BarVQA: Reverse Engineering on Visualisation/ Text Recognition on Bar Charts to Improve the Chart Data Extraction

Shahira KC*, Ankit Raj, Anup Kumar, Shubham and Lijiya A

Department Computer Science and Engineering, National Institute of Technology Calicut, Kozhikode, Kerala.

ARTICLE INFO

Keywords:
visualisation
chart data
artificial intelligence
data extraction
reverse engineering

ABSTRACT

Visualisation data forms an integral part of web pages and documents. Chart image understanding is possible by automating the data extraction from the chart images. In this paper, we present a modified framework based on object detection to detect the visual elements of the bar charts. We propose an algorithm based on contour approximation to find the find the coordinates of the bar and this gives a higher IoU between the bounding box of the corrected bar coordinates and the ground truth. The text localization shows 5% improvement in the accuracy by introducing an image enhancement based on GANS (General Advesarial Networks) as a pre-processing to text recognition. The source code and dataset of this work are available at Github:

1. Introduction

Machine interpretation of the chart is a novel research area. Data visualisation in the form of chart pictures is an integral part of websites and other documents. Humans can read these graphs at a glance by associating the visual elements and the textual regions. However machines fails to do so []. From an artificial intelligence(AI) perspective, understanding charts in the same way as humans is achievable by reverse engineering the visualization pipeline.

Natural scene understanding involves identifying the object of interest from the scene given. Scene text detection is a related subfield that aims to locate text in complex scenes, a prerequisite for street signboard detection in autonomous vehicles [ref], document image understanding [refs] and many. There are several models for processing the document image that performs with high accuracy [ ] and even assist visually impaired people in reading those materials. However, there are hardly any when it comes to interpreting charts and graphs and answering questions about chart pictures. Figure 1 illustrates how the region of interest varies from a natural image and a chart image.

The works [1, 9, 11, 8] deals with data extraction for chart images. The work [9] introduced a synthetic visual reasoning corpus and they focuses on reasoning from chart images. The questions are based on binary answers on the synthetic data and no numerical answers are involved. Furthermore, the chart labels are set to the colour name of the corresponding plot element in the chart. While in [8] the dataset generation gave the control of the appearance of visual elements in chart images and access to the metadata which will be missing with real data. Since DVQA deals with synthetic data there is a limited variation in title, axis and legend labels. It is limited to bar charts only. The work [11] focuses on bridging the gap between synthetic and real-world charts. However, the visual element detection of chart pictures indicated the need for further accuracy enhancement. All of the models reported poor text recognition accuracy, which reduced overall accuracy.

We introduce a framework that improves chart element detection by [11] and also textual component recognition. We propose an algorithm for bounding box approximation of the bars and this gives a 5 % higher accuracy than the localization by objcet detection . The whole pipeline involves object detection for detecting the visual contents in the plot. We also employ the TAPAS model[6] to answer the query using the table. A pre-processing algorithm is also introduced which improves the text recognition by — % accuracy.

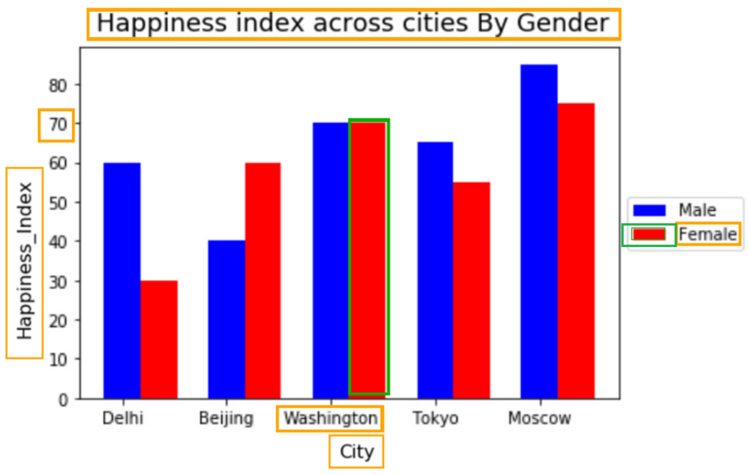
We choose bar charts because bar charts are the most common visualization type. There are variants among bar charts like simple bar charts, multiple bar charts, grouped bar charts, horizontal bar charts, vertical bar charts, and stacked bar charts. Bar charts reading errors are discussed in []. Our work looks forward to automating bar charts' reading using computer vision and AI algorithms.

1.1. Contributions of this paper

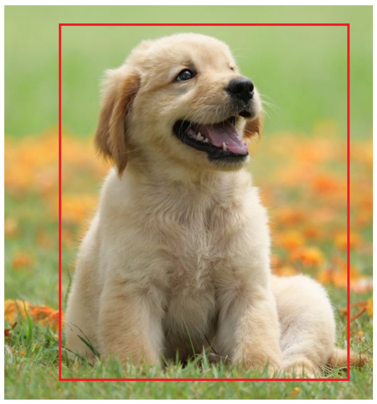
This paper aims to extract the raw data from the bar chart images and make the chart data available and thus making it accessible for all. In order to do so, the following contributions have been made:

- We survey the recent and state-of-the-art works in chart data extraction and the available datasets.
- We describe how to use object detection for finding the chart components, text extraction and recognition from the image, as well as a neural network capable of processing these representations to infer an answer.
- The proposed pipeline involves object detection for finding the chart components, text extraction and recognition from the image. The novel algorithm for bounding box approximation of the bars is introduced and is compared with the object detection module.
- The text recognition is improved by adding a pre-processing algorithm using super-resolution based on General Ad-

*Corresponding author
shahira_p170096cs@nitc.ac.in (S. KC)
ORCID(s):



(a)



(b)

Figure 1: A chart image vs natural image. The bounding box proposals are marked to highlight the visual elements in each of them. Fig.(a) Understanding a chart element corresponding to a bar involves associating the texts, x value, y value and the legends marked by boxes. Fig (b) The region of interest is a puppy in the foreground

versarial Networks (GANs) to enhance the textual quality. This shows an accuracy of ..%.

- We use the table question answering using TAPAS [6] on the the chart data extracted as a table.

The remainder of this article is organised as follows: Section 2 examines recent techniques towards chart data extraction. The proposed methodology and results are discussed in Section 3 and Section 4 respectively. Section 4 is devoted to performance analysis, while Section 5 finishes the article by discussing future developments.

2. Related Works

In this section, we review the related work in chart reverse engineering. Chart images contain graphical primitives like bars, lines, dots, and a textual portion. Text portion can be the title, X-axis ticks, Y-axis ticks, X-label or Y-label. Data extraction from a chart involves identifying the graphical primitives and the textual data and associating a value to these primitives.

Extracting visual encodings from bar chart images is solved by image processing and feature extraction as in [19, 18]. The quality of the feature extraction is based on the accuracy of the data extraction method in these methods. The textual detection by OCR failed 80% of the time. In [20] uses Hough primitives for bar pattern recognition while data extraction was not considered. Traditional methods alone make it difficult to extract the original data that was used to generate these charts.

The paper [15] reviews the traditional methods and recent trends in visualization data, the potential future of artificial intelligence in reverse engineering the chart data. Some methods for visual question-answering on chart pictures have been proposed, including DVQA [8], FigureQA [9], LEAFQA [1], and PlotQA [11]. Each one of these models can be separated into two parts. *(a)* The process of extracting visual and textual information from a graph. *(b)* The different ways in which different models use data to respond to user enquiries.

Kafle et al. [8] uses DVQA dataset which contains almost 3 million picture question-answer pairs connected to bar chart graphics. It employs two distinct algorithms. SANDY: SAN with DYnamic Encoding and Multi-Output-Model (MOM) . The DVQA Model Multi-Output-Model employs a dual-network design where one sub-networks produces chart-specific answers. Optical character recognition (OCR) in the Multi-Output-Model generates chart-specific answers. SANDY uses a dynamic encoding technique to encode chart-specific terms in the query and create chart-specific replies. It builds a local word dictionary with the aid of the OCR system, which provides the location and string for all text areas in the bar chart. After that, it assigns indices to all of the detected boxes. However, this approach is only applicable to the bar chart.

The work by Samira et al. [9] works with five distinct types of charts: line, dot-line, vertical and horizontal bar charts, and pie plots. The dataset introduced is FigureQA and it contains around 1 million question-answers and 100,000 chart pictures. The researchers of FigureQA deployed a relation network (RN). Relation Network encodes the pairwise interaction for every pair of objects in the image and employs this to answer queries involving relationships. However, this model merely provides a Yes/No answer. The major drawback of both of these models is that their datasets are synthetically generated rather than real-world datasets.

FigureQA datasets are created using Bokeh, whilst DVQA datasets are created using Matplotlib.

PlotQA [11] is a real-world dataset comprising 28.9 million question-answer pairs spread over 224,377 visualizations based on real-world data. PlotQA's primary model consists of a binary classifier determines if a question can be answered using a small fixed vocabulary or whether more complex reasoning is required. A simple tiny classifier to answer questions with limited language (such as "how many bars are there in the chart?"). The model uses visual elements detection to extract the components from the chart and assign to the relevant class followed by an OCR to extract the textual components in the chart image. The extracted data from the graphic is then converted into a semi-structured table. The table is converted into a knowledge graph and the question into a set of candidate logical forms.

LEAFQA [1] is another approach that uses real-world data to create considerably more complicated variations in the dataset. It extracts distinct items from the chart using bounding box creation, and then classifies them into twenty categories. Paragraphs are generated for each question template using the Google Translate API, and one of the paraphrases is chosen at random. The chart-specific sections of the paraphrase are substituted with words from the chart.

3. Methodology

This section proposes data extraction from bar chart images using traditional and machine learning-based methods. The former method successfully extracts a few structural properties of the bar chart, but an end-to-end information extraction was limited. Therefore as an improvement to this method, we propose data extraction utilizing computer vision and deep learning methodologies.

3.1. Data extraction from the charts using the traditional methods

Initially, data extraction using traditional image processing methods was done in an attempt to make the chart data available to everyone. Image is processed to find the structural details of the graph by Algorithm 1. Algorithm 1 takes the processed image and an empty array as an argument and returns the graph's most prominent and smallest values, as given in figure 2. By the proposed Algorithm 1 we were able to extract the minimum and maximum values in a bar chart as well as the width of the bar. Further information extraction and reasoning was more accurate and easier by using AI algorithms. As a result, in the proposed methodology, we examine data extraction utilizing computer vision and deep learning methodologies.

3.2. Data extraction from the charts using the AI based methods

In this section, we propose decoding data from bar charts among the chart images. This is because a public, well-annotated dataset available for real value data is only PlotQA [11], while for other types of charts, it is very rare. Also, due to variations in the chart types, the proposed pipeline may not work for other kinds of charts.

Algorithm 1: Algorithm for Data Extraction from Bar Graphs

```
1 Input: The Processed image
2 Output:The largest and smallest values in a bar
  chart
 1:  $Xmax$ = width of the image;
 2:  $Ymax$ = height of the image;
 3:  A = [ ]
 4:  im1[ ] = input image
 5:  value = 0
 6:  for $i = 1$ to $Ymax$ do
 7:      for $j = 1$ to $Xmax$ do
 8:          $value = im1[i, j] + value$
 9:          $A.append(value)$
10:      end for
11:  end for
12:  largest = A[0]
13:  lowest = A[0]
14:  for $i = 1$ to $length(A)$ do
15:      if A[i] > largest then
16:          largest2 = largest
17:          largest = A[i]
18:      else if largest2 == None or largest2 < A[i] then
19:          largest2 = A[i]
20:      end if
21:  end for
```

The different types of document images are categorised into different classes like bar charts, pie charts, line charts and other figures as in [14]. A Convolutional Neural Network (CNN) with six layers similar to AlexNet with Adam optimiser and Relu gives 99.01% accuracy. Further, information extraction from bar charts is considered.

The design of the pipeline diagram shown in Figure 3 depicts the entire process of analysing bar chart images and answering questions based on them, which includes the following stages:

- Detecting components from a chart, such as bars, legend names, titles, x-tick labels, y-tick labels , axes names and others, by drawing a bounding box around them.
- Correcting these bounding boxes to increase IoU value.
- Pre-processing involving General Adversarial Networks (GAN) based super-resolution.
- Detecting text using OCR from the bounding boxes.
- Using this information, creating a table.
- Table Question Answering, using the table information to answer queries linked to a chart.

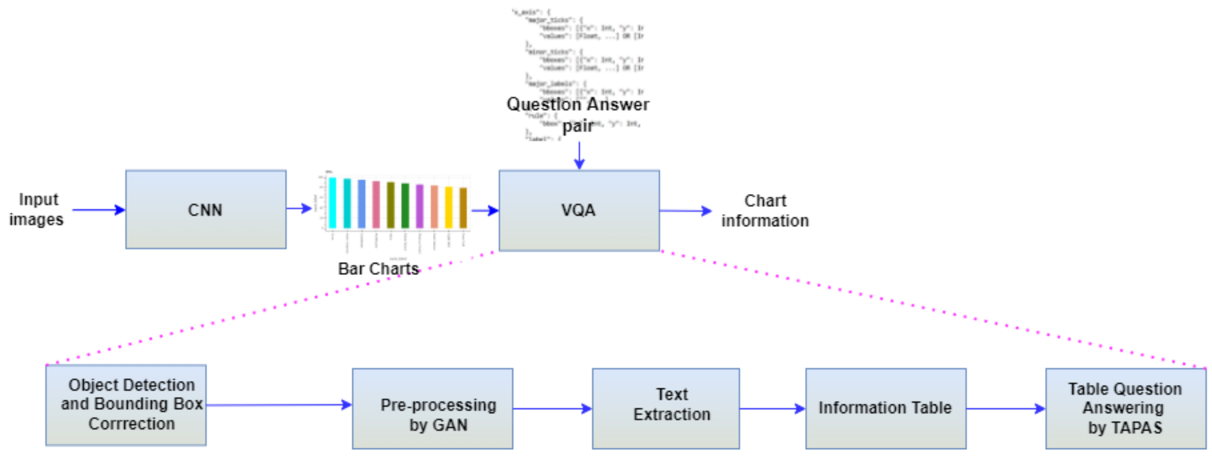
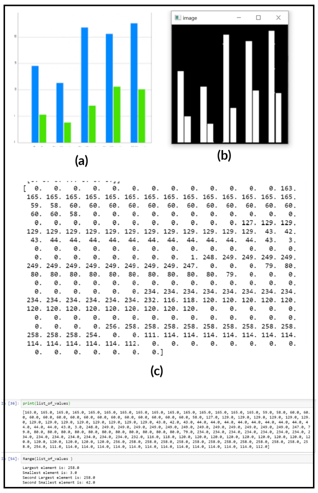


Figure 2: The pipeline for the proposed method on a random input bar image



```
[   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0. 163.
 165.  165.  165.  165.  165.  165.  165.  165.  165.  165.  165.  165.  165.  165.
  59.   58.   60.   60.   60.   60.   60.   60.   60.   60.   60.   60.   60.   60.
  60.   60.   58.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  127.  129.  129.
 129.  129.  129.  129.  129.  129.  129.  129.  129.  129.  129.  129.   43.   42.
  43.   44.   44.   44.   44.   44.   44.   44.   44.   44.   44.   44.   43.    3.
   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
   0.    0.    0.    0.    0.    0.    0.    0.    1.  248.  249.  249.  249.  249.
 249.  249.  249.  249.  249.  249.  249.  249.  247.    0.    0.    0.   79.   80.
  80.   80.   80.   80.   80.   80.   80.   80.   80.   80.   79.    0.    0.    0.
   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
   0.    0.    0.    0.    0.    0.  234.  234.  234.  234.  234.  234.  234.  234.
 234.  234.  234.  234.  234.  234.  232.  116.  118.  120.  120.  120.  120.  120.
 120.  120.  120.  120.  120.  120.  120.  120.  120.    0.    0.    0.    0.    0.
   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
   0.    0.    0.    0.  256.  258.  258.  258.  258.  258.  258.  258.  258.  258.
 258.  258.  258.  254.    0.    0.  111.  114.  114.  114.  114.  114.  114.  114.
 114.  114.  114.  114.  114.  112.    0.    0.    0.    0.    0.    0.    0.    0.
   0.    0.    0.    0.    0.    0.    0.]
```

```
In [36]: print(list_of_values)

[163.0, 165.0, 165.0, 165.0, 165.0, 165.0, 165.0, 165.0, 165.0, 165.0, 165.0, 165.0, 165.0, 165.0, 165.0, 59.0, 58.0, 60.0, 60.
0, 60.0, 60.0, 60.0, 60.0, 60.0, 60.0, 60.0, 60.0, 60.0, 60.0, 60.0, 58.0, 127.0, 129.0, 129.0, 129.0, 129.0, 129.0, 129.
0, 129.0, 129.0, 129.0, 129.0, 129.0, 129.0, 129.0, 129.0, 43.0, 42.0, 43.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 4
4.0, 44.0, 44.0, 43.0, 3.0, 248.0, 249.0, 249.0, 249.0, 249.0, 249.0, 249.0, 249.0, 249.0, 249.0, 249.0, 249.0, 249.0, 247.0, 7
9.0, 80.0, 80.0, 80.0, 80.0, 80.0, 80.0, 80.0, 80.0, 80.0, 80.0, 80.0, 79.0, 234.0, 234.0, 234.0, 234.0, 234.0, 234.0, 234.0, 2
34.0, 234.0, 234.0, 234.0, 234.0, 234.0, 234.0, 232.0, 116.0, 118.0, 120.0, 120.0, 120.0, 120.0, 120.0, 120.0, 120.0, 120.0, 12
0.0, 120.0, 120.0, 120.0, 120.0, 120.0, 256.0, 258.0, 258.0, 258.0, 258.0, 258.0, 258.0, 258.0, 258.0, 258.0, 258.0, 258.0, 25
8.0, 254.0, 111.0, 114.0, 114.0, 114.0, 114.0, 114.0, 114.0, 114.0, 114.0, 114.0, 114.0, 114.0, 114.0, 112.0]

In [54]: Range(list_of_values )

Largest element is: 258.0
Smallest element is: 3.0
Second Largest element is: 258.0
Second Smallest element is: 42.0
```

Figure 3: *(a)* Input bar chart *(b)* The Threshold image *(c)* The intermediate processing step towards extracting the chart information by algorithm 1 *(d)* The largest and smallest values and the bin size

3.2.1. Object Detection and Bounding Box Approximation

The chart's information-containing elements belong to ten categories: x-labels, y-labels, title, x-tick labels, y-tick labels, legend, preview, line, dot-line, and bar. The first task of our model is to take the chart as input and detect the visual elements or the objects present in that chart by drawing bounding boxes around them and categorise them among the ten classes. This can be accomplished by using any of the object detection models like YOLO[ ], SSD[ ] or Faster RCNN [ ].

We choose Mask-RCNN because it outperforms the winners of the COCO 2016 challenge, and it is easier to train [5]. We use Mask- RCNN [ ] because it gives the best performance compared to other models, as shown in Table 1. Mask-RCNN is built on top of Faster-RCNN []. It uses Resnet-101 to extract the visual features, and these feature maps are forwarded to a Region Proposal Network (RPN), which produce region proposals or candidate boxes. A fully connected convolution layer at the end classifies and outputs the bounding boxes and the segmentation mask for each region of interest with IoU greater than 0.5.

The bar chart images and their related annotations in the *.json* format make up the input data. Mask- RCNN [5] is used for object detection and localisation of the chart elements given by Figure 5.

- *Training:* The training set consists of 1,05163 vertical and horizontal bar charts and 3.3 million question-answer pairs selected from PlotQA[] dataset, which is based on data from real-world sources and questions based on crowd-sourced question templates.

- *Testing:* The testing set has around 22,528 images, 725285 question-answer pairs form PlotQA's test set.

Mask- RCNN [5] is suitable for classification jobs on this dataset, but we need improved accuracy for localisation. We need a bounding box with a greater IoU to determine the precise length of the graph, and we are enhancing the bounding box accuracy of the bars by utilising Laplace edge detection. We use a Laplacian filter to convolve the given image. We obtain an image that comprises the edges of the input image. The laplacian is preferred because it is a second-order derivative operator that finds only the points that have local maxima in gradient values and consider them edge points [12]. It is isotropic and this helps to find the horizontal and vertical bar edges more accurately while compared to a manually annotated bounding box.

Whenever there is an edge, the Laplace edge detector draws an edge on both sides of the detected edge, thus there will be an edge on the outside and an edge on the inside for each bar of the image. We draw an edge around the inner edge and that can be used for the localisation of the bar. We use the coordinate of the detected contour as the coordinate of the bounding box. Candidate bounding boxes are approximated to the minimum bounding box. Bounding Box approximation is a crucial step in this because the model behaves well for natural images with high IoU but for

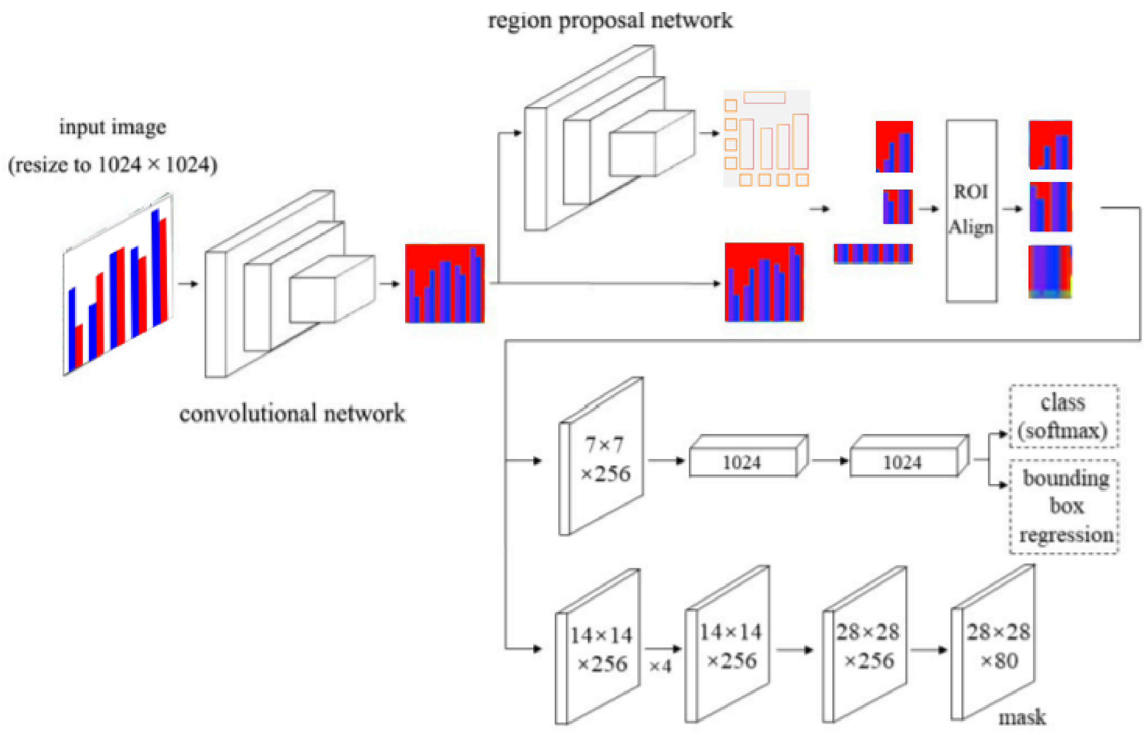


Figure 4: The model diagram of a Mask-Rcnn [5]

Dataset split	Vbar	Hbar	Question-answer pairs
Training	52463	52700	3387525
Validation	11249	11292	7255286
Testing	5646	11292	362643
Total	139039		11005454

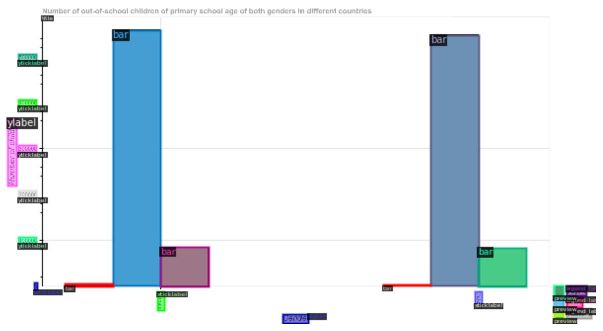


Figure 5: The architecture of Mask R-CNN [5]

chart images, it works with less IoU as given in Table II. The proposed bounding box proposals from edges, use the area of minimum contours to find the candidate boxes. There is a match between the predicted and ground truth bounding boxes if Iou exceeds a threshold of 0.5. Mean of this IoU scores averaged over the entire training set gives a higher IoU than without doing the bounding box approximation (results in 5.2).

3.3. Text Localization and Recognition

Text detection in images and videos is indeed a widely discussed research area. The textual component of that image conveys the majority of the information in it. As a result, it is critical to extract text with as little error as possible. The text on a traffic sign displays the caution to be taken or speed limits, the text in the news depicts the incident, and the text on a chart depicts the data analysis.

The title, X and Y-axis ticks, legend names, and X and Y axis titles are all part of the textual element. Because of the small size, variable font, and orientation of the text in the image, the existing OCR technology often fails to recognize it.

We present an algorithm for detecting text in bar chart images that can be utilized as a preprocessing algorithm.

The accuracy of recognition by OCR increases ——% by this.

The textual areas were missed in the detection. The following are a few potential reasons for the missed detections.

- The text is missed in detection due to tiny font sizes and a diversity of typefaces.
- Annotations were not completed for the smaller font sizes. Fine character annotated datasets are rarely found,

Algorithm 2: Algorithm for correcting the coordinates of the bar and calculating IoU of bounding boxes between ground truth and corrected one.

```
Input: Image $\mathfrak{J}$, ground truth bounding box coordinates ($x1^{gt}$, $y1^{gt}$, $x2^{gt}$, $y2^{gt}$)
Output: Coordinate points of bars and $IoU$
    $Initialisation := 0$ , Area of ground truth box $A^{gt} = (x2^{gt} - x1^{gt}) * (y2^{gt} - y1^{gt})$
 1: $A^{gt} = (x2^{gt} - x1^{gt}) * (y2^{gt} - y1^{gt})$
 2: $A^{p} = (x2^{p} - x1^{p}) * (y2^{p} - y1^{p})$
 3: for $i = 1$ to $n$ do
 4:     $\mathfrak{J}[i]' = \text{Laplacian}(\mathfrak{J}[i])$
 5:     $\mathfrak{J}[i]'' = \text{Contour}(\mathfrak{J}[i]')$
 6:     if contour overlap $\geq 0.8$ then
            BC ($x1^{bc}$, $y1^{bc}$, $x2^{bc}$, $y2^{bc}$) = contours (x1 y1, x2, y2)
 7:     end if
 8:     $A^{bc} = (x2^{bc} - x1^{bc}) * (y2^{bc} - y1^{bc})$
 9:
            $$IOU = \frac{\cap_{i=1}^{n}(A_i^{bc}, A_i^{gt})}{\cup_{i=1}^{n}(A_i^{bc}, A_i^{gt})}$$
10:     $\wp = \wp + IoU[i]$
11: end for
12: return $\wp$
```

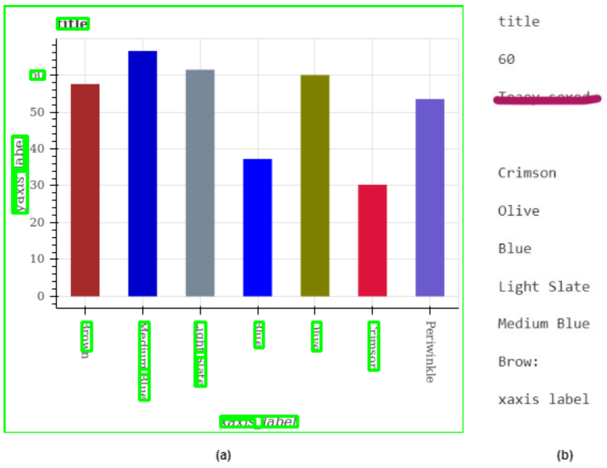


Figure 6: (a) The text detections proposals generated FigureQA dataset (b) The detected text

and this makes the object detection on small text a difficult task [cite].

- We can address each of these concerns. To begin with, missing text localization owing to poor resolution can be rectified by employing SR methods as pre-processing.

3.3.1. GAN based enhancement for text recognition

Figure 3 shows the text region detected on Microsoft's FigureQA dataset and the corresponding text detected by OCR. The extra text that appeared is struck. The vertical text at the Y-axis is missed out, and the small texts along the Y-axis are also missed out.

To improve text recognition, we employ GAN-based preprocessing. During the OCR, the textual regions were missed. In a natural image, a misplaced character or word might be ignored, but in charts, this information is vital. Figure 5 shows the textual components in a natural image vs barchart image.

The text in the chart graphic contains crucial information. As a first step to image enhancement, SRGAN, SR-Resnet, ESRGAN, Real-ESRGAN are done.

Text recognition algorithms have found an improvement in performance due to deep learning. Nevertheless, text recognition in chart images has not yet received attention. A novel pre-processing algorithm for text image restoration or enhancement by using GAN is proposed. Our experiment on the public dataset shows an improvement in the recognition accuracy of chart text. Specifically, ESRGAN [] performs well comparing to SRGAN [] and SRResnet [] and a few other interpolation methods.

Image enhancement improves the quality of image for improving perception. Popular image enhancement methods include interpolations, zooming and super resolution. Text recognition in the scene is accomplished by MSER [] and super resolution methods. Therefore we compare our method with them.

Zooming by nearest neighbour interpolation is simple, but the output is blurred. Maximally Stable Extremal Regions (MSERs) was successful in scene text detection in [7]. MSER followed by Stroke Width Transform (SWT) is used for finding text candidate regions [4, 2]. Natural scene text detection using MSER and SWT improved the text detection rate. GAN is used primarily for generating realistic images; it is applied for producing high-resolution images. Text super-resolution using GANs are used in [], [].

A few GAN based super-resolution methods on scene text images and document images are compared with bicubic, and MSER [] based methods for analysis. We found that the ESRGAN method gives better text detection and therefore, we use this method in our pre-processing algorithm.

Single image super-resolution produces the high resolution (HR) counterpart of its low resolution (LR) image. The use of GANs for super-resolution is introduced by [10] to restore the fine texture details. GAN based networks consist of a generator, G and a discriminator D as given in Figure 4. The low-resolution image is produced from the high-resolution image by down sampling. The generator G generates the HR image from the input LR image.

The G is trained with the goal of fooling the discriminator D that is trained to distinguish super-resolved images from real HR images. Write on Losses.....

SR is used in text recognition [17, 16]. Super resolution on in Street View Text (SVT) from Google Street View [],... is successful in natural scenes. The SR for text image enhancement is done on cropped word image recognition in datasets: ICDAR 2013 (ICR13) [16], in IIIT 5K-Words (IIIT5K) [17]. All these are natural images while [17] is based on document images. SRRGAN [17] shows an improvement of 10-20% improvement in accuracy in scene and handwritten texts. This shows the relevance of image enhancement by SR methods in text recognition. The same method is applied to chart images for the first time.

diff types of enhancements figure expalain what is SR, Why you do it here? comparision results

Algorithm 3: Algorithm for pre-processing for enhancing the textual areas based on Adversarial Networks

```
Input: Chart image $\mathfrak{I}$
Output: Preprocessed image $\mathfrak{I}$', text
    Initialisation :
1: for $i = 1$ to $n$ do
2:     $\mathfrak{I}$[i]' = Laplacian($\mathfrak{I}$[i])
3:     $\mathfrak{I}$[i]" = Contour($\mathfrak{I}$[i]')
4: end for
5: return $\wp$
```

3.4. Data Extraction

Algorithm for Data Extraction from Bar Graphs

3.5. TAPAS

4. Evaluation Metrics

The performance is evaluated using the following metrics: *(a)*text-line detection rate, *(b)* precision, *(c)* recall, *(d)* F-score, and *(e)* IoU.

The evaluation of the entire method is difficult as there is no common work on which the various works can be quantitatively compared. Some work focus on object detection of the chart elements and reasoning, data extraction and while few works on text extraction in chart images. There are very

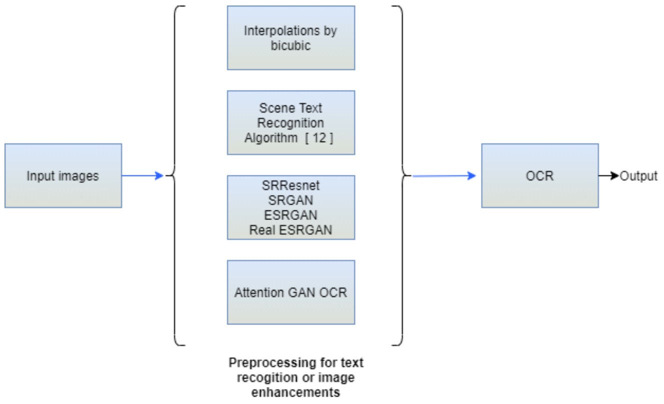

Input images
Interpolations by bicubic
Scene Text Recognition Algorithm [12]
SRResnet
SRGAN
ESRGAN
Real ESRGAN
Attention GAN OCR
Preprocessing for text recogition or image enhancements
OCR
Output

Figure 7: Image enhancement or zooming techniques for detecting small texts

few pipelines in total, and if there are any, the results are based on different datasets and models. As a result, we'll analyse the system by using various metrics for evaluating various modules.

4.0.1. Object Detection and Bounding Box Approximation

$$\text{IOU} = \frac{\text{area}\left(B_p \cap B_{gt}\right)}{\text{area}\left(B_p \cup B_{gt}\right)} \tag{1}$$

4.0.2. Text Detection and Recognition

4.0.3. GANs

4.0.4. TAPAS

5. Results and Discussions

5.1. Object Detection and Bounding Box Approximation of Bar chart

Mask-RCNN is used to locate visual cues in chart images. Figure 10 shows the bounding box proposals and the class labels on the input image in Figure 9. After training Mask-RCNN for object detection and localization, we evaluated it on a dataset consisting of twenty-two thousand horizontal and vertical bar images. Table 3 shows the average precision (AP) for different categories at different IoU thresholds, mainly at 0.5, 0.75 and 0.9. For a high threshold IoU value, the Average Precision is low. The Average Precision for IoU threshold 0.5 is 95.10. The average precision at IoU 0.5 is desirable for any object detection on natural images, but for bar elements, the overlap of gt and pp box should be maximum to get the exact length of the bars. In table 3, the AP at $\text{IoU}_0.9 is the maximum for every class using Mask-RCNN$.

Figure 9 clearly shows that the bounding box for the bar is not perfectly giving the height of the bar. In terms of improving localization accuracy, we propose algorithm 2.

The proposed algorithm (Algm 2) corrects the bar's bounding box to give the exact length of the bar, resulting in specific bar value. We took 22,000 images and compared the IoU values of the bounding box of a different bar before

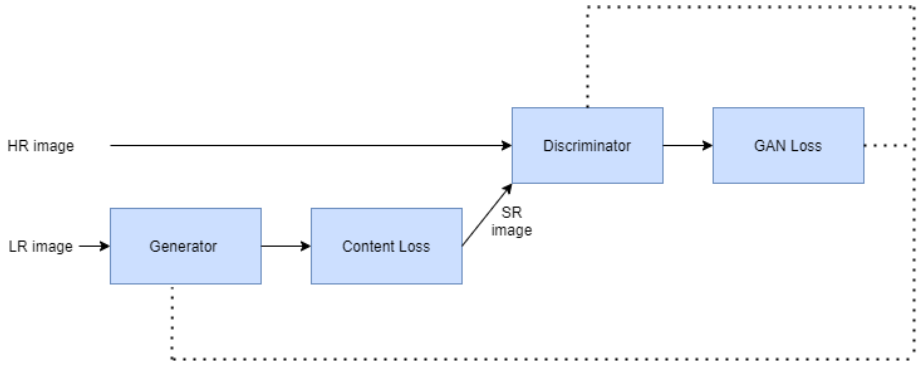


Figure 8: GAN

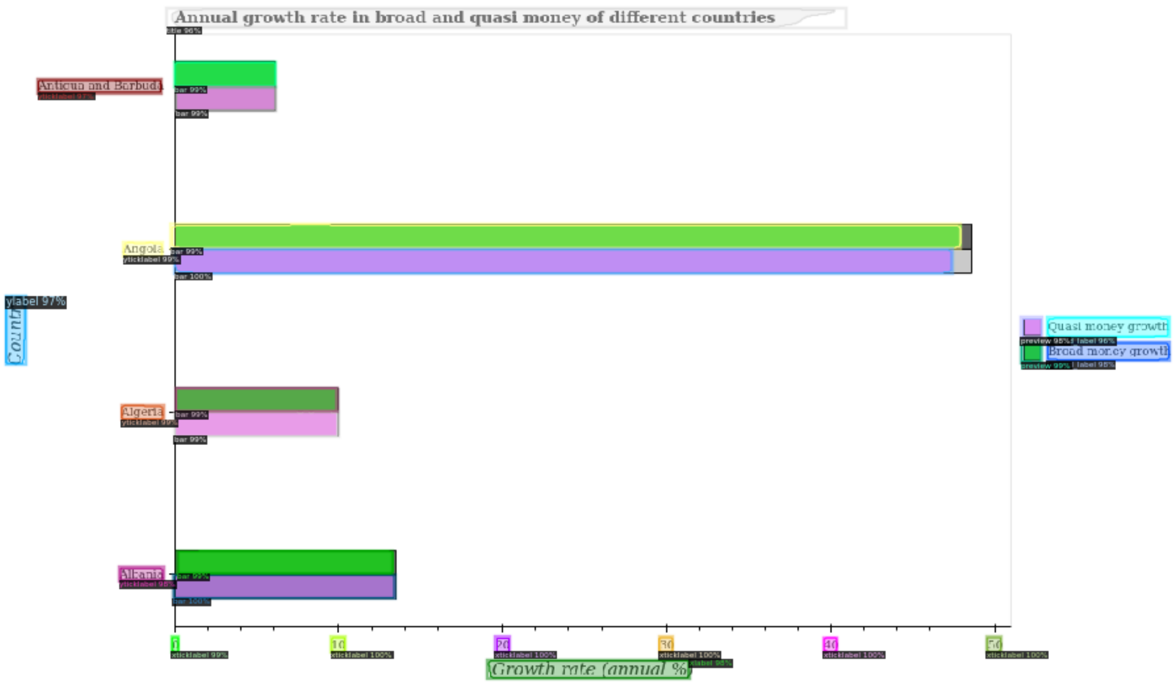


Figure 9: Localisation error - the bar bounding box not accurate on a random sample from PlotQA [11] dataset.

and after applying algorithm 1. We got that the total sum of IoU value (compared to the ground truth) of bar elements of all the images without correcting the bounding box was 210419.52 and a sum of IoU values after correcting with the algorithm was 215246.80. Figure 10 shows Algorithm 2's intermediate edge detection step, whereas Figure 11 represents the bar's perfectly bounded bars for the input Figure 9.

5.2. Text Localization and Recognition

first what u do... then flaws

The textual regions recognition can be evaluated by dividing into *(a)* detection accuracy and *(b)* recognition accuracy.

Detection The detection of text is measured by *(a)* word level accuracy and *(b)*character accuracy. The result of OCR done by Tesseract and is given in Figure 14 along with the

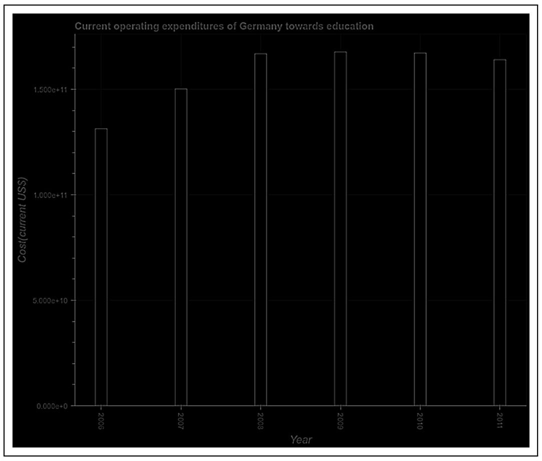


Figure 10: Laplacian output for Figure 9, an intermediate output of Algorithm 2.

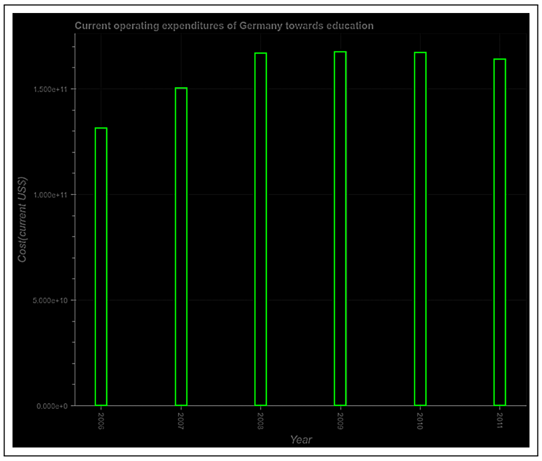


Figure 11: An example for bar detection by contour approximation on synthetic data set

Table 1
IoU values of the corrected bar

	IoU$_{BF}$	IoU$_{AF}$
Image 1	0.893	0.897
Image 2	0.827	0.9482
Image 3	0.829	0.9481
Image 4	0.891	0.9482
Image 5	0.853	0.9981
Image 6	0.910	0.9461

IoU$_{BF}$: $IoU of bar elements before Algm2, IoU_{AF}$: $IoU of bar elements after Algm2$

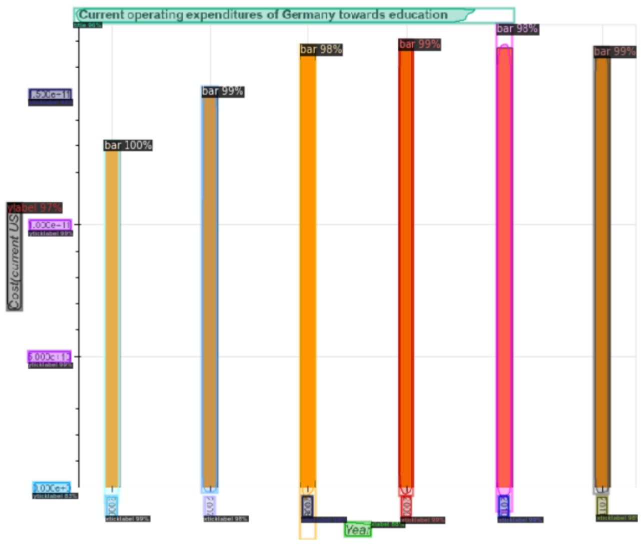


Figure 12: The bounding box proposals generated by proposed method on a random image from PlotQA dataset

textual output and word level accuracy. In this case, the title and the X ticks as horizontal texts is detected with 100 % accuracy, where as, the y axis ticks gives 1 % accuracy.

The detection rate improved by 60 % by using the proposed pre-processing algorithm. The qualitative results are given in Figure 16 and quantitative results are given in Table IV.

The text detection determines whether or not a text is present and where the text instance is located. Figure 12 depicts the qualitative results of the text region proposal. The text localising errors include detecting the text at a different angle than horizontal text. The chart text having a small font size was also missed during detection. The lengthier title compared to the ground truth annotations had localisation issues as given in Figure 13. The text recognition using Tesseract OCR module is suitable for text extraction, it is not accurate as it is not detecting all the x-tick labels correctly.

5.3. Data Extraction

5.4. Table Question Answering

The final step in our method is using the table we have generated to answer the question about the bar chart. Table Question Answering (TQA) is done with the help of a pre-trained model called TAPAS [6].
Input: The table created and the question answer pair.
Output: Chart information.

TAPAS is built on BERT's architecture with tables as input, starting with a successful joint pre-training of text segments and tables. The design of TAPAS model (Figure 5) is based on BERT's encoder but with extra positional embeddings to encode tabular structure.

The table is flattened into a series of words, the words are broken into word pieces (tokens), and the question tokens are concatenated before the table tokens (as in Figure 6). There are two classification layers, one for picking table cells and other for cell aggregation operators. This classification layer selects a subset of the table cells. Depending on the chosen aggregation operator, these cells can be the final answer or the input used to compute the final solution.

5.4.1. Results: Table Question Answering

5.5. Comparison Between Different Models

5.5.1. Qualitative Results

5.5.2. Quantitative Results

5.6. Accuracy of Different Modules

6. Future Works and Applications

7. Conclusion

References

[1] Ritwick Chaudhry et al. "Leaf-qa: Locate, encode & attend for figure question answering". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 3512–3521.

[2] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. "Detecting text in natural scenes with stroke width transform". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 2963–2970.

[3] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[4] Leibin Guan and Jizheng Chu. "Natural scene text detection based on swt, mser and candidate classification". In: *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*. IEEE. 2017, pp. 26–30.

[5] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

Table 2
Table 3: Quantitative comparisons on different methods

	Method 1			Method 2		Method 3	Method 4
Categories	**AP@0.75**	**AP@0.9**	**AP@0.5**	**AP@0.75**	**AP@0.9**	**AP@0.5**	**AP@0.9**
bar	78.531/%	77.998%	96.24/%	77.57/%	47.54/%	40.959/%	15.51/%
x label	78.282/%	79.288/%	99.94/%	89.90/%	33.72/%	47.2/%	4.39/%
ytick label	58.380/%	57.356/%	99.97/%	88.42/%	31.11/%	34.215/%	1.70/%
preview	90.361/%	87.11/%	99.75/%	89.64/%	32.43/%	49.802/%	11.70/%
xtick label	77.944/%	72.445/%	99.84/%	89.90/%	33.72/%	35.372/%	8.08/%
legend label	81.115/%	81.089/%	99.77/%	98.13/%	50.83/%	45.4/%	7.15/%
title	59.900/%	68.970/%	99.91/%	81.41/%	38.86/%	79.5/%	0.02/%
y label	82.150/%	83.976/%	99.84/%	98.86/%	80.53/%	0.00/%	9.59/%

Method 1 : Proposed method, Method 2: Methani et al. [11], Method 3: Fast RCNN [3], Method 4: Yolo V3 [13]

Table 3
Average precision at IOU 0.9

Category	AP	Category	AP
bar	77.998	x_ticklabel	72.445
x_label	79.288	legend_label	81.089
y_ticklabel	57.356	title	68.970
preview	87.11	y_label	83.976

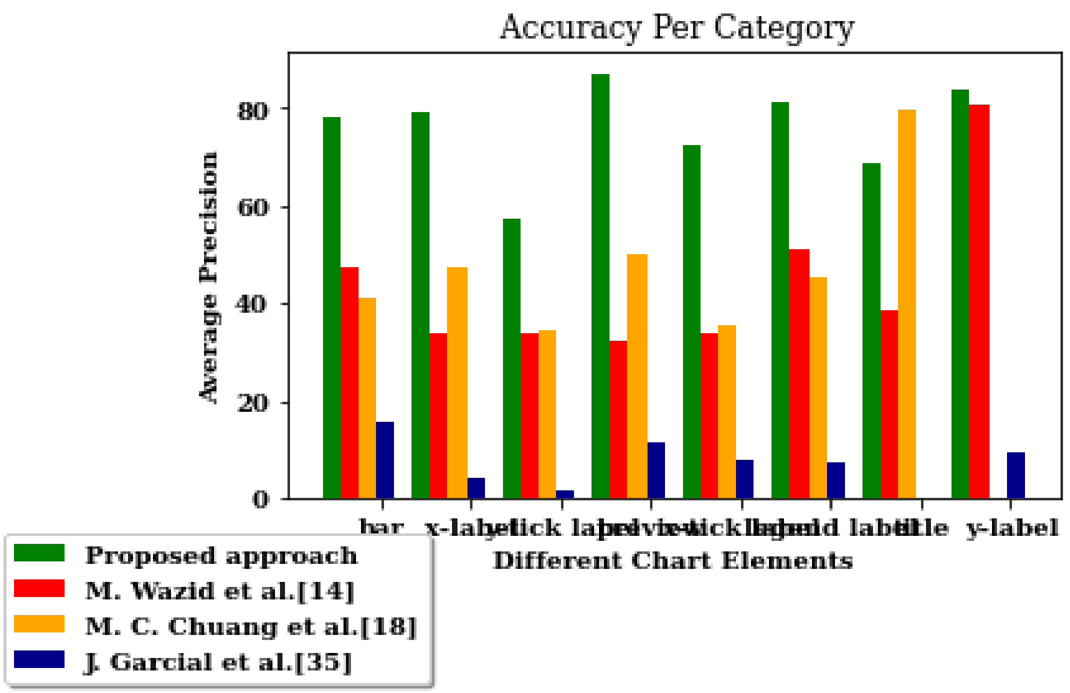
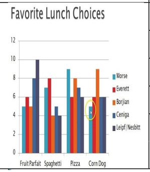


Figure 13: Quantitative precision-recall curves (, middle) and Average Precision (right) of all the four approaches. Clearly, our approach achieves significant improvement compared to others.

Image	Method	Word level Accuracy						Character level Accuracy						Overall Accuracy
		a	b	c	d	e	f	a	b	c	d	e	f	
	SR+ MSER	2/6	5/6	0	0	-	-	20/39	28/33	0	4/20	-	-	
	Bicubic													
	SRGAN													
	ESRGAN													
	Enhanced ESR													

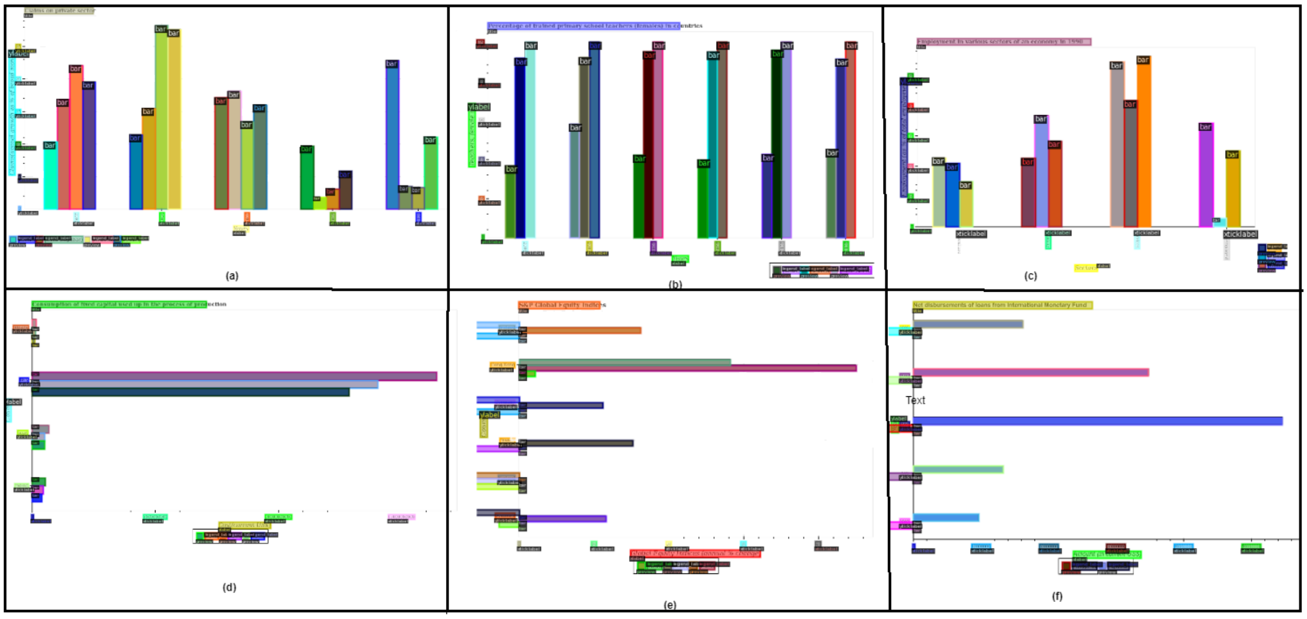
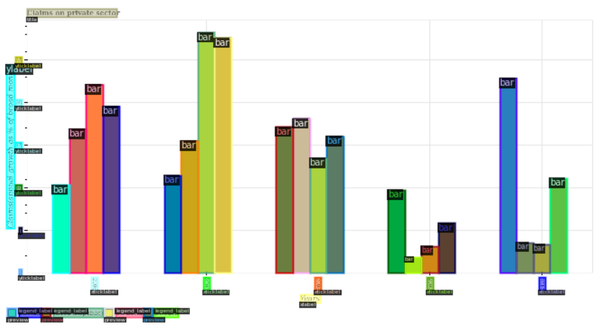


Figure 14: Object detection by MaskRCNN producing the bounding boxes and masks on PlotQA [11] dataset.

0.25
(a)
!htb



0.25
(b)
!htb

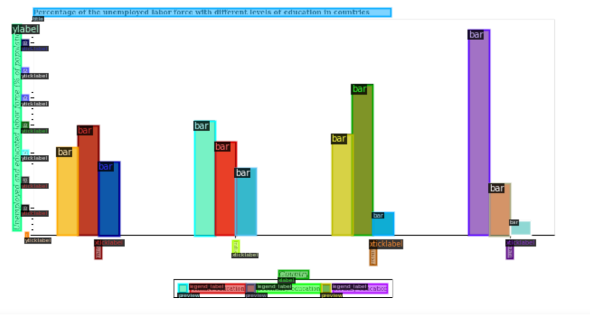


Figure 15: *UPX*: Autoruns.exe, WinAudit.exe, rufus.exe

[6] Jonathan Herzig et al. "TaPas: Weakly supervised table parsing via pre-training". In: *arXiv preprint arXiv:2004.02349* (2020).

[7] Weilin Huang, Yu Qiao, and Xiaoou Tang. "Robust scene text detection with convolution neural network induced mser trees". In: *European conference on computer vision*. Springer. 2014, pp. 497–511.

[8] Kushal Kafle et al. "Dvqa: Understanding data visualizations via question answering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5648–5656.

[9] Samira Ebrahimi Kahou et al. "Figureqa: An annotated figure dataset for visual reasoning". In: *arXiv preprint arXiv:1710.07300* (2017).

[10] Christian Ledig et al. "Photo-realistic single image super-resolution using a generative adversarial network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4681–4690.

[11] Nitesh Methani et al. "Plotqa: Reasoning over scientific plots". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 1527–1536.

[12] Brian G. Schunck Ramesh Jain Rangachar Kasturi. *Machine Vision*. Published by McGraw-Hill, Inc., ISBN 0-07-032018-7, 1995. URL: https://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter5.pdf.

[13] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

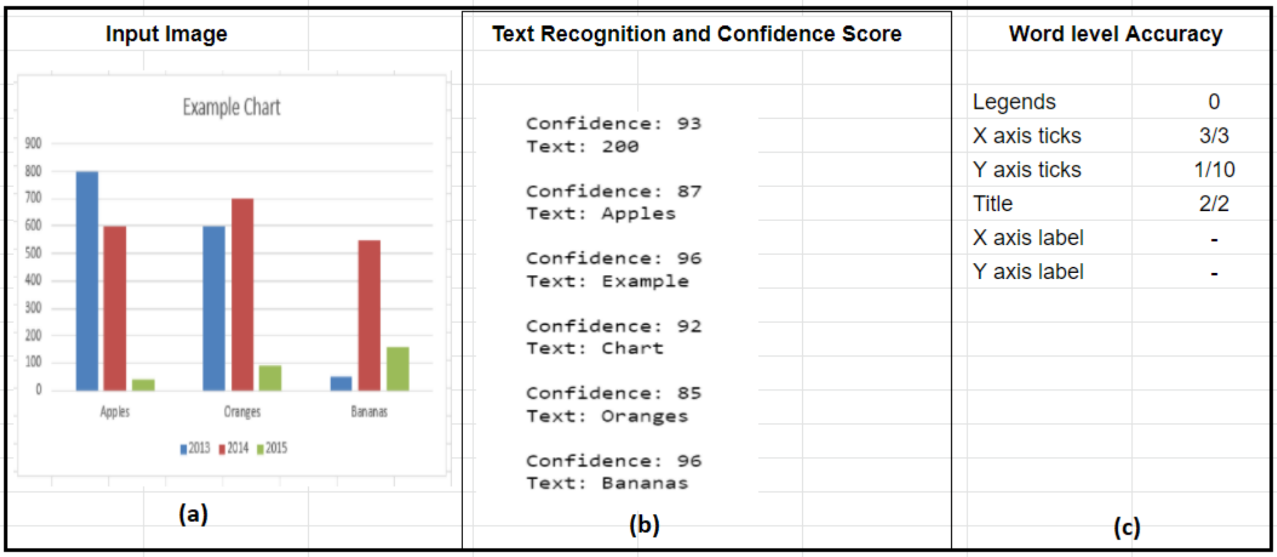
[14] KC Shahira and A Lijiya. "Document Image Classification: Towards Assisting Visually Impaired". In: *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*. IEEE. 2019, pp. 852–857.

[15] KC Shahira and A Lijiya. "Towards Assisting the Visually Impaired: A Review on Techniques for Decoding the Visual Data from Chart Images". In: *IEEE Access* (2021).

[16] Yuyang Wang, Wenjun Ding, and Feng Su. "Super-resolution of text image based on conditional generative adversarial network". In: *Pacific Rim Conference on Multimedia*. Springer. 2018, pp. 270–281.

[17] Ming-Chao Xu, Fei Yin, and Cheng-Lin Liu. "SRR-GAN: Super-Resolution based Recognition with GAN for Low-Resolved Text Images". In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE. 2020, pp. 1–6.

[18] Rabah A Al-Zaidy, Sagnik Ray Choudhury, and C Lee Giles. "Automatic summary generation for scientific data charts". In: *Workshops at the thirtieth aaai conference on artificial intelligence*. 2016.



```
Confidence: 93
Text: 200

Confidence: 87
Text: Apples

Confidence: 96
Text: Example

Confidence: 92
Text: Chart

Confidence: 85
Text: Oranges

Confidence: 96
Text: Bananas
```

Legends	0
X axis ticks	3/3
Y axis ticks	1/10
Title	2/2
X axis label	-
Y axis label	-

Figure 16: An input image and the corresponding text recognition with the confidence score and word level accuracy in order (a,b,c)

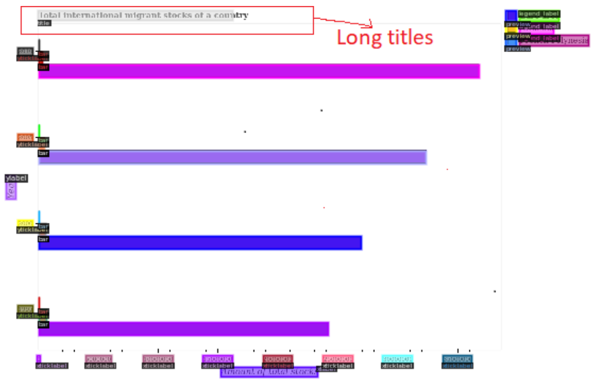


Figure 17: Localisation error for the lengthier titles- a random sample from PlotQA [11] dataset.

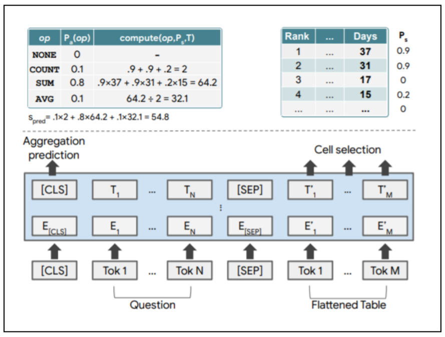
[19] Rabah A Al-Zaidy and C Lee Giles. "Automatic extraction of data from bar charts". In: *Proceedings of the 8th international conference on knowledge capture*. 2015, pp. 1–4.

[20] Yan Ping Zhou and Chew Lim Tan. "Hough technique for bar charts detection and recognition in document images". In: *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*. Vol. 2. IEEE. 2000, pp. 605–608.

Author biography without author photo. Author biography.

Author biography with author photo. Author biography.

Author biography with author photo. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography. Author biography.



op	$P_a(op)$	compute(op, P_s, T)
NONE	0	-
COUNT	0.1	.9 + .9 + .2 = 2
SUM	0.8	.9×37 + .9×31 + .2×15 = 64.2
AVG	0.1	64.2 ÷ 2 = 32.1

s_{pred} = .1×2 + .8×64.2 + .1×32.1 = 54.8

Rank	...	Days	P_s
1	...	**37**	0.9
2	...	**31**	0.9
3	...	**17**	0
4	...	**15**	0.2
...	...	...	0

Figure 18: Architecture of TAPAS [6]