ChartSight: An Automated Scheme for Assisting Visually Impaired in Understanding Scientific Charts

Mandhatya Singh[a] and Puneet Goyal[b]
Indian Institute of Technology, Ropar, Punjab, India

Keywords: Visually Impaired, Chart Classification, Chart Extraction, Document Reader, Assistive Technologies.

Abstract: The visual or Non-Textual components like charts, graphs, and plots are frequently used to represent the latent information in digital documents. These components bolster in better comprehension of the underlying complex information. However, these data visualization techniques are of not much use to visually impaired. Visually impaired people, especially in developing countries, rely on braille, tactile, or other conventional tools for reading purposes. Through these approaches, the understanding of Non-Textual components is a burdensome process with serious limitations. In this paper, we present ChartSight, an automated and interactive chart understanding system. ChartSight extracts and classifies the document images into different chart categories, and then uses heuristics-based content extraction methods optimized for line and bar charts. It finally represents the summarized content in audio format to the visually impaired users. We have presented a densely connected convolution network-based data-driven scheme for the chart classification problem, which shows comparatively better performance with the baseline models. Multiple datasets of chart images are used for the performance analysis. A comparative analysis of supporting features has also been performed with the other existing approaches.

1 INTRODUCTION

A digital document contains mainly two components: Textual and Non-textual. Extraction and transformation of the textual components have been well explored (Goncu and Marriott, 2012). However, for the Non-textual parts (charts, graphs, plots, tables, etc.), the process is arduous and an active research area. The Non-textual or visual components are commonly used for representing the underlying quantitative information in a summarized way and generally preferred over the textual parts because of their cognitive advantages.

In today's world, we are equipped with advanced visualization methods to represent and understand the available information. However, visually impaired users (VIU) are deprived of insights and understanding as offered through these Non-Textual components. The VIU relies mainly on the textual components and often uses the braille technique for reading purposes; all the notable books and study materials are generally in braille. The braille has its own limitations. It is also challenging, and for some cases, it is currently impossible to represent these visualization methods through conventional techniques like braille. Apart from conventional techniques, VIU relies on screen readers (JAWS, NVDA ec.) to read and understand digital documents. However, these existing screen readers fail to comprehend the understanding from these Non-Textual components, causing loss of information. The existing chart understanding approaches and systems have limitations like manual intervention, generalizability, and compatibility.

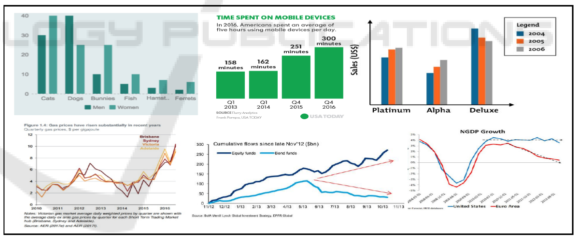

Figure 1: Sample line and bar charts demonstrating the variability and complexity in representation.

The complexity and design variability of the chart images makes the chart understanding process more challenging. The representation of the same quanti-

[a] https://orcid.org/0000-0001-9581-352X
[b] https://orcid.org/0000-0002-6196-9347

Singh, M. and Goyal, P.
ChartSight: An Automated Scheme for Assisting Visually Impaired in Understanding Scientific Charts.
DOI: 10.5220/0010201203090318
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP*, pages 309-318
ISBN: 978-989-758-488-6

tative data can be performed in different styles and through various chart types. Some sample line and bar charts used in this work, demonstrating the complexity and variability, are shown in Fig. 1. The proposed system ChartSight receives pdf files as input, specifically research papers and articles. It automatically extracts the Non-Textual components (all images) from the given pdf document. ChartSight classifies these images into respective chart categories (11 Categories). Non-Textual component classification is a critical step of an automatic chart understanding process as errors in this stage will propagate further in the chart understanding pipeline and affect the whole understanding process. In this work, we have primarily focused on the Non-Textual component classification task and propose a classification system to predict the respective component categories, that can be utilized in the content extraction stage. We have also proposed a preliminary content extraction approach based on image processing and heuristics-based techniques for content extraction from line and bar charts.

Through this work, the following contributions have been made :

1. We present a holistic trainable deep learning approach for Non-Textual (charts and tables) components classification.
2. This paper presents a preliminary heuristic based content extraction approach, respective to line and bar charts.
3. We performed a user response based analysis, demonstrating that in most of the graphs, there is always a deviation in perceiving the exact values, it differs from user to user.
4. This paper presents an automatic, interactive and easy to use graphical user interface, specifically customized for the visually impaired users (VIU), and its features comparison with recent methods.

The rest of the paper is organized as follows: Related work is described in Section 2. ChartSight Framework module, including classification, content extraction, and summarization sub-modules, is presented in Section 3. Experiment and Results are discussed in Section 4. Prototype and a comparative feature-based analysis with existing tools and approaches are discussed in Section 5 and 6. Conclusion and Future Scope are discussed in Section 7.

2 RELATED WORK

Assistive technologies related to reading the Non-Textual components for VIU had some less scholarly focus until recently. Screen readers like JAWS and NVDA are popular and preferred screen readers, particularly in developing countries. A comparative study between these two has been shown in (McCarthy et al., 2012). The issue with these readers is the access and efficient interpretation of graphical components. Standard devices like Flash PIAF and IVEO-3 can make graphs tactile images, but the hardware requirement is one of the concerns.

SIGHT (Elzer et al., 2007) assists VIU in reading bar charts, present in the web pages. GraphicReader (Nazemi and Murray, 2013) extracts the content from graphical components like bar charts, pie charts, line charts, and maths graphs. Graphics Accelerator (GraphicsAccelerator, 2020) is a tool, specially designed for VIU, for reading the contents of chart images. The tool's main concern is that it does not support general chart images; it only works for standard chart images prepared on their dedicated environment. For making the whole chart understanding process automatic, Non-Textual component classification is an important step. In literature (Huang and Tan, 2007), (Savva et al., 2011) various machine learning-based approaches have been proposed that rely on handcrafted features like histogram of oriented gradients (HOG), scale-invariant feature transform (SIFT) etc. Recently, various deep learning approaches (Jung et al., 2017), (Poco and Heer, 2017), (Choi et al., 2019) have been introduced for chart and table images classification task. The proposed approach comprises of a deeper architecture for capturing the inter-class dissimilarities and intra-class similarities.

Automated Analysis of line charts has been performed in (Nair et al., 2015). Emphasis on chart fields extraction has been done, rather than the content extraction. ReVision (Savva et al., 2011) classifies chart images into 10 categories and automatically extracts data from bar charts and pie charts. It has a classification accuracy of 80% on average, and it extracts marks successfully from 71% of bar charts and 64% of pie charts. ChartSense (Jung et al., 2017) is semi-automatic method that adopts a mixed-initiative approach for chart data extraction. FigureSeer (Siegel et al., 2016) is an application, utilizing quantitative and qualitative approaches for better performance, and is designed for querying charts, plots, and other figures in the scholarly articles. The existing figure question answering (QA) approaches (Kahou et al., 2017; Kafle et al., 2018) fails to retrieve numerical data from the charts in case of single color marks (i.e., single color bar). The ChartSight pipeline can aid existing chart figure captioning and figure QA approaches (Kahou et al., 2017; Kafle et al., 2018) by presenting the charts' raw content and thus improving

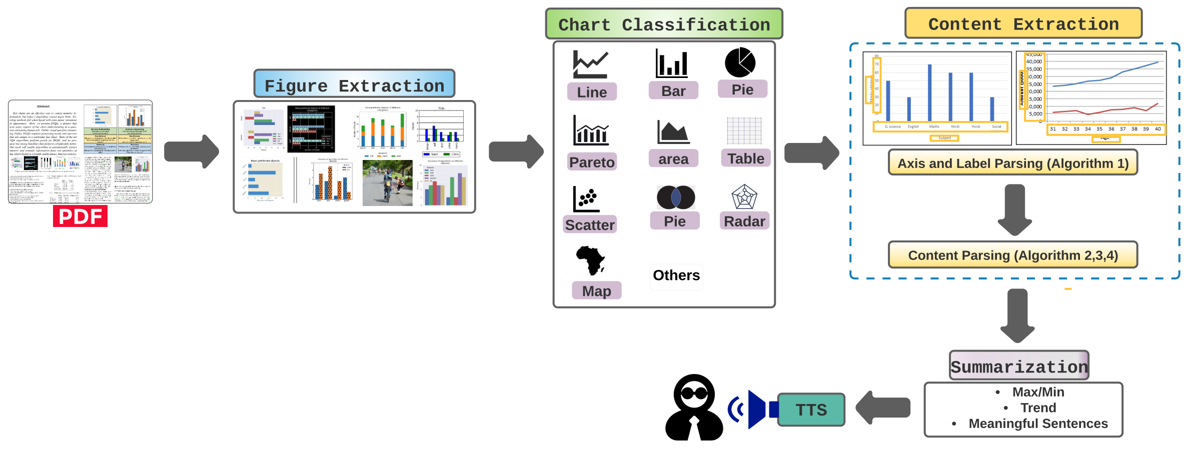

Figure 2: Block Diagram. The proposed system receives pdf document or chart image as input. It extracts the Non-Textual components (all images) from the document, classifies it into respective chart category (11 categories), extract the content, summarizes it and convert it into audio format using text-to-speech (TTS) engine for VIU.

the performance in case of reasoning based questions.

WebPlotDigitizer (Rohatgi, 2014) is a web-based tool and has both automatic and manual modes. Accuracy is a concern in automatic mode. In manual mode, users have to specify some initial points in the image. iVoLVER (Méndez et al., 2016) is a chart reconstruction, web-based tool and requires users' inputs for extraction. It heavily relies on the user for an efficient extraction process. Ycasd (Gross et al., 2014) also, requires user input for extracting the data from line charts.

3 PROPOSED FRAMEWORK

ChartSight offers an automated and efficient chart understanding process for VIU. The pipeline encompasses of following steps: (1) Image extraction, (2) Chart classification, (3) Content extraction, (4) Summarization. The pipeline has been shown in Fig. 2. Individual components in the pipeline are discussed below:

3.1 Figure Extraction

In this module, we have performed document segmentation and Image extraction. The document is segmented into two parts - Textual and Non-Textual. The Non-Textual segment consists of graphical (charts, plots, graphs, etc.) as well as generic images. The extracted images will get passed into the classification engine that classifies it into respective chart categories. For the purpose, we have experimented with three different pdf extractors libraries - (a) PyMuPDF - a python binding for the MuPDF library, (b) Poppler - a python binding for the poppler-cpp library, and (c) Minecart - a python package for figure extraction from PDF's. We have evaluated these libraries' performance over different PDF documents and selected the best performing one (results are presented in Section 4.1).

3.2 Chart Classification

Classification of chart images differs from classification of natural images. Some of the distinctive features/characteristics in chart images are: low pixel variances in neighborhood, similar pattern of graphical fields (i.e ticks, marks and labels), frequent occurrence of textual fields within image such as legend, labels, tick values etc, corner points and Edges distribution, spread of graphical part area and orientation of the graphical fields.

Recent advancements in deep learning approaches enable the network to incorporate more depth via short connections between the network layers. Theoretically, the deeper networks proved to be efficient, but empirically the training process in these deeper network suffers from issues like - *vanishing gradient* and *large number of parameters* further affecting the overall *information flow* across the network. The presented scheme utilizes the DenseNet (Huang et al., 2017) architecture, which eliminates the mentioned issues with robust feature propagation, feature reuse, and self regularizing ability. DenseNet provides the access (to each layer) of gradients from the loss function as well as the input layer, as shown in Fig. 3.

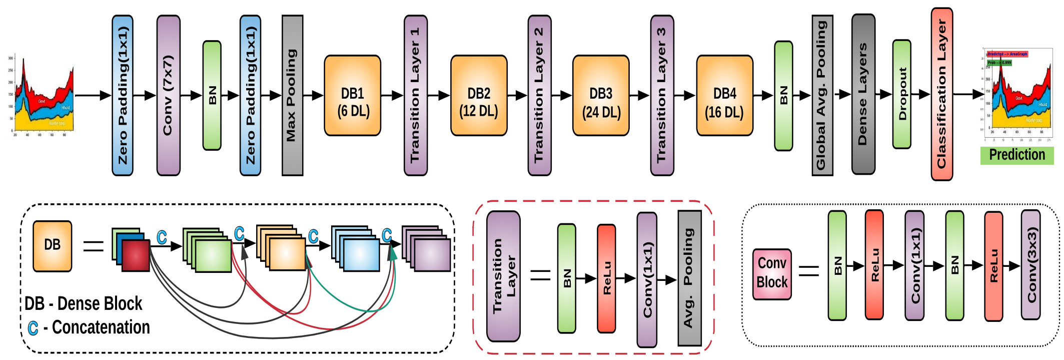

Figure 3: Proposed Classification framework. Upper network shows the overall flow. In lower part, the individual components are unfolded. Each dense block (DB) consists of multiple convolutional blocks. Four DBs with a total of 58 dense layers (DL) have been used.

3.2.1 Implementation Details

The two main components of the network are Dense blocks (DB) and Transition blocks. Each DB consists of multiple convolution blocks (CB). Each CB consists of two convolution layers of 1×1 and 3×3 stacked with Batch Normalization (BN) and ReLu [(BN-RELU-CONV(1×1)-BN-RELU-CONV(3×3)]. Four dense blocks with a set of 6, 12, 24, 16 CBs are used. For reducing the number of feature maps, each DBs are connected through transition layers. The transition layer improves the overall model compactness by compressing the model via [(BN-RELU-CONV(1×1)- AVERAGE POOLING] operations. The total number of layers in the presented architecture is 121. The growth rate is set to 32 in our experiment. Here growth rate denotes the size of feature maps produced at any layer and represents the actual contribution from that layer. We have used a global average pooling layer just before the fully connected layer for flattening.

The input image is resized to 224×224 as image dimensions vary within the data sets. An extra set of dense layers has been added with a dropout of 0.3 for regularization. We have initialized the learning rate with 0.1 with a decay rate of 1e-6 and Nesterov momentum of 0.8. SGD optimizer with categorical cross-entropy as loss function is used. The model is initialized with pre-trained weights on the ImageNet dataset. The first four layers are frozen to avoid relearning the same generic standard features of training images and accelerating the overall training process. The model is trained with mini-batches of size 16 for 50 epochs with data augmentation (horizontal flip, vertical flip, rotation, and zoom) and patience factor of 4. For efficient training, we have normalized the category weights for weighting the loss function (performed during training). So, the model can equally focus on data points from an *under-represented* category. The training and testing are performed on Nvidia GTX1080 GPU, and for other experimentation, Intel Core i7-8700 CPU has been used.

3.3 Chart Content Extraction

Automatic and efficient content extraction from chart images is a complex process. We have proposed a preliminary content extraction framework using image processing and heuristic-based approaches. Through our proposed algorithms [EHLC (*Extraction heuristics for line chart*) and EHBC (*Extraction heuristics for bar chart*)], we have improved upon the limitations of previous works including performance (in the absence of axis), and multi-scale support. The proposed algorithms utilize the texture as well as color-based information. It works effectively even when multi-color bar/lines are present in chart images. The proposed algorithms use some assumptions like chart image should be 2D, and axes appear at the left and bottom of the chart image. The content extraction steps and algorithms are described in the sections below.

3.3.1 Axes and Label Parsing

The Axes and Label parsing involves locating axes (X and Y), followed by extracting axes tick values and axes labels, as described in Algorithm 1. Before parsing, images are resized to 640x480. Hough (Duda and Hart, 1972) method is applied for detecting the vertical and horizontal lines. The extreme bottom line (EBL) and extreme left line (ELL) are selected from the detected lines as the X and Y-axis. Pattern matching and LSTM based OCR method (Kay, 2007) is applied to the regions below the EBL and on the left of ELL for axis tick and label recognition. All the axes tick and label regions are detected as text boxes. The text boxes with the same x values (below EBL) and the same y values (left of ELL) have been extracted. Further, for enhancement of the extracted fields, image sharpening (using unsharp masking) is applied. The proposed approach works even in the absence of axis lines in the chart images. In this scenario, a virtual axis is constructed with the help of axis ticks position. Firstly, OCR is applied over full chart image region for detection of all the textual components. A heuristic (denoted as HA in Algorithm 1) involving detecting the largest group of text boxes aligned with their x and y values are applied to the OCR output. These groups represent the axes tick marks and based on their position, both y and x-axis are created.

Algorithm 1: Axes and Label Parsing.

```
   Input    : I [Original Chart Image]
1  Resize I to $640 \times 480$ ;
2  DetectLines(I);
3  if DetectLines(I) $! = \phi$ then
4      Select ELL and EBL;
5      $TB \leftarrow OCR(EBL, ELL)$;
6      $TextBoxes \leftarrow DetectTextBox(TB)$ ;
7      $count \leftarrow length(TextBoxes)$ ;
8      while count $! = 0$ do
9          ImageSharpening(TextBoxes(count));
10         Save(TextBoxes(count)) ;
11         count = count-1 ;
12     end
13 else
14     $TB1 \leftarrow OCR(I)$;
15     $TextBoxes \leftarrow HA(TB1)$ ;
16     Repeat Steps 8 to 12 ;
17 end
```

3.3.2 Content Parsing

In the content parsing step, we aim to retrieve the embedded data in line and bar chart images (Algorithm (2,3)). The extracted data (image space) needs to be mapped to the data space (actual values in X and Y space). Generally, the Y-axis tick values are on linear scale or logarithmic scale. So, computing the scaling factor (yFactor) is an essential step in the MAPPING process (Algorithm 4). yFactor is calculated between each possible pair of Y-axis tick values, as shown in Eq. (1), and all these values are stored in yFactorList. Final yFactor is computed as a median of the yFactorList. This will ensure yFactor values correctness even when some of the labels were incorrectly detected or even missed.

$$\boxed{yFactor = \frac{[yTick_i - yTick_j]}{yP}} \tag{1}$$

Where,
yP = number of pixels between the $yTick_i$ and $yTick_j$,
$yTick_i$ = numerical value of $yTick_i$,
$yTick_j$ = numerical value of $yTick_j$.

In some cases where the maximum and minimum values of the Y axis are not present, the obtained results show some discrepancies. Both algorithms tackle this issue by correctly predicting those values based on yFactor. The proposed approaches use pattern matching and LSTM based OCR (Kay, 2007) engine for extracting the textual contents within the graph, i.e. (axes labels and axes marks values). The sensitivity factor is chosen on hit and trial based method. Morphological operation (opening) ensures the removal of irrelevant information.

Algorithm 2: ehlc: Line Detection.

```
   Input    : L [Line Chart]
1  $L1 \leftarrow MO(L,2)$; // Morphological operation
2  Convert L1 to HSV Space. ; // Only Hue
     channel is considered
3  $CL \leftarrow Clust(L1)$; // Clustering pixels having
   same ranges of color values
4  $NumLines \leftarrow Length(CL)$;
5  for $i \leftarrow 1{:}NumLines$ do
6      TRACING(CL(i)) ;
7      MAPPING((CL(i)) ;
8  end
```

Morphological operation (opening) is applied with a sensitivity factor of 2. It's tuning provides control over the selection of targeted lines and bars. It also removes the legends, labels, and gridlines which are not required in this stage. For further processing, the image is converted into HSV space. Clustering based on color values is performed. Each cluster accounts for the respective lines (multiple lines in case of a multi-line chart). For each cluster, the corresponding points are located in the image. Multiple points are located on each line by approximating the curve using the canny edge method termed as TRACING in Algorithm 2. In the intersection case, the next point of the corresponding line is traced using neighborhood pixels information. MAPPING process (Algorithm 4) is applied iteratively for each line to get the exact content.

MO, followed by horizontal line detection using the Hough method, provides the upper part of individual bars, as shown in Fig. 5 (j). X coordinates of the detected horizontal lines (of each bar) are matched (MATCHING) with the axes ticks positions, height, and span of each bar. MAPPING process is applied for each bar to get the exact content.

Algorithm 3: ehbc: Bar detection.

```
  Input     : B [Bar Chart]
1 $B1 \leftarrow MO(B,4)$;
2 $HorizontalLines \leftarrow HoughLines(B1)$;
3 Replace each horizontal lines with their
    middle point ;
4 $[UpperPartLowerPart] \leftarrow$
  $MATCHING(HorizontalLines)$; ;
5 MAPPING([UpperPart LowerPart]) ;
```

Algorithm 4: mapping.

```
  Input     : A : [Array of X & Y
               Co-ordinates]
1 counter = 0 ;
2 $L \leftarrow Length(A)$;
3 while counter ¡ L do
4     if A(counter(Xcordinate)) ==
        AXES(Xcordinate) then
5         NewYcordinate =
            (Ypixelval(Ycordinate)$*$ yFactor)
            +yMin;
6     end
7 end
```

Only those points are chosen in the MAPPING process whose X-axis ticks coordinate matches with extracted Y-axis ticks coordinates. Corresponding Y-axis values are calculated by multiplying the Y-axis pixel coordinates of that point and yFactor followed by adding yMin (Minimum value on Y-axis).

3.4 Summarization

In this module, we are generating a meaningful textual summary of the extracted content. Depending upon the figure type and data extracted, a textual summary considering the VIU persona usage is being generated. For example, increasing/decreasing trend, maximum/minimum, slope information, etc. We compare each data value corresponding to the respective axis tick for calculating the trend characteristics. The extracted visual fields were combined with the trend characteristics and then fed to the text-to-speech engine for converting it into audio format.

4 EXPERIMENTS AND RESULTS

4.1 Figure Extraction

We have evaluated the performance of all three pdf extractor libraries on scientific articles. 25 research articles from IEEE Xplore Digital Library, have been used for the performance analysis of the pdf extractors. A total of 96 images are present in the pdf documents. In Table 1, it is showed that PyMuPDF performance is better than the other two popular pdf extractors. PyMuPDF achieves 96.8% accuracy showing better rendering capability.

Table 1: PDF extractor's performance.

pdf extractor	Images Extracted	Accuracy
poppler	86	89.5%
minecart	82	85.4%
PyMuPDF	93	96.8%

However, extraction of images that contains watermarks is not possible with the current versions of these libraries. We have tested some pdf files that contain watermarked images, and found that, these libraries were not able to extract most of the images.

4.2 Chart Classification

The dataset, evaluation scheme and results for proposed classification approach are discussed in this section.

4.2.1 Dataset Description

For training and evaluation purpose, we have used MixChart, Savva(Savva et al., 2011) and Junior (Junior et al., 2017) datasets. Each image represents a single chart category.

- **MixChart.** This dataset consists of chart and table images collected from other available sources (Cliche et al., 2017), (Siegel et al., 2016) along with a set of Non-Textual images downloaded from the web. This dataset contains 11 categories of chart images (including "other" class). The total image count in this *MixChart* data set is 11,319.
- **(Savva et al., 2011).** A subset from ReVision(10 Category) dataset is acquired. We are only able to acquire 544 images from this dataset as most of the images in the dataset are not available. We have used this entire dataset only for testing different methods.
- **(Junior et al., 2017).** This dataset contains 10 categories of chart images (including table class). The total number of images are 4,891.

4.2.2 Evaluation Scheme

The evaluation scheme for estimating the effectiveness of the proposed classification framework is de-

Table 2: Description of evaluation scheme for proposed chart classification approach. # denotes the number of images and NA denotes not applicable.

Training Set	***Testing Sets***				
	MixChart (Test set)	(Junior et al., 2017) (Test set)	(Savva et al., 2011) (Complete)	(Junior et al., 2017) (Complete)	MixChart (Complete)
MixChart Training set # 9269 (11 classes)	S_1 # 2050	S_2 # 451	S_3 # 544	S_4 # 4440	NA
(Junior et al., 2017) Training set # 4440 (10 classes)	S_5 # 1786	S_6 # 451	S_7 # 544	NA	S_8 # 11319

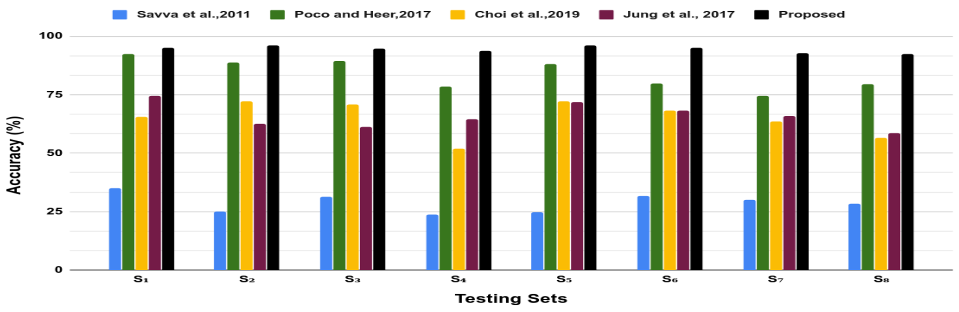

Figure 4: Dataset wise performance. The graph shows the weighted average accuracy of models across different test sets (S_1,S_2...S_8).

scribed in this section. We have evaluated the proposed framework based on two factors : *Generalization* i.e how the model generalizes when tested across different datasets (Dataset wise performance), and other is *Prediction* i.e how accurate the category wise predictions are across different datasets (Category wise performance). We have categorised the datasets into multiple train and test sets as shown in Table 2. The rows represents a combination for training and testing pairs. There are respective group of testing sets [$S_{j=1:4}$, $S_{j=5:8}$] for both the training sets. The test sets ($S_{j=1:8}$) are defined as $S = \{S_j : \bigcap_{j=1}^{8} S_j = \phi\}$ except pairs of (S_2,S_6) and (S_3,S_7). In spite of being the same set (S_2-S_6 and S_3-S_7), these both test sets pairs are formulated as different due to the different corresponding training sets. The evaluation scheme intends to capture the model performance on the similar sets when trained on different train sets. First row shows the models trained on MixChart (Train) set and tested on [MixChart (Test) set, (Junior et al., 2017) (Test) set, (Junior et al., 2017) (Complete) set, (Savva et al., 2011) (Complete) set]. Second row follows the similar pattern. Testing sets S_3, S_4, S_7, and S_8 consist all the images i.e (training + testing) from the respective dataset. For fair evaluation the respective *complete* (the *complete* denotes the full dataset) set of the training set have not been considered and it is denoted by NA as shown in Table 2. The set S_5 and set S_1 are the same sets, even then size(S_5) $<$size(S_1). The difference is of additional *Other* class in MixChart dataset. For dataset-wise performance, the average accuracy over each set $S_{j=1:8}$ has been computed. For category-wise performance, the category wise average accuracy for all test sets have been considered. Training and testing split is set to approximately 80-20 (80-train and 20-test) ratio for the MixChart dataset. Junior (Junior et al., 2017) dataset consists of pre-defined set of 4,440 images in training set (Junior(Train)) and 451 images in test set (Junior(Test)). The *complete* denotes the full dataset. Existing models (Savva et al., 2011), (Jung et al., 2017), (Poco and Heer, 2017), (Choi et al., 2019) are also evaluated with the same evaluation scheme.

4.2.3 Analysis

In this section, we have evaluated the proposed classification approach for Non-Textual images classification task. Fig. 4 reports the test set wise average accuracy (overall $S_{j=1:8}$) for all models. The proposed classification framework shows consistently top performance over all the test sets as compared to other approaches. These show the proposed approach better generalization aspect as compared to parallel approaches. Table 3 reports the category wise performance (average accuracy) of the models. The proposed framework is competent in capturing the intrinsic discriminative features of Non-Textual images. The results demonstrate the proposed classification approach robustness towards the inter class similarity problem (for example, Pareto chart holds a significant similarity with line graphs and bar graphs, same goes for a group of pie-radar-Venn charts (circles and fragments are common)). Also, specifically for these set

Table 3: Category wise performance (average accuracy) of the chart classification models.

Category	**(Savva et al., 2011)**	**(Jung et al., 2017)**	**(Poco and Heer, 2017)**	**(Choi et al., 2019)**	**Proposed**
Area	42.50%	59.10%	88.00%	79.26%	**96.73%**
Bar	60.21%	74.33%	89.41%	67.84%	**96.28%**
Pie	02.82%	88.40%	81.10%	69.09%	**97.18%**
Line	22.60%	36.63%	76.80%	63.32%	**87.88%**
Map	79.47%	34.82%	79.49%	12.11%	**91.52%**
Radar	28.76%	70.40%	85.84%	75.54%	**95.33%**
Table	30.50%	92.74%	86.97%	**96.60%**	94.56%
Scatter	14.16%	63.37%	77.11%	68.97	**92.00%**
Pareto	08.53%	51.32%	82.74%	37.51	**84.20%**
Venn	03.41%	84.44%	84.33%	87.50	**96.86%**
Others	02.70%	66.28%	92.20%	11.36%	**98.86%**

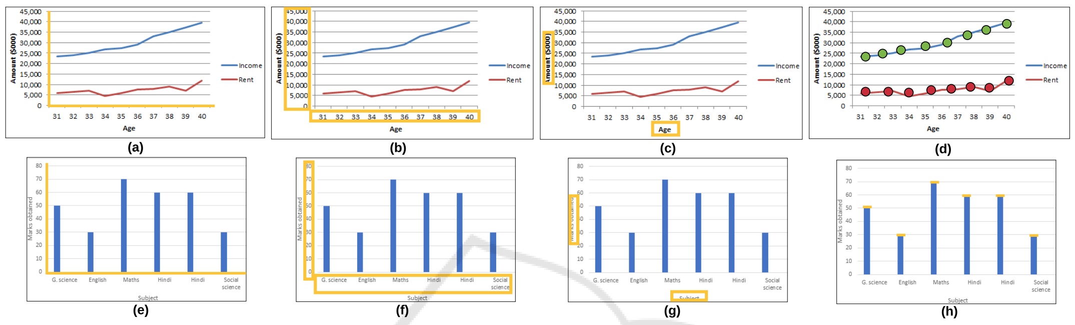

Figure 5: Results of proposed content extraction pipeline on two sample images from MixChart dataset. (a,e) Axes extraction, (b,f) Axis ticks extraction, (c,g) Label extraction, (d) Line tracing, (h) Bar detection.

of categories having high inter-class similarities, the machine learning model (Savva et al., 2011) have the least performance (*the handcrafted features are not detailed enough to capture the intrinsic differences*). The consistent and accurate results across different evaluation schemes show the proposed classification approach generalization and consistent prediction capabilities. Also, the proposed model has a comparatively lesser number of parameters, which is around 7.30 million as compared to (Poco and Heer, 2017) and (Choi et al., 2019) (*both have more than 20 Million Parameters*) models.

4.3 Content Extraction

The final content extraction process's performance depends on the sequential performance at each pipeline stage (axis extraction, axis tick values extraction, label extraction, and line and bar detection), as shown in Table 4. The results on sample images (for individual steps) are shown in Fig. 4. Randomly selected 200 images (100 line and 100 bar charts) from the MixChart dataset have been used to evaluate individual stages' performance in the pipeline. For evaluating the performance of axes and label parsing, we individually measured the accuracy of axes detection, axis-tick values, and label recognition. We have used the approach of (Siegel et al., 2016) for computing the axes, axis ticks, and label parsing accuracy.

Table 4: Chart content extraction performance.

	Line Chart	**Bar Chart**
Axes Detection	92.3%	94.6%
Axis tick recognition	85.2%	87.2%
Axis label recognition	82.1%	83.6%
Content extraction	80.6%	81.5%

For axes detection, the predicted bounding box (PB) of the respective field is compared with the ground truth bounding box (GB) of that field, based on computed Jaccard index(J) If J$>$ 0.6, then it regards as correct prediction. We have used the edit distance metric for evaluating the performance of axes tick, and axes label recognition.

We have considered the approximated tolerance value; if the predicted numerical content (Y-axis) is within the tolerance value, then it has been considered a correct prediction. We have demonstrated the results of intermediate steps in Fig. 5.

4.3.1 User Study for Tolerance Approximation

We conducted a controlled experiment to capture the variations in users' responses to approximate the tolerance value. The experiment's objective is to record the deviations and hence track the average tolerance for the exact content in line and bar charts. We have considered the prediction within this tolerance as an accurate prediction. The contemplation of chart images are more inclined towards the trend information

Table 5: ChartSight Feature Comparative analysis.

Approaches	**Automatic**	**Input (pdf)**	**Output (Audio)**	**Low Quality Images**	**Active Exploration**	**Multi-line color| black**
(Méndez et al., 2016)	-	-	-	-	-	-|-
(Rohatgi, 2014)	+	-	-	+	-	-|-
(Savva et al., 2011)	+	-	-	+	-	+|-
(Jung et al., 2017)	-	-	-	+	-	+|-
(Siegel et al., 2016)	+	-	-	-	+	+|+
(GraphicsAccelerator, 2020)	+	-	+	-	+	+|+
ChartSight	+	+	+	+	+	+|-

such as the trend is increasing or decreasing, the slope between points, etc. So, even in case of false extraction within some appropriate ranges can serve to understand the given charts.

We recruited 8 normally sighted participants from our institution. We have divided the participants into two categories, first who are well versed with graphs and charts; these participants were research scholars, second those who have an intermediate level of experience with the charts, these participants were undergrad students. At the starting of each session, we had provided a brief introduction regarding the procedure. Participants were asked to note down the content of the chart images. Each session takes around 55 minutes for completion. We analyzed the participants' responses and observed an average deviation of 4.8% for line charts and 3.1% for bar charts. This deviation has been used as tolerance in the performance analysis of the content extraction module. The performance is shown in Table 4.

5 PROTOTYPE

We implemented ChartSight as a desktop application. Users can input the individual pdf file, and in output, the content of Non-Textual parts (line and bar charts) will be represented in audio format. ChartSight can also be used as a chart content extractor; a user can directly provide the chart image as input and receive the raw and summarized content in audio format. ChartSight supports active exploration, as the user can access and explore the output at individual stages. VIU can use the ChartSight prototype through existing screen readers. A sample demonstration of the prototype as a chart content extractor has been provided with the article.

6 FEATURE COMPARISON

Comparison has been performed, based on the feature compatibility and capability of existing approaches. In Table 5, the comparison is shown based on overall supportive features that are required in the case of VIU. The first three features (Automatic, Input, Output) represent the compatibility, and the last three features represent the methods' capability. + shows the presence of that feature and - shows the absence of that feature in an individual approach. Features are chosen from both the content extraction potential and the essential requirements of visually impaired persons. Most of the approaches are not fully automated, making them unsuitable for the visually impaired scenario. Also, the system's input is a concern as all mentioned methods directly take the images as input instead of pdf. The analysis shows better performance ability and compatibility for the visually impaired scenario of ChartSight.

7 CONCLUSION AND FUTURE SCOPE

We presented ChartSight, a framework that classifies charts, extracts their graphical fields and underlying content and then represents the textually summarized content in audio format to facilitate better reading experience to visually impaired. We conducted a controlled experiment to show that there is always a tolerance regarding observing the chart images content. The results show that ChartSight is comparatively more relevant and efficient for visually impaired users. The proposed tool can also be used by sighted users for an enhanced audio reading experience of any pdf document, especially when multitasking. In the future, we would like to enhance our tool to support more features, including additional chart categories and evaluation on a much larger dataset. We would like to extend the content extraction pipeline. Focus on hybrid sequential models for generating textual summaries is also a future scope of the current work. We also aim to conduct usability studies, including visually impaired persons, to better access and address their needs.

ACKNOWLEDGEMENT

This research is supported by the IIT Ropar under ISIRD grant 9-231/2016/IIT-RPR/1395 and by the DST under CSRI grant DST/CSRI/2018/234.

REFERENCES

Choi, J., Jung, S., Park, D. G., Choo, J., and Elmqvist, N. (2019). Visualizing for the non-visual: Enabling the visually impaired to use visualization. In *Computer Graphics Forum*, volume 38, pages 249–260. Wiley Online Library.

Cliche, M., Rosenberg, D., Madeka, D., and Yee, C. (2017). Scatteract: Automated extraction of data from scatter plots. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–150. Springer.

Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

Elzer, S., Schwartz, E., Carberry, S., Chester, D., Demir, S., and Wu, P. (2007). A browser extension for providing visually impaired users access to the content of bar charts on the web. In *WEBIST (2)*, pages 59–66. Citeseer.

Goncu, C. and Marriott, K. (2012). Accessible graphics: graphics for vision impaired people. In *International Conference on Theory and Application of Diagrams*, pages 6–6. Springer.

GraphicsAccelerator, S. (2020). Product-specific resources.

Gross, A., Schirm, S., and Scholz, M. (2014). Ycasd–a tool for capturing and scaling data from graphical representations. *BMC bioinformatics*, 15(1):219.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Huang, W. and Tan, C. L. (2007). A system for understanding imaged infographics and its applications. In *Proceedings of the 2007 ACM symposium on Document engineering*, pages 9–18. ACM.

Jung, D., Kim, W., Song, H., Hwang, J.-i., Lee, B., Kim, B., and Seo, J. (2017). Chartsense: Interactive data extraction from chart images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 6706–6717. ACM.

Junior, P. R. S. C., De Freitas, A. A., Akiyama, R. D., Miranda, B. P., De Araújo, T. D. O., Dos Santos, C. G. R., Meiguins, B. S., and De Morais, J. M. (2017). Architecture proposal for data extraction of chart images using convolutional neural network. In *2017 21st International Conference Information Visualisation (IV)*, pages 318–323. IEEE.

Kafle, K., Price, B., Cohen, S., and Kanan, C. (2018). Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656.

Kahou, S. E., Michalski, V., Atkinson, A., Kádár, Á., Trischler, A., and Bengio, Y. (2017). Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*.

Kay, A. (2007). Tesseract: An open-source optical character recognition engine. *Linux J.*, 2007(159):2.

McCarthy, T., Pal, J., Cutrell, E., and Marballi, T. (2012). An analysis of screen reader use in india. In *Proceedings of ICTD 2012, the 5th ACM/IEEE International Conference on Information and Communication Technologies and Development*. ACM.

Méndez, G. G., Nacenta, M. A., and Vandenheste, S. (2016). ivolver: Interactive visual language for visualization extraction and reconstruction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4073–4085. ACM.

Nair, R. R., Sankaran, N., Nwogu, I., and Govindaraju, V. (2015). Automated analysis of line plots in documents. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 796–800. IEEE.

Nazemi, A. and Murray, I. (2013). A method to provide accessibility for visual components to vision impaired. *International Journal of Human Computer Interaction (IJHCI)*, 4(1):54.

Poco, J. and Heer, J. (2017). Reverse-engineering visualizations: Recovering visual encodings from chart images. In *Computer Graphics Forum*, volume 36, pages 353–363. Wiley Online Library.

Rohatgi, A. (2014). Web plot digitizer. ht tp. *arohatgi. info/WebPlotDigitizer/app/(accessed June*, 2.

Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M., and Heer, J. (2011). Revision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 393–402. ACM.

Siegel, N., Horvitz, Z., Levin, R., Divvala, S., and Farhadi, A. (2016). Figureseer: Parsing result-figures in research papers. In *European Conference on Computer Vision*, pages 664–680. Springer.