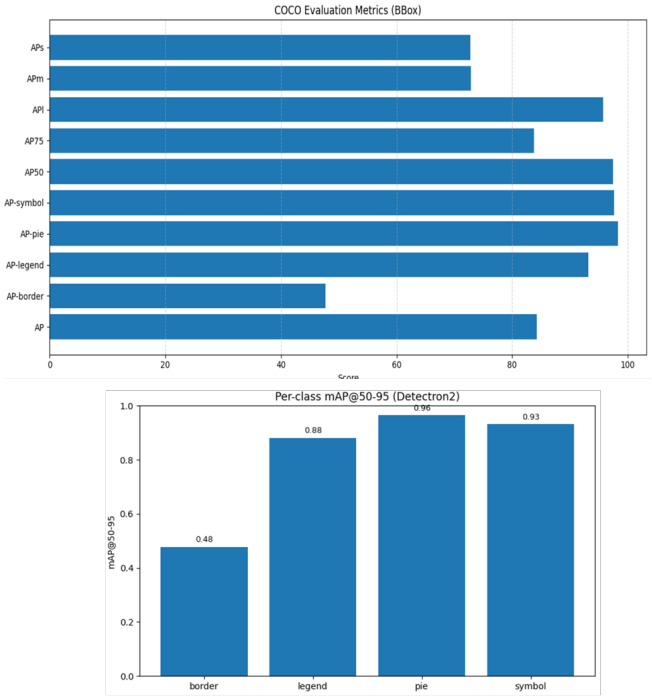


Fig. 7: Evaluation of results using Faster RCNN for pie charts .

The COCO evaluation metrics further indicate high average precision (AP) values across multiple IoU thresholds, with AP@50 reaching 0.97 and AP@75 maintaining 0.86. Class-specific AP scores demonstrate excellent detection of pie slices

(0.99), symbols (0.99), and legends (0.96), while borders remain weaker (0.48), reflecting the inherent difficulty in identifying fine angular boundaries.

The per-class mAP@0.5–0.95 distribution (bottom) confirms this trend, with pie slices and symbols achieving near-perfect values (0.96 and 0.93, respectively), legends perform strongly (0.88) borders significantly lower (0.48). These results highlight YOLOv8's robustness in detecting semantically important components for accessibility tasks, while suggesting that border detection could benefit from further dataset augmentation or refined labeling strategies.

5 Application: queries by BVI readers

The result of the object detection phase produces a custom XML file that we call "raw data". Its contents realize the formal semantics data structures defined in section 2. That file then serves as intermediary between the object detection and the "reader interface". Some readers are interested in summaries of the diagram, others want to survey all of the chart's content and that depends on context and uses. As a proof of concept we have designed a set of queries that the reader can issue to obtain the raw data information, or any more complex information that can be derived from it (figure 8).

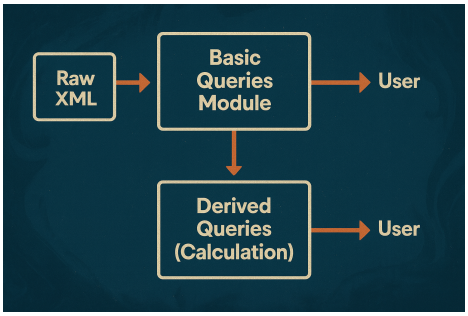


Fig. 8: Example of query visualization.

Table 1: Mathematical formulation of queries designed for blind users to interact with different types of bar charts.

Basic Queries	Description
Title()	Extract the chart title (all types).
ChartType()	Identify chart type: simple, grouped, stacked, or horizontal.
BarCount()	Return total number of bars. For grouped charts: groups × subcategories; for stacked: bars = categories (segments handled separately).
Legend()	List legend elements. Essential for grouped and stacked charts.
XLabel()	Return X-axis label (categories or groups).
YLabel()	Return Y-axis label (values/measurements).
XCategories()	List X-axis categories. For grouped charts: returns groups; for stacked charts: returns main categories.
YTicks()	List Y-axis tick values (or X-axis ticks if horizontal).
BarValue(i)	Return value of bar i. In stacked charts: may refer to (a) total bar height or (b) individual stack segment (must be specified).
Derived Queries	
ArgMax(BarValue)	Identify bar with maximum value. For stacked: clarify total vs. segment.
ArgMin(BarValue)	Identify bar with minimum value. Same clarification as above.
Mean(BarValue)	Compute average bar value (all bars, groups, or segments depending on chart type).
Range(BarValue)	Compute difference between maximum and minimum values.
Sort(BarValue, $\uparrow$ / $\downarrow$)	Sort bars ascending/descending. In grouped charts: must clarify whether sorting is global or within-group.
Compare(i, j)	Compare values of two bars. Works across all chart types (totals or sub-segments in stacked).
MaxChange(Category/Group)	Find category or group with largest change. Supports sub-bar projections (e.g., one subcategory across groups).
Summary()	Provide high-level summary of key trends (mentions grouped/stacked distinctions where relevant).

Table 2: Mathematical formulation of queries designed for blind users to interact with pie charts.

Basic Queries	**Description / Mathematical Formulation**
Title()	Extract the chart title.
ChartType()	Identify chart type (simple pie, donut, exploded, 3D).
SliceCount()	Return total number of slices n.
Legend()	List legend entries and their associated colors.
Labels()	Return slice labels (categories or annotations).
SliceAngle(i)	Return angular span θ_i of slice i. Two methods: (a) Area-based: $\theta_i = \frac{n_i}{N} \times 360°$, (b) Geometric: $\theta_i = (\beta - \alpha) \bmod 360$, where α, β are boundary angles from atan2.
SliceValue(i)	Return absolute or relative value of slice i. If percentages: $v_i = \frac{\theta_i}{360°} \times 100$. If absolute values: $v_i = \frac{\theta_i}{360°} \times V_{\text{total}}$.
SliceColor(i)	Return the color c_i of slice i.
Derived Queries	
ArgMax(SliceValue)	Identify the slice with the maximum value.
ArgMin(SliceValue)	Identify the slice with the minimum value.
Mean(SliceValue)	Compute average slice value: $\bar{v} = \frac{1}{n} \sum_{i=1}^{n} v_i$.
Range(SliceValue)	Compute difference between maximum and minimum values.
Sort(SliceValue, $\uparrow$ / $\downarrow$)	Sort slices by value ascending/descending.
Compare(i, j)	Compare values or angles of two slices: $\Delta v = v_i - v_j$ or $\Delta\theta = \theta_i - \theta_j$.
Percent(i)	Return normalized percentage of slice i: $p_i = \frac{v_i}{\sum_j v_j} \times 100$.
Summary()	Provide high-level summary of dominant slices, proportions, and trends.

5.1 Assistant for Bar Charts

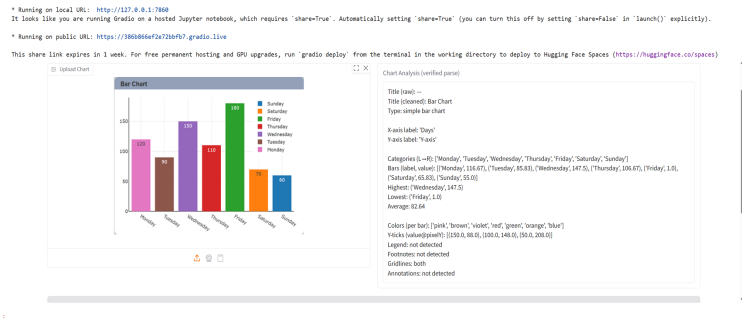


Fig. 9: Assistant for bar chart for blind people.

Figure 9 shows our interactive bar chart assistant for Blind readers . The system accepts an uploaded bar chart image and XML file and applies our detection and

parsing pipeline to extract its structural and semantic elements. On the left, the original bar chart is displayed, while on the right, the verified parse provides a structured textual representation.

The output includes chart metadata (title, type, axis labels), categorical information (days of the week), and numerical values for each bar (e.g., Monday: 116.67, Wednesday: 147.5). Derived statistics are also computed automatically, such as the highest bar (Wednesday: 147.5), lowest bar (Friday: 1.0), and overall average (82.64). Additional features include extraction of bar colors, pixel-based axis ticks, and basic chart properties (legend, footnotes, gridlines, annotations).

Beyond keyboard entry, users may also *speak* their queries. We implement low-latency audio I/O with `PyAudio`: the microphone stream is captured for on-device speech recognition, and answers are returned as synthesized speech and streamed back via `PyAudio`, while the same content is also shown as text for screen readers. This dual modality—*type or speak*—supports hands-free operation and inclusive access.WE can ask the assistant any of these queries example : Title of the chart, what is its type , how many bars are there,is there any legend or not ,which is the highest bar , which is the lowest bar etc .These queries we are asking we can see by our eyes by analyzing them but for blind readers there is no way except traditional tactile approaches ,there is no interaction exploration and feedback.

This demonstration highlights the assistant's ability to bridge visual content and non-visual accessibility by transforming bar chart images into structured, queryable descriptions that can be read by screen readers or explored through voice-based interaction.

6 Conclusions and Future Work

Our work introduces a full multimodal system that makes bar chart and pie chart information accessible to blind and visually impaired users. It uses advanced object detection, OCR, and semantic query generation to help these users understand the data. Through detailed testing, we found that YOLOv8 performs much better than Faster R-CNN in detecting important parts of bar charts, like bars, legends, axis labels, and tick marks. YOLOv8 achieved an AP@0.5 score of 95.9, with very high precision for labels and legends, showing that it can reliably detect even small and detailed elements needed to understand the chart. For pie charts, our framework supports slice detection, angle and value calculation, and semantic query answering.

These results show that a single-stage detector like YOLOv8 is very effective in situations where speed and accuracy are important—especially for real-time use in assistive devices with limited resources. The strong performance of YOLOv8 also helps improve other parts of the system, such as text-to-speech, answering questions, and interpreting structured data from the charts.

In addition to detecting elements, we developed a math-based query system that lets blind and visually impaired users explore bar chart and pie chart data using natural language-like commands. In future work, we plan to expand the

system to handle other diagrams too. We also want to improve OCR so that it works better with low-contrast and low-quality or complex layouts. We will explore the addition of voice assistants for easier interaction and conduct more user studies with a larger group of blind people to better understand how usable, easy to use, and trustworthy the system is in real-life settings such as education and work.

Moreover, recent studies show that almost all employed blind workers in the U.S. rely on screen readers such as JAWS and NVDA, often using multiple tools for efficiency[8]. Guided by these findings, we are planning to make our user interface fully compatible with screen readers by developing a standards-compliant HTML-based accessible page. This will allow blind and visually impaired users to benefit from our diagram recognition framework directly within standard browsers and screen readers (e.g., NVDA, JAWS, VoiceOver), without requiring specialized standalone software.

References

1. Bajić, F., Job, J.: Review of chart image detection and classification. International Journal on Document Analysis and Recognition (IJDAR) **26**(4), 453–474 (2023)
2. Dhote, A., Javed, M., Doermann, D.S.: A survey and approach to chart classification. In: International conference on document analysis and recognition. pp. 67–82. Springer (2023)
3. Gorniak, J., Kim, Y., Wei, D., Kim, N.W.: Vizability: Enhancing chart accessibility with llm-based conversational interaction. In: Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology. pp. 1–19 (2024)
4. Kim, J., Srinivasan, A., Kim, N.W., Kim, Y.S.: Exploring chart question answering for blind and low vision users. In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems. pp. 1–15 (2023)
5. Kumar, A., Ganu, T., Guha, S.: Chartparser: Automatic chart parsing for print-impaired. arXiv preprint arXiv:2211.08863 (2022)
6. Liu, X., Klabjan, D., NBless, P.: Data extraction from charts via single deep neural network. arXiv preprint arXiv:1906.11906 (2019)
7. Luo, J., Li, Z., Wang, J., Lin, C.Y.: 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)
8. McDonnall, M.C., Boydstun, J., Steverson, A.: Is one enough? screen reader use among employed people who are blind or have low vision in the us. Disability and Rehabilitation: Assistive Technology pp. 1–11 (2025)
9. Moured, O., Alzalabny, S., Osman, A., Schwarz, T., Müller, K., Stiefelhagen, R.: Chartformer: A large vision language model for converting chart images into tactile accessible svgs. In: International Conference on Computers Helping People with Special Needs. pp. 299–305. Springer (2024)
10. de Oliveira, C.L.T., de Almeida Silva, A.T., de Morais, J.M., Mota, M.P.: Accessible bar charts through textual description templates. Journal of the Brazilian Computer Society **29**(1), 1–18 (2023)