Extracting Information From Images Of Charts

Romil Sakariya
MSc in Computing
Dublin City University
Email: romil.sakariya2@mail.dcu.ie

Khilti Dedhia
MSc in Computing
Dublin City University
Email: khilti.dedhia2@mail.dcu.ie

Abstract*— *Can existing state-of-the-art object detection algorithms produce efficient results to extract information from images of scientific charts that are arguably different from natural images?
In recent years, researchers have extensively developed sophisticated algorithms that can detect objects in natural images. Reasoning over images of scientific charts is a significantly complex challenge owing to the ability of charts to encode data in a visual context to convey and emphasise a particular point or irregularity that would have otherwise gone unnoticed if represented as a table or text paragraph. In our research, we intend to study the methodologies of past researchers in this field, classify chart images based on their type and modify sophisticated algorithms that work on object detection in natural images to extract chart elements (textual and non-textual) from images of charts. Upon examining the results produced by our trained models we can state with certainty that existing object detection algorithms can be used to categorise images of charts by type and extract text and position of chart elements directly from the images of charts. We also discuss our observed results and their implications alongside the future scope of this work.

I. INTRODUCTION

Computer Vision is one of the most significant subsets of Machine Learning because of its vast and absolute applications. Ranging from face detection technology to the detection of real-time objects to navigate self-driving cars, computer vision is experiencing a humongous growth in interest and investment in recent years. Owing to this recent peak in research, Computer Vision has made substantial strides in reasoning and interpreting digital natural images by using sophisticated Machine Learning algorithms and Artificial Intelligence. This huge shift towards obtaining unprecedented results while analysing digital images is largely credited to and accelerated by, two main factors: a category of Machine Learning Algorithms called Artificial Neural Networks and the availability of Big Data alongside the high computational ability of cloud computing. Convolutional Neural Networks (CNNs), a prominent subcategory of Neural Networks, have proved instrumental in analysing complex types of data such as audio, images, and videos. The additional convolution or pooling layers in these algorithms provide the ability to analyse and process non-traditional complex data such as images and videos, while also providing significantly better results.

Whilst digital images of natural objects are complicated to interpret, images of charts pose a considerably more complex challenge owing to the ability of charts to encode information. Reasoning over charts requires complex abilities to extract and interpret information concealed in the position, colour, orientation and type of chart elements embedded in the chart. Charts have been used repeatedly to convey and emphasise a particular point or irregularity that would have otherwise gone unnoticed if represented as a data table or text paragraph. Conversely, interpreting and deducing conclusions from data-dense charts can often be a daunting task, especially when we consider charts that use some sort of referencing legends, convoluted relations, complex patterns, or multiple attributes. Automated extraction and interpretation of information from charts would have vast applications ranging from assisting visually impaired people in understanding interpretations from charts to generating a textual summary of charts.

In this paper we analyse, modify and improve the performance of object detection algorithms in extracting information from images of charts in the PlotQA dataset. We use two state-of-the-art models, YOLOv5 and VGG16, to construct an execution pipeline to perform three tasks: (a) classify images of charts based on their type, (b) extract chart elements (such as bars, lines, etc.) from images of charts and (c) extract chart's textual elements (chart title, legend, etc.) from images of charts. Given our computational limitations, we could only use a randomly chosen subset of our chosen dataset, but our promising results are an indication of how the pipeline would perform over the entire dataset.

In Section 2, we perform an in-depth review of previously conducted research around the field of extracting information from charts. Having conducted this review, we obtain valuable knowledge and in Section 3, we summarize the PlotQA dataset. In Section 4, we describe each stage of our execution pipeline. In section 5, we discuss and analyse the implications of the results obtained by our pipeline. Finally, in Section 6, we build upon our work to discuss possible future research works and summarize our research work. The code files and results pertaining to the work conducted in this paper can be found on our GitLab Repository: https://gitlab.computing.dcu.ie/sakarir2/2022-mcm-chart-info-extraction.

II. Related Work

A. Datasets

The main requirement to build effective systems capable of extracting information from images of charts is a comprehensive dataset of such images of charts. In recent years, owing to the availability of open data, there have been some significant attempts toward assembling a sizable dataset comprising images of charts. An ideal dataset would comprise a sufficiently large number of real-world images of charts, incorporating the different types of charts.

Reasoning over charts generally extracts information that tends to answer some questions about the chart, or the topic represented by the chart. Based on this idea, the recent attempts have focused on compiling a Visual Question Answering (VQA) dataset that comprises (a) a set of images of charts and (b) a set of question-answer pairs based on the information embedded in the charts.

The effort by Kahou et al. (2017) is one of the first teams to assemble such a VQA dataset based on images of charts. Their dataset, FigureQA, consisted of a dataset of synthetically generated images of charts and corresponding question-answer pairs based on said charts. Although FiqureQA made a valiant attempt in compiling a comprehensive dataset, their approach had two major flaws: (a) all their questions were answerable using a binary Yes/No answer and (b) the charts are synthetically generated based on data that are not from the real world. These shortcomings are critical especially because reasoning over charts reveals information that can answer many complex questions that cannot be represented as only binary.

Kafle et al. (2018) tackled this drawback and proposed a dataset, called DVQA, that consisted of questions that can be answered from information extracted directly from charts (such as legend) or from a fixed vocabulary of 1000 words. Although their approach did overcome the primary issue with the FigureQA dataset, since their images of charts were also synthetically generated using data that was not sourced from the real world, their dataset, although better than their predecessors, was also lacking in certain areas.

Chaudhry et al. (2020) took inspiration from these previously built datasets to assemble a massive dataset based on data from the real world. Their dataset, called LEAF-QA, comprises over 250,000 images of charts varying across five types: Bar plot, Pie chart, Dot plot, Line chart, and Box plot. Their question-answer dataset can broadly be divided into two categories: structural questions and relational questions. Unlike its predecessors, LEAF-QA included a few complex relational questions that could not be simply answered from a fixed vocabulary or by extracting a chart element from the image itself. Chaudhry et al. introduced the idea of an automated system that reasons over an image of a chart to answer questions that require knowledge-based analysis of the features of that chart.

A very prominent attempt towards assembling such a comprehensive dataset is performed by Methani et al. (2020). They recently published their dataset called PlotQA which comprises 224,377 images and 28.9 million question-answer pairs corresponding to said charts. Their dataset stands apart from the rest in three major ways: (a) Their charts are based on real-world sourced data, (b) their dataset comprises four different types of images of charts, and (c) a good amount of their questions are not answerable using a fixed vocabulary or cannot be extracted directly from the images of charts. PlotQA has provided researchers an opportunity to build systems that can be deployed in the real world since their dataset is a very close representation of the types of charts that persist in the real world. We discuss in detail the PlotQA dataset in the next section.

B. Methodologies

Based on approaches adopted by past researchers in this field, we can break down the entire process of extracting information from charts into the following steps:

- Type Classification – Categorizing different charts according to the type of chart elements (Dot plot, Bar charts, Pie charts, etc.)
- Feature Recognition – Identifying the key elements present in the chart (Dots, Bars, Lines, etc.)
- Feature Extraction – Extracting the position, size or value of the feature that is recognised.
- Text Extraction – Locating and extracting the various text elements of a chart (Axes labels, Chart title, Legend, etc.)
- Context Analysis – Reasoning over the context of the elements represented in the chart.
- Information Extraction – Representing the extracted information in the form of a table or paragraph.

Researchers have generally followed the above-outlined pipeline with a few adjustments in their strategies. For instance, researchers that have dealt with a wide variety of charts in their work have felt the need to incorporate a step to classify processes based on the type of charts.

Luo et al. (2021) focuses on data extraction from various types of charts, and subsequently, they incorporate suitable steps in their approach to classifying chart features based on chart types. Their proposed framework called ChartOCR uses a modified version of the popular CornerNet [6] object detection framework that generates paired key points of detected objects. Owing to CornerNet's hourglass network, ChartOCR is able to achieve good accuracy in detecting the key elements for a given chart.

Ganguly et al. (2020) emphasised the importance of using algorithms at higher IOU values (around 0.9) while dealing with extracting information from images of charts. They acknowledge the good performance of object detection algorithms such as Fast/Faster R-CNN that produce a decent mAP score (around 80%) while detecting chart elements and text content from images of charts in the PlotQA dataset. Although these results are promising, Ganguly et al. emphasises that while dealing with images of charts, a minor error while locating object positions in the chart can lead to large-scale errors in the numerical aspect of interpretation. Using a higher IOU value ensures the minimization of these errors while detecting visual elements of a chart. But the standard object detection algorithms used by them do not produce good mAP values at higher IOU values. Subsequently, their proposed model, called PlotNet, is built to overcome this shortcoming while also ensuring faster detection speed.

Text extraction is an important subtask for extracting valuable information from charts since once elements such as chart title and axis labels are extracted, it would provide the model contextual information about the data represented in the chart. To extract such textual content from images, researchers have used standard OCR engines such as Microsoft OCR [8] and Tesseract [9]. Boschen et al. (2018) provide an excellent imperial comparison amongst these competitive OCR systems in their survey. Their work provides future researchers with an excellent reference to choose an appropriate OCR engine for their models.

Most researchers have used some form of neural network (convolutional or recurrent) for their extraction processes and have obtained acceptable results in terms of accuracy, but Zhou et al. (2020) proposes an alternative approach. They criticise the use of rule-based approaches since slight variations in chart structures can lead to misleading results. Their proposed model uses an encoder-decoder framework alongside a convolutional neural network to maximize the accuracy of their model. Although their model produces very high accuracy, certain drawbacks in their approach cannot be overlooked - their dataset lacks variation and size.

The most impressive approach we came across was performed by Methani et al. Their proposed model used the PlotQA dataset to reason over images of charts to answer out-of-vocabulary (OOV) questions. This task was not previously tackled by other researchers. They emphasise the need for having an automation system that does more than answer binary questions or questions that can have answers extracted directly from the chart. Building on this premise, they proposed a model that establishes a relationship between chart elements and extracted values. Once a relationship is established, their multi-stage model reconstructs the data table corresponding to the chart which can be used for complex high-level reasoning.

Shahira and Lijiya (2021) conducted an insightful survey that focuses on the importance of extracting information from images from charts from an accessibility point of view. Implementing a good chart information extraction system would have revolutionary implications but none greater than the support that would be available for visually impaired users. In the next section, we utilize the gained knowledge from reviewing literature published by past researchers to choose an appropriate dataset and build our pipeline.

III. PlotQA Dataset

In order to select an appropriate dataset, we shortlisted all available open sources and peer-reviewed datasets of images of charts and ranked them based on their usability, size, relevance, quality, and resemblance with the real world. We learned from our research that for an effective result, any system would require a dataset that comprises a large number of images of different types of charts [12]. Subsequently, we eliminated all options that contained fewer than 150,000 images or were published before 2016. Another key aspect is variation in terms of the types of charts in a dataset since an automated system should not be limited to only one type of chart which would significantly differ from the real world. Based on the abovementioned criteria, we arrived at the following four possible options: PlotQA, DVQA, FigureQA, and LEAF-QA. Table 1 presents a comparative analysis of FigureQA, DVQA and LEAF-QA datasets.

After carefully evaluating the above-mentioned options, we decided to utilize the latest and most relevant dataset for our work: PlotQA. Methani et al. released their dataset called PlotQA which quickly established itself as the new state-of-the-art VQA dataset based on images of charts.

Methani et al. published their dataset consisting of 224,377 images and 28.9 million question-answer pairs on their Git repository. They split the dataset into Train, Validation, and Test subsets in a 70:15:15 proportion. Each subset includes three files: a folder consisting of the actual images of charts, a JSON file containing the annotations of the images, and a JSON file consisting of the question-answer pairs based on the images in the subset. PlotQA dataset consists of images categorised across four types of charts: Horizontal Bar Chart (33.50%), Vertical Bar Chart (33.50%), Dot Plot (16.50%), and Line Chart (16.50%).

A minor drawback of this massive cumulative effort by Methani et al. is the lack of structural information about the JSON files containing the annotations and QA pairs. The semi-structured dataset for annotations is a complex hierarchical structure that required a significant effort to navigate and analyse. It would behoove future researchers if some sort of nomenclature of the semi-structured annotations file would be provided alongside the actual dataset.

While exploring training data from the PlotQA dataset we observed a discrepancy regarding the width of bounding boxes for chart titles. For long chart titles, the bounding box width is lesser than the actual content of the title.

Dataset	Dataset Features	Dataset Limitations
FigureQA	1) Consists of around 180,000 images of charts spread across five different types: line plots, dot-line plots, vertical and horizontal bar graphs, and pie charts. 2) Includes over two million question-answer pairs based on the images of charts.	1) All the questions in this dataset are binary Yes/No type which is a major drawback of the FigureQA dataset. 2) The chart images are synthetically generated based on data that is not from the real world.
DVQA	1) Comprises over 300,000 images of charts that were generated from synthetic data. 2) Ensured availability of questions that require non-binary answers. 3) Good proportion of their questions could only be answered from a combination of words from a fixed set of vocabulary.	1) Questions do not require complex reasoning to answer. 2) Answers could be easily put together from the fixed set of words or directly extracted from the chart's image.
LEAF-QA	1) Consists of over 250,000 images of charts varying across five types: Bar plot, Pie chart, Dot plot, Line chart, and box plot. 2) Images of charts are based on open-source data from various public databases. 3) Include a few complex relational questions that could not be simply answered from a fixed vocabulary or by extracting a chart element from the image itself.	1) Proportion of complex questions was very small in comparison to other questions.

TABLE I: Comparative analysis of available VQA datasets based on images of charts.

Figure 1 illustrates the ground truth bounding box of the title for a couple of images. Upon visual inspection, the inaccuracy is evident. This inaccuracy is also factored into the results obtained after training the model and is further discussed in Section 5.

While exploring the annotations file, we understood the structure of the dataset and mapped out the hierarchy of the dataset. For each image in the dataset, there is a corresponding entry in the annotations list. The relationship between the image and the entry in the annotations list is established using the name of the image in the dataset. The images are named numerically, and the numbers are correspondingly represented in the annotation entry for that image.

The primary elements of a chart such as bars for bar charts, dots for scatter plots, and lines for line charts are represented separately in an independent section of the annotation element for a chart. This section consists of various bits of information about the chart elements such as the bounding box information, colour representation, name of the element, etc. The bounding box information is provided in standard format with one floating point entry each for the top left x-coordinate and y-coordinate along with the width and height of the bounding box.

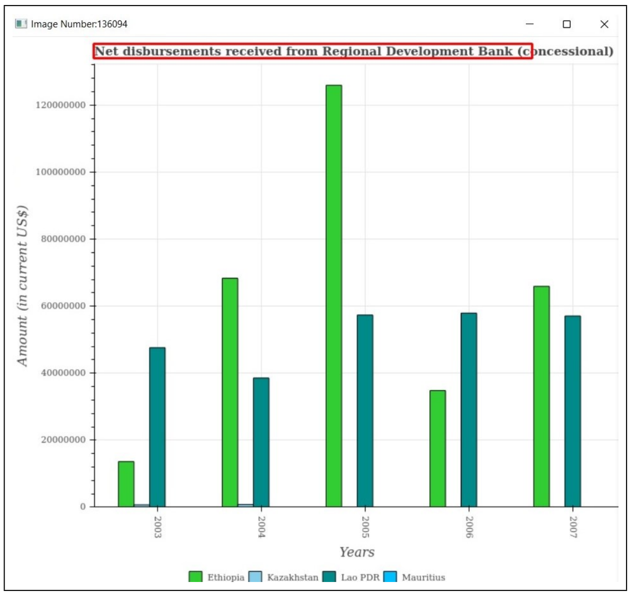


Fig. 1: Discrepancy with ground truth bounding boxes of the chart title in PlotQA dataset. The bounding box width of chart titles is not correctly recorded for almost all images in the PlotQA dataset.

The general figure information for a chart is presented in another section in the annotations' element for an image. This section includes various key pieces of information pertaining to the chart. Information about the legend, axes, title, and gridlines is provided in this section along with corresponding names, colour representation, and bounding box coordinates. This section enables the automated system to grasp a complete understanding of the chart and the information represented in the chart. The principal purpose of this section is to provide additional knowledge about the chart enabling the system to reason over these images of charts. In absence of this section, it would be an uphill task to accomplish sound reasoning and understanding of the contextual information of the chart. Another section of the annotations' element provides information on the type of chart represented in the image. Figure 2 illustrate the basic structure of the annotations data.

The question-answer pairs data is straightforward. Each item in the data comprises an image index that establishes a relationship between the item and a corresponding image in the dataset. Alongside this image index, there are individual attributes that contain a question ID, the question string, an answer ID, the answer string, and bounding box information about the answer in case the answer is extracted from the image.

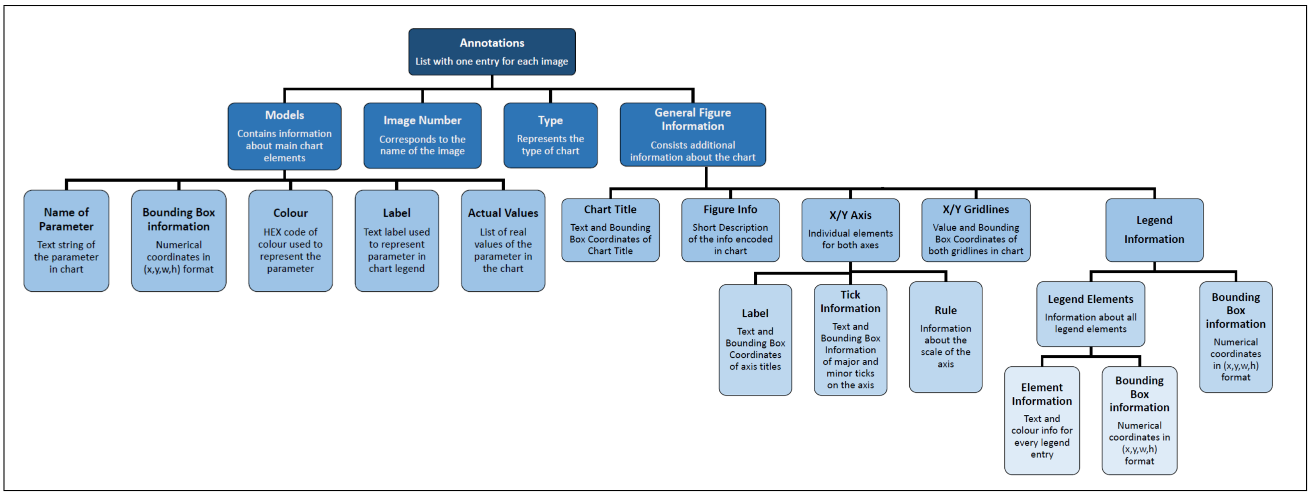

Annotations
List with one entry for each image
Models
Contains information about main chart elements
Image Number
Corresponds to the name of the image
Type
Represents the type of chart
General Figure Information
Consists additional information about the chart
Name of Parameter
Text string of the parameter in chart
Bounding Box information
Numerical coordinates in (x,y,w,h) format
Colour
HEX code of colour used to represent the parameter
Label
Text label used to represent parameter in chart legend
Actual Values
List of real values of the parameter in the chart
Chart Title
Text and Bounding Box Coordinates of Chart Title
Figure Info
Short Description of the info encoded in chart
X/Y Axis
Individual elements for both axes
X/Y Gridlines
Value and Bounding Box Coordinates of both gridlines in chart
Legend Information
Label
Text and Bounding Box Coordinates of axis titles
Tick Information
Text and Bounding Box Information of major and minor ticks on the axis
Rule
Information about the scale of the axis
Legend Elements
Information about all legend elements
Bounding Box information
Numerical coordinates in (x,y,w,h) format
Element Information
Text and colour info for every legend entry
Bounding Box information
Numerical coordinates in (x,y,w,h) format

Fig. 2: Hierarchy of the annotations file in the PlotQA dataset.

IV. Proposed Pipeline

We propose an execution pipeline consisting of three stages: (a) Classification of images of charts based on the chart type, (b) Extraction of chart elements from images of charts and (c) Extraction of textual elements from images of charts. In the following subsections, we describe the execution of each of these stages.

A. Stage 1: Classification of Chart Images by Type of Chart

Classification of chart images based on their type would be the first step towards approaching the main problem of extracting information from the charts. For this purpose, we chose the VGG-16 model (Visual Geometry Group) [13], object detection and classification model, to act as a classifier with 4 classes: Dot Plot, Horizontal Bar Chart, Line Chart, and Vertical Bar Chart. The VGG-16 model is a convolutional neural network that uses 13 convolutional layers and 3 fully connected layers, adding up to 16 layers. It can classify millions of images of size 224 X 224, up to 1000 common object categories such as a mouse, keyboard, and many other animals. It is the best model to perform transfer learning where the last layer can be modified according to the personalized classes.

We have used the pre-trained version of the VGG-16 model imported from the KERAS library in Python and performed transfer learning. Owing to the low local computational capacity, we conducted all our experiments on a subset of the original dataset. For training, we chose two subsets of 5000 randomly chosen images: one subset had the proportion of chart types replicated the original dataset (unbalanced) and the other subset had equally distributed images of each chart type (balanced). The original unbalanced subset had a split of 16.50:33.50:16.50:33.50 of the types of charts viz. dot plot, horizontal bar chart, line chart, and vertical bar chart, whereas the balanced subset dataset had a 25% split across each chart type. The dataset was further split into train and validation sets in an 80:20 ratio respectively. After creating the required subsets, we ran some preprocessing steps to resize all the images uniformly. Then, we froze all the convolutional layers and the first 2 fully connected layers of the model, modifying the last fully connected layer of the model from 1000 classes to the 4 defined classes, and trained the model to classify the images to their respective types.

Initially, we trained an experimental model on a subset of just 200 images to prove the concept of classification and analyse the performance of the model. With 12 epochs and a batch size of 32 images set as default, our model performed average with an accuracy of 52.5% and a loss of 1.25. Increasing the epochs to 20, the accuracy went up to 82.5% and the loss dropped to 0.836 which indicated a direct proportionality between the number of epochs and accuracy. Hence, we further increased our epochs to 25 and were flabbergasted to notice the accuracy dropping down to 17.5% and later discovered it was causing overfitting in the model. We realized the number of epochs is a key parameter with regards to the better performance of our model.

Another important parameter is the batch size of images per epoch. We further trained our experimental model at two different batch sizes of 20 and 32 images keeping the epochs constant at 20. Noticeably, the accuracy of the model with 20 images per epoch was 94.17% and a loss of 0.1351 whereas with 32 images per epoch was 95.83% and a loss of 0.1065. This proves that although a smaller batch size leaves more execution memory available, a larger batch size improves the performance of the model. Experimentally, we deduced the performance of our model at 12 epochs with a batch size of 32 images per epoch. We discuss the performance results of our actual model and their implications in the next section.

B. Stage 2: Extraction of Chart Elements from Images of Charts

After categorizing the charts based on their types, we moved on to the next stage of our pipeline: extracting key chart elements from the images. To accomplish this task, it is vital for the system to understand, learn and comprehend the type of chart in the image and the elements of that type of chart. For this stage, we chose the YOLOv5 model. YOLO (You Only Look Once) [14] is a state-of-the-art computer vision algorithm that detects common objects in natural images and real-time videos. It predicts the bounding box of various defined classes and gives class probabilities simultaneously. YOLO's architecture has 24 convolutional layers and 2 fully connected layers at the end. It divides the images into N grids of equal dimension and each grid is responsible for the detection of the object and generation of bounding boxes of that class in the images. These boxes are then weighted by the predicted probabilities. YOLO has many versions, the latest being version 5, the fastest, open source and most accurate version of all. Owing to these benefits, we chose the YOLOv5 model for our pipeline.

We began by using an open source, and pre-trained YOLOv5 model from Ultralytics[1] and cloned their repository on Google Drive. The YOLOv5 model requires images alongside their class labels and bounding box annotations in a specific format as an input. The class labels and bounding box annotations need to be provided in the '.txt' file and the name of the file should correspond to the name of the image. The format of the labels and annotations is: [class_label, x_coordinate of the bounding box, y_coordinate of the bounding box, width of bounding box, height of bounding box]. If there are multiple instances of a class in the image, multiple annotation entries can be provided in the corresponding text file. YOLOv5 uses a normalized central coordinate system wherein the bounding box coordinates are calculated considering the size of the image. Since these values are normalized, all the bounding box coordinates range between 0 and 1. The model also requires a '.yaml' file defining the total number of classes and their numeric representations such as 0 corresponding to Dot Plot, 1 to Horizontal Bar Chart, 2 to Line Cart, 3 to Vertical Bar Chart, and 4 to Legend. These integer class values are used in the text file to categorise the annotations for each type of element in the chart image.

Having explored the annotations data file in the PlotQA dataset in the previous section, we were able to compile an extraction script to fetch the coordinates of bounding boxes for all our desired elements in a chart image. The extraction script created '.txt' files corresponding to the images that were used for training the model. Before creating the files, we needed to transform the bounding box coordinates from standard integer format to the normalized format as required by the YOLOv5 model. We also performed some cleaning steps such as: ensuring the dimensions of a bounding do not exceed the size of the image, none of the coordinates are negative, etc. These compilation, transformation, and cleaning steps ensure the integrity of the data used for training the model which also translated to the trustworthiness of the obtained results.

The Intersection of Union is a metric that measures the amount of overlap between the predicted bounding box and the actual ground truth bounding box. The IoU value can be adjusted to obtain more precise bounding boxes depending on the requirement of the classification model. Researchers have emphasised the use of higher IoU values while extracting chart elements from images of charts since minor adjustments of bounding boxes for a chart element can alter the value of that element in a misleading manner. Adhering to this practice, we trained our YOLOv5 model at an IoU value of 0.9. This ensured the model understood the importance of maximum overlap between predicted bounding boxes over ground-truth bounding boxes. To comparatively analyse the implications of using a higher IoU value, we also trained models at an IoU value of 0.5. We discuss the performance and results of both models in the next section.

C. Stage 3: Text Extraction from Chart Images using Tesseract OCR

Textual elements in charts convey vital information regarding the relationships between variables that are represented in the chart. Extracting textual elements such as chart title, legend elements, and axes labels can provide much-needed information for an algorithm to gain an understanding of the chart. These textual elements lay the groundwork for contextual understanding and finally higher-level reasoning of chart images.

This task can be broken down into two subtasks: (a) Locating textual elements in the chart image and (b) Extracting text from the identified elements. To identify chart elements that contain text, we modified a pre-trained YOLOv5 model. We used an extraction script to fetch ground truth bounding boxes from the annotations dataset for the training images. For the purpose of this task, we trained our model to identify four elements in a chart image: (1) Chart Title, (2) X-Axis Label, (3) Y-Axis Label, and (4) Legend Elements. In our previous stage, we had identified the whole legend in a chart image but for the purposes of text extraction, we would require locating each entry in the legend to ensure the reliability of the extracted information. We ensured the bounding box coordinates were converted into the YOLOv5 model's required normalized format. We also ensured assigning appropriate classes to each bounding box coordinate in the corresponding '.txt' file of every image in the training subset. These four elements constitute almost all the text content in a chart and therefore would provide valuable information about the data represented in the chart.

[1]Ultralytics github repo: https://github.com/ultralytics/yolov5

We trained two versions of our modified YOLOv5 model at IoU values of 0.5 and 0.8. Observing the difference in the predictions made by each model would lead to a productive analysis highlighting the importance of choosing an appropriate IoU value. Given our computational limitations, we were able to train the model on a subset of only 2500 images chosen randomly from the training dataset of PlotQA. This training subset was balanced across all four types of chart images present in the PlotQA dataset. Using a validation subset, we were able to generate evaluation metrics for our trained model. Once trained, we generated model predictions for locating chart textual elements for a test subset consisting of 375 images. Both models performed decently in predicting bounding boxes for the textual elements of all the images in the test subset. As expected, the model trained at a higher IoU value predicted bounding boxes that are tighter than those predicted by the model trained at an IoU value of 0.5.

To extract the actual text from the predicted bounding boxes of these textual chart elements, we used a state-of-the-art open-source OCR software called Tesseract OCR [9]. We cropped the textual chart elements to their predicted bounding boxes and converted them into binary images and corrected the orientation if required. We then passed it into the Tesseract OCR module which extracted and returned the text present in the image. We built a compilation script that used this extracted text from the OCR and predicted class labels from the YOLOv5 model to assemble a dictionary format. We discuss the results obtained from this stage in the next section.

V. Results and Discussions

Having established the execution strategy of each stage in the previous section, in this section we present the results obtained from various stages of our proposed pipeline and discuss the implications of these obtained results.

A. Stage 1: Classification of Chart Images by Type of Chart

To accomplish chart type classification, we trained a VGG-16 model on a subset of 5000 images chosen at random from the training dataset. As mentioned in the previous section, we generated a balanced version of this training subset and an unbalanced version. Table 2 describes the training performance of our model on both the balanced and unbalanced subsets.

Model trained on balanced subset		Model trained on unbalanced subset	
Accuracy	Loss	Accuracy	Loss
99.4%	0.0126	99.3%	0.0543

TABLE II: Performance of our VGG-16 classifier on two different training subsets.

In order to comparatively analyse the effect of balanced or unbalanced class of the number of images in the training subset. These test images were chosen at random from the PlotQA testing dataset. We also included some images of different types of charts that were not part of the PlotQA dataset (such as spatial maps, heatmap, and pie charts) just to observe the shortcomings of the model. Upon examining the performance of both the models (one trained on a balanced subset of images and another on an unbalanced subset), we observed that the model trained on a balanced subset was better in terms of accuracy but by a very narrow margin. Both models were able to correctly classify over 99% of the images in the testing subset. The unfamiliar charts are categorised as vertical bar charts since the model is not trained to predict classes for them correctly.

After further examination, we observed that both models have some difficulty in classifying certain types of line charts. When the line/lines in a line chart are parallel to the horizontal x-axis, the trained model was uncertain regarding the classification of the chart. Although it did classify these line plots correctly, the probability value was around 0.6 to 0.8. A possible explanation for this behaviour is the abundance of white background space available in these charts. Figure 3 depicts a line chart that was misclassified as a dot plot with a probability of around 0.6 by our model. The line in the chart is clearly almost parallel to the x-axis. This phenomenon leaves a significant amount of white background area unused. This situation frequently arises in dot plots and therefore the model finds it difficult to make a confident prediction for such an image. A detailed analysis sheet of the testing subset is uploaded to our GitLab repository.

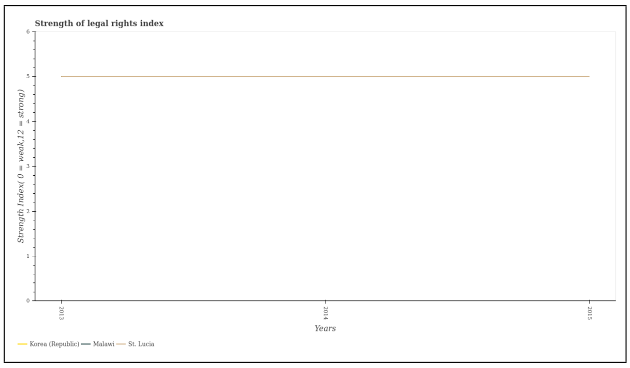


Fig. 3: An example of a line chart incorrectly classified as a dot plot by our model. As observed, the line element is almost perfectly parallel to the horizontal axis which misleads the model.

B. Stage 2: Extraction of Chart Elements from Images of Charts

Given our computational limitations we were able to train a model on a subset of only 2500 images from the PlotQA dataset. We randomly selected two versions of this subset: one having an equal ratio of all chart types (balanced) and another consisting of different types of chart images in the original ratio present in the PlotQA dataset (unbalanced).

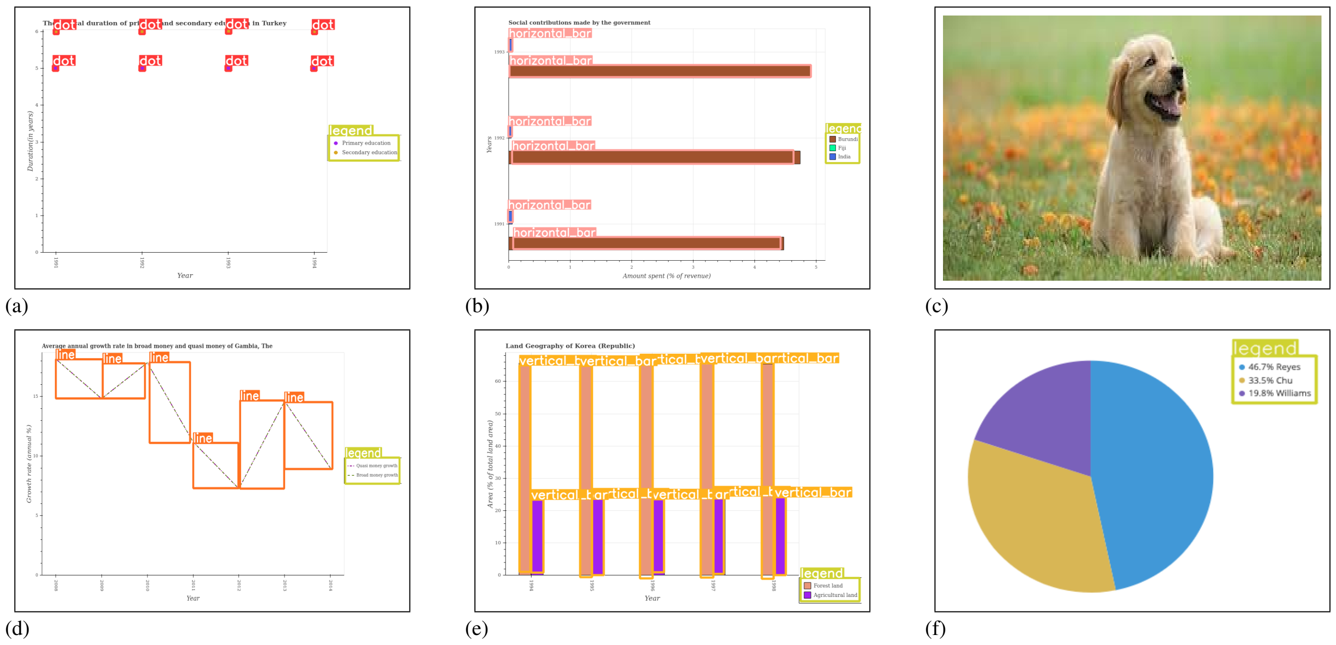


Fig. 4: Tested images on our trained YOLOv5 chart extraction model: (a) Dot plot from PlotQA, (b) Horizontal bar chart from PlotQA, (c) Natural image containing dog, (d) Line chart from PlotQA, (e) Vertical Bar chart from PlotQA, (f) Synthetically created pie chart (not in PlotQA). As observed, the model was able to extract chart elements from (a), (b), (d), and (e) since it is trained on images from PlotQA.

Having two varieties of subsets enabled us to understand the effect of balance in classes. We created the randomly extracted training subset of images, extracted the required annotations from the PlotQA annotations file using our script, and modified the YAML file to indicate the classes for the model. We then began training the YOLOv5 model over 10 epochs and a batch size of 32 images.

Once trained, we validated the model's performance using a validation subset consisting of randomly chosen images (15 percent of the number of images used in the training subset). The validation process generated some evaluation metrics like a confusion matrix, Precision vs Recall curve, etc. Mean Average Precision (mAP) is widely used to contemplate the performance of a machine learning model and it is calculated using the Precision vs Recall curve. Table 3 depicts the mAP values of all the different versions of models we trained.

Although the mAP values are acceptable, we inspected the tested images visually. The predicted bounding boxes for the testing images are drawn over the images and a visual inspection provides a good understanding of the performance of the model. Our trained model performed very well in extracting chart elements in the test images at either IoU values.

[2]Given our computational limitations, we could only train the model over 10 epochs (underfit). We reckon the performance would improve if the epochs were increased by a reasonable amount.

Classes	Model trained on a balanced subset		Model trained on an unbalanced subset	
	Average Precision @0.5	Average Precision @0.9[2]	Average Precision @0.5	Average Precision @0.9[2]
Chart Legend	99.5%	81.3%	99.5%	75.4%
Dot	95.4%	8.9%	96.4%	10.3%
Line	68.7%	29.1%	67.7%	26.9%
Horizontal Bar	91.1%	25.2%	87.3%	22.6%
Vertical Bar	93.2%	51.0%	94.6%	62.8%
mAP	89.6%	39.1%	89.1%	39.6%

TABLE III: Performance of our modified YOLOv5 model in extracting chart elements from chart images.

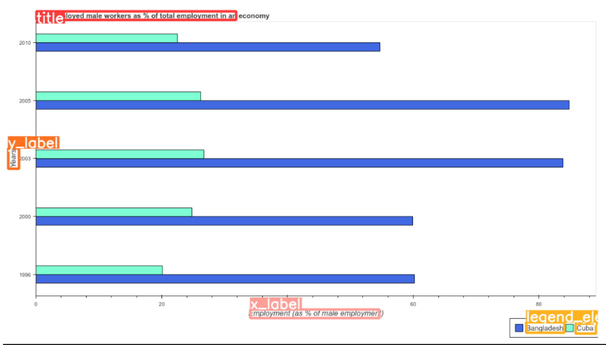
As expected, at the higher IoU value the predicted bounding boxes were tighter than the predictions made at a lower IoU value. This minor change in IoU makes a significant difference overall since the bounding box dimensions are later used to extract the graphical value of chart elements. Figure 4 shows an example of each chart type from the test subset. As seen in the images, the model efficiently predicts bounding boxes around all the chart elements in the image. All the generated evaluation metrics for each model can be found in our Gitlab repository.

C. Text Extraction from Chart Images using Tesseract OCR

Our modified YOLOv5 model performs well in predicting bounding boxes for textual chart elements in a chart image. Table 4 depicts the average precision values for each predicted class for our model. One noticeable discrepancy is visually observed in the width of the chart title predictions. This is an expected outcome since a similar discrepancy exists in the original ground truth training annotations as discussed in section 3. Passing a cropped binary image for each bounding box to the OCR module enables great performance in text extraction for all elements. Our compilation script works on both these previous subtasks to create a dictionary (hierarchical structure) as shown in Figure 5. We exported this dictionary to a JSON file which can be found in our Gitlab repository.

Classes	Average Precision @0.5	Average Precision @0.8[3]
Chart Title	99.5%	37.8%
X-axis label	99.5%	93.8%
Y-axis label	99.5%	88.3%
Legend Elements	99.4%	87.9%
mAP	99.5%	77.0%

TABLE IV: Performance of our model in predicting bounding boxes for textual chart elements.



(a)

```
{
    'image_index': 5980,
    'chart_title': 'oe a Pe sloved male workers as of total employment in ai',
    'x_axis_label': 'umployment as Percentage of male empioymer',
    'y_axis_label': 'Yearg',
    'legend_items': ['Cuba', '3angladesh']
}
```

(b)

Fig. 5: (a) Predicted textual elements from our model and (b) extracted text from our OCR module. As observed from the model's predictions in (a), the chart title bounding box discrepancy in PlotQA dataset is inhereted by the model.

VI. Conclusion and Future Work

In our paper, we highlighted the complexity and significance of extracting information from images of scientific plots. We extensively reviewed published literature by past researchers to understand their conducted work and contribution to the field. After comparing some published VQA datasets based on chart images, we proceeded with the PlotQA dataset. Our intricate exploratory analysis of the PlotQA dataset reveals a discrepancy in their ground-truth bounding boxes for chart titles in their images. We proposed and implemented an execution pipeline to accomplish three tasks that are essential to extract information from images of scientific plots: (a) Classification of images of charts by chart type, (b) Extraction of Chart Elements from the images, and (c) Extraction of Textual Chart Components embedded in the chart images. These three stages of our pipeline generate a valuable understanding of the entire extraction process from charts. We used modified versions of two object detection models, VGG16 and YOLOv5, at various stages of our pipeline. We also performed an experimental analysis based on different subsets (balanced and unbalanced) to observe the effect on the final model's prediction.

There is a large scope of future research that can take advantage and build upon our work conducted in this paper. An obvious next step following our contribution would be a stage that brings together all the outputs from our previous stages to recreate the tabular form of the data embedded in the chart image. This stage would require text or value extraction of tick values on both the axes and checking for bounding box coordinates that coincide with the coordinates of these ticks. This step can be easily accomplished by the model we built in our third stage for extracting textual chart elements from chart images. Once a representative semi-structured table is created for a chart, subsequent neural networks can be built to use the table along with the image to answer questions regarding the data represented in the chart image. This stage is vital to obtain a higher level of understanding of the context embedded in the image of a chart. Although the future scope does seem promising, there are still significant challenges with regard to increasing the accuracy of proposed solutions. Our obtained results prove that existing object detection models can be modified to extract information from images of scientific plots.

[3]Given our computational limitations, we could only train the model over 10 epochs. We reckon the performance would improve if the epochs were increased by a reasonable amount.

References

[1] S. E. Kahou, V. Michalski, A. Atkinson, A. Kadar, A. Trischler, and Y. Bengio, "FigureQA: An annotated figure dataset for visual reasoning," *arXiv [cs.CV]*, 2017.

[2] K. Kafle, B. Price, S. Cohen, and C. Kanan, "DVQA: Understanding data visualizations via question answering," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5648–5656.

[3] R. Chaudhry, S. Shekhar, U. Gupta, P. Maneriker, P. Bansal, and A. Joshi, "LEAF-QA: Locate, encode attend for figure question answering," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 3501–3510.

[4] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar, "PlotQA: Reasoning over scientific plots," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 1516–1525.

[5] J. Luo, Z. Li, J. Wang, and C.-Y. Lin, "ChartOCR: Data extraction from charts images via a deep hybrid framework," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 1916–1924.

[6] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," *arXiv [cs.CV]*, pp. 734–750, 2018.

[7] P. Ganguly, N. Methani, M. M. Khapra, and P. Kumar, "A systematic evaluation of object detection networks for scientific plots," *arXiv [cs.CV]*, no. 2, pp. 1379–1387, 2020.

[8] "Computer Vision," *Microsoft.com*. [Online]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/. [Accessed: 28-Jul-2022].

[9] R. Smith, "An overview of the tesseract OCR engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, 2007, vol. 2, pp. 629–633.

[10] F. Böschen, T. Beck, and A. Scherp, "Survey and empirical comparison of different approaches for text extraction from scholarly figures," *Multimed. Tools Appl.*, vol. 77, no. 22, pp. 29475–29505, 2018.

[11] F. Zhou et al., "Reverse-engineering bar charts using neural networks," *arXiv [cs.CV]*, 2020.

[12] K. Davila, S. Setlur, D. Doermann, B. U. Kota, and V. Govindaraju, "Chart mining: A survey of methods for automated chart analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 3799–3819, 2021.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv [cs.CV]*, 2014.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[15] K. C. Shahira and A. Lijiya, "Towards assisting the visually impaired: A review on techniques for decoding the visual data from chart images," *IEEE Access*, vol. 9, pp. 52926–52943, 2021.