# AUTOMATIC SUMMARY GENERATION BY DATA EXTRACTION OF SUNBURST CHARTS

Hinguralage Lahiru Pradeep Ariyasinghe
218520U

Master of Science in Artificial Intelligence

Department of Computational Mathematics
Faculty of Information Technology

University of Moratuwa
Sri Lanka

April 2024

# AUTOMATIC SUMMARY GENERATION BY DATA EXTRACTION OF SUNBURST CHARTS

Hinguralage Lahiru Pradeep Ariyasinghe

218520U

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree.

Master of Science in Artificial Intelligence

Department of Computational Mathematics

Faculty of Information Technology

University of Moratuwa

Sri Lanka

May 2024

# DECLARATION

I declare that this is my own work, and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date: 28/04/2024

The above candidate has carried out research for the PhD/MPhil/Master's thesis/dissertation under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of Supervisor: Dr Thushari Silva

Signature of the Supervisor: Date: 28/04/2024

# DEDICATION

The research thesis is dedicated wholeheartedly to my beloved wife and parents who inspired, motivated and supported me throughout the study.

# ACKNOWLEDGEMENT

I would like to thank my academic and research supervisor, Dr Thushari Silva, for her generous support and assistance. Also, I want to express my deep appreciation to Dr. Sagara Sumathipala., for their professional support and supervision during this study.

I am thankful to the academic staff at the Faculty of IT, Department of Computational Mathematics, University of Moratuwa, for their guidance and knowledge during our undergraduate years. Also, members of my family, our friends and batch mates who have supported us in numerous ways.

# ABSTRACT

Sunburst charts are useful for visualizing hierarchical data structures. However, they can be challenging to interpret because of their nested hierarchy and circular layout. Viewers must determine the color and size and of each segment and navigate through various hierarchy levels, which can be confusing, especially with many or unevenly distributed categories.

To address these challenges, this research proposes a method to extract summary data from sunburst charts, enabling quick identification of key trends and patterns. This approach is particularly beneficial for business analysts and marketers who need to analyze large datasets efficiently. By automating the extraction process, analysts can focus their efforts on areas of significance, saving time and improving efficiency.

Existing literature indicates a gap in tools for extracting data and generating summaries of sunburst charts. To fill this gap, the author implemented a system that leverages Machine Learning (ViT models), image processing, optical character recognition (OCR), and LLM. While several methods could have been used for this project, the chosen technologies offer a robust solution for data extraction and summary generation from sunburst charts.



**Keywords**: ViT Models, NLP, OCR, Machine Learning

# TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction

This project aims to find a solution to get data from sunburst charts using Machine Learning (ViT Models), computer vision and NLP technologies to make short summaries. This chapter explains the problem, why the idea is new, and how the research will be done.

## 1.2 Prolegomena

Charts are commonly used in digital publications like research papers and web pages to showcase data [2]. Understanding charts automatically requires the ability to extract chart data automatically. But it's challenging to define strict rules because charts come in various visualizations, such as bar charts, line charts, and pie charts [1]. To further study or make charts easier to understand when raw data isn't available, extracting data from chart images is necessary [2]. This requires in-depth research in chart data reading. Existing systems have proposed various designs using machine learning and computer vision, but many rely on datasets that are not accessible to the research community [3].

The volume of scientific and electronic data has grown significantly in recent decades, partly due to the effectiveness of visual representations [4]. This data often includes multimodal documents or information graphics. However, charts are often not well explained in text and lack contextual comments, making it challenging for screen readers to interpret them for visually impaired individuals. Therefore, information retrieval techniques are needed to analyze chart images and extract relevant information [4].

While bar charts and other charts are typically used for visually summarizing data, this study is on pie charts because of their various formatting options and types. Pie charts also differentiate into many types depending on user requirements and different purposes, with the complex pie chart, also named as the sunburst chart, we can identify as being one of these subtypes [6].

Decision-making and Data analysis increasingly rely on data visualizations like line graphs, bar charts and other data representation techniques. Analyzing data often means answering tough questions that need calculations and logical reasoning, especially with numerical data [13]. Summarizing numerical data using conversational language can be challenging. The target audience for such summaries includes individuals in marketing and IT, researchers, students, and visually impaired individuals [4]. Linguistic research suggests using protoforms to simplify the creation of different types of sentences [5].

## 1.3 Problem Definition

Currently, I couldn’t find any method implemented to directly generate a summary from sunburst charts. So, it’s important to clarify, this system takes the sunburst chart images as input. And it is possible to generate a relevant summary by detecting and extracting

data from the input chart images using ML, computer vision and NLP techniques. After extensive research, it has been found that there have very limited approaches have research and developed to the extraction of chart data from sunburst charts [15]. As a result, there are currently no systems , tools and applications, or specifically developed to summarize the data from sunburst charts while also generating a summary.

### 1.3.1 Problem Identified

There I couldn’t to find fully automatic data extraction and summarization application for data charts, as most of chart summarization applications are semi-automatic. Additionally, there have no tools available specifically for extracting data and generate summary from sunburst charts. If you're looking for such tools, you might need to explore custom solutions or consider developing one yourself based on existing techniques and libraries for data extraction and summarization.

## 1.4 Background and Motivation

The motivation for researching automatic sunburst chart summarization is to tackle challenges in understanding and extracting valuable insights from complex hierarchical data. Sunburst charts normally represent hierarchical levels and their relationships, but manually extracting the information from them can be error-prone and labor-intensive. The aim of this research is to design and develop automated application that can generate coherent and concise summaries from advanced pie charts. This automation will enable efficient and relevant data exploration, support decision-making processes and enhance data understanding. By automating the data extraction and summarization, this project target to provide comprehensive insights into advanced data patterns, facilitating informed decision-making and analysis to the end users.

## 1.5 Gaps in Research

Other researchers who conducted this study identified differences between existing studies on data extraction from data graphs, specifically focusing on collecting data from sunset graphs. Although previous studies have focused on the analysis of other types of graphs, such as heat maps and waffle graphs, the processing and evaluation of sunburst graphs is important in this regard. Bajiä and Job (2022) were the first to present new results in this field, and according to their findings, this presents an important gap to fill for the design, development and evaluation of the process, especially considering the role of sunburst charts in many general applications. Including business, marketing and IT.

## 1.6 Contribution to the Body of Knowledge

### 1.6.1 Technological Contribution

Data extraction is about getting information from various sources for storage or processing. First, the information is brought into a middle system. It's then changed and maybe given more details before being sent to the next part of the data process, as Mishra et al. explain.

Before Bajić and Job introduced their innovative automated process, no strategy had been implemented to read data from sunburst charts. Their system takes a raster(jpg/png) image as input and structure the image as an object-oriented structure for the chart. But their strategy does not consist of generating a summary. Our goal is to do research and develop an application which can read features from circular-shaped(pie) charts, like sunburst charts, and to improve features extraction and understand through a summary. Additionally, the researcher has compiled a dataset including of 500 chart images.

### 1.6.2 Domain Contributions

Data representation involves displaying data visually, such as in graphs or maps, to help understand and analyze it. The primary goal is to make it easier to identify patterns, trends, and anomalies in large datasets [13].

Charts and plots are useful for conveying dataset information, but they're normally don't directly contribute to prior research and only provided as images. Sometimes, the images can be examined without access to the underlying data, so information extraction from images may be necessary [6].

The research demonstrates the benefits of an automated data extraction system across various industries, including healthcare, finance, and scientific research. The system is developed to extract data from sunburst charts, which are created to display advanced data. This system could reduce the effort, increase the accuracy and reduce the time needed for manual data extraction, also enhancing consistency of information extracted. It could lead to faster and more effective data processing, significantly impacting the fields of science, medicine, and finance.

## 1.7 Research Challenges

Automatically extracting features from sunburst charts are challenging for mainly two reasons. Initially, the ad hoc connections among components are typically missed in conventional image processing techniques. Secondly, different chart types require different algorithms and processing models. For example, pie charts and sunburst present data and features extraction challenges due to their advanced and multi-layered design. Moreover, researchers have noted difficulties in reading sunburst charts because of the small size of letters, words, symbols, and percentages in each section. The challenge of extracting features from sunburst charts is exacerbated by a lack of original chart datasets and the available datasets are presence with noisy datasets. Also, Sunburst charts are very less common than line charts, bar charts and other pie chart types, which are more widely used. This emphasizes the needful of more researches towards this challenges into what end users prefer and need.

## 1.8 Aims

The aim of this research project is to develop a complete system for Automatic summary generation by data extraction of sunburst charts.

## 1.8 Objectives

Following objectives have been identified to develop a bilge alarm monitoring system for warships using fuzzy logic.

- A critical review of the evolution of sunburst chart summarization methods and related systems developed.
- In-depth study of technologies used in modern image processing and NLP systems.
- Design and implement an automatic summary generation system by data extraction of sunburst charts.
- Evaluate the automatic summary generation system by data extraction of sunburst charts.

## 1.9 The Novelty of the Research

Summarizing sunburst charts by extracting their features presents a unique challenge due to the intricate nature of hierarchical data and the circular format of the visualization they depict. We can see several developed conventional strategies for creating such summaries and understanding data, of this area of research existing. Challenges include identifying hierarchies, segments, other features and relationships, deciding which details to include, and ensuring an understandable and concise summary. Machine learning models, and NLP techniques together offer an approach to this problem.

Generating summaries of sunburst charts by extracting features provides a new way to automatically understand advance data of the charts. By automatically identifying and extracting key details from sunburst charts, this proposed approach make possible users to identify trends and patterns in their context quickly. This can be especially helpful for huge image datasets or at that time, the accuracy is limited. Moreover, generating a summary of sunburst charts has not been attempted previously, making this approach both innovative and promising.

## 1.10 Project Scope

Here I am going to achieve the following scope,

- User is able to of input the images of sunburst charts as PNG / JPEGs.
- A system capable of generating the summary of the sunburst charts. For that, the system is able to be analyzing percentages of each segments in each hierarchies in the sunburst chart.

## 1.11 Chapter Summary

This chapter serves as an introduction to the topic, providing essential background information and summarizing the relevant area. It also proposes a research plan to fill the gap and address the research problem within the framework of the presented research project.

# CHAPTER 2

# LITERATURE REVIEW ON EXISTING CHART SUMMARIZATION SYSTEMS

## 2.1 Introduction

This chapter will review existing research in the field, comparing different algorithms and methodologies. It will also examine current models that integrate data visualization, image processing, and other techniques to identify and fill any gaps in research. The goal is to offer a comprehensive understanding of these subjects.

## 2.2 Problem Domain

### 2.2.1 Data Visualization

Data visualization provides valuable tools for any field dealing with complex and extensive data, offering creative ways to present information. The development of computer graphics has played a significant role in the evolution of modern visualization techniques [25]. Visualizations utilize our visual perception, which is adept at quickly identifying patterns and anomalies, to enhance our understanding of the data. Assessing the effectiveness of a visualization in achieving this goal is a crucial objective in visualization research, with practical implications [25]. Information visualizations often involve data representation, where the main aim is to link data values with graphical representations. Visualization designers use fundamental graphical elements called visual encodings to translate data into graphical form.

Creating effective visualizations often requires users to be knowing with both their data and the visualization and analyzing tools. Ideally, people would have access to applications that giving visualizations automatically, enabling users to select the most suitable ones. However, this is challenging, if not impossible, because there is no widely accepted way to measure how closely a visualization mimics human experience [1].

### 2.2.2 Objects and texts extraction from charts by machine learning

Machine learning research has developed and has advanced significantly, particularly with the development of the Machine learning models. Deep learning is inspired by the intricate architecture of the human brain, processing information through multiple layers of abstraction. These techniques enable the direct extraction of feature representations, allowing systems to extract complex learn from raw images without the need for manually extracting features [1]. Users can experiment with deep architectures to automatically extract information from data at various levels of abstraction. Recent studies have shown the successful application of deep learning techniques in various domains, including image and video classification, voice recognition, visual tracking, and NLP [1].

To overcome the problems of extracting features from charts, many researchers have turned to deep neural networks. For chart image representations (bar charts), one approach involves using a generic object detection method, considering a one bar as a separate object to including its own components. A researcher suggests developing a rotation algorithm to get the data for circular charts.

Even though deep machine leaning approaches have greatly improved efficiency, they still struggle with certain types of charts. For example, the method they use to detect bounding boxes in bar charts doesn't work as well for other chart formats like line charts.

Moreover, machine leaning approaches find it difficult to get improved results such as plot area information and data range [1]. Existing tools that extract information from charts using old computer vision techniques are less useful because they are rely on advance, human-oriented criteria [15].

### 2.2.3 Features Extraction and Generating Summary from charts.

Feature extraction is considered vital for identifying pattern categorization by identifying important information in a pattern. This technique is widely used in tasks like image classification and recognition. Its main objectives include identifying the more critical related information in the input image data and representing it in a lower-dimensional space [4].

## 2.3 Existing Systems

### 2.3.1 Data extraction and generate the summary.

Analyzing charts includes several activities, like extracting data and categorizing different types of charts. However, these processes encounter problems and challenges. For instance, the Bar chart analyzer categorization model may struggle with bar charts that have patterns or hollow bars, and it may have difficulty extracting the chart image in unique test and noisy situations. Sometimes, it may fail to recognize text placed in hand-drawn charts or negative bars etc. [6].

In contrast, this method utilizes both low-level data and extracted texts to identify the different types of pie chart. This innovative approach has resulted in the creation of a new feature set that achieves an average categorization accuracy of 96% [5]. Furthermore, WebPlotDigitizer, another technology, provides both automated and manual data extraction processes from the provided chart images [6].

Machine learning models have proven effective in tackling object identification, classification and other challenges in chart reading. For example, ChartSense and FigureSeer (Siegel et al., 2016) used CNN for chart classification. Moreover, ChartSense applies ML model to categorize different varieties of charts, while uses fine-tuning methods by FigureSeer. Text extraction is vital for data extraction and chart inference, this has been accomplished by separately detecting graphical components and text sections [5].

ChartOCR introduces a Hybrid system that utilizes key objects extracting techniques for detecting chart components. This method streamlines the extraction of segments from data charts and can be used with different graphic styles. Rather than directly detecting object bounding boxes, it focuses on identifying significant main points of target text objects. This method is shown to be flexible in detecting various chart items

and eliminates the wants to create different networks for each item [1]. However, Chart OCR is currently restricted to bar charts.

NBless and Liu, Klabjan (2019) developed a deep learning approach for extracting features from pie charts, line and bar charts. That approach consists of three stages: which are the chart type detection, components detection, and inference. There they used a one deep-learning method to find pertinent data from all chart types. In the second stage, which involves text recognition, element detection and bounding box matching, they retrieve the original numerical information. Integrating relevant network components and their details is significant for matching objects such as bars with x-axis and y-axis values and segments of a chart with parts of the legend.

## 2.4 Technological review

### 2.4.1 Machine Learning methods

Many studies have utilized applications of neural networks to tackle the challenges of extracting features from charts. Conventional object identifying methods, used to identify objects in common images with simple patterns, have been adapted to bar and line charts to detect individual bars and lines as their features. An alternative method, put forth by a researcher, entails applying the feature rotation method and also recurrent networks to take out the circular shape data from pie charts. While machine learning has made significant strides in improving time efficiency, but researchers still face limitations in ability to handle different types of charts. As an example, the method of bounding box detection used in bar charts does not readily apply to other chart formats, such as line charts. Moreover, deep learning and machine learning approaches face difficulties in learning good results such as data range, plot area information, etc. [15].

Chart classification is a process as same as image classification. Studies done by Prasad et al. (2007), they outlined an image vision related method for classifying chart images into five types of categories. They are using Support Vector Machines for classification. And histograms of Oriented Gradients and Scale Invariant Feature Transform descriptors for extraction. Savva et al. (2011) designed and developed another system that combines three tasks that categorization, details extraction, and chart redesign. This system uses Support Vector Machines (SVMs) to extract low-level visual characteristics for chart classification. For object detection tasks, Faster R-CNN generally provides more accurate results compared to YOLO and SSD, despite requiring a longer training period.

Several OCR models implemented for text detection, some of those applications developed by Shi and Belongie (2017), Tian and Qiao (2016). Tian et al. (2016) used kind of an anchor box method to predict and identify text bounding boxes. Shi et al. (2017) introduce the segment-linking method, it is having the ability of handling-oriented text detection.

Conventional computer vision methods have been the focus of numerous studies aimed at identifying chart components.

Zhou et al. (2000) used the Hough transform and boundary tracing to identify bars and Huang et al. (2007) used rules and edge maps like approaches for chart details identification. In 2011, Savva et al., used some conventional methods like color details of pixels and shape were used to detect pies or bars. However, ML approaches have the ability to automatically identify chart components and texts within an image.

The next task is data reconstruction, after features are identified. which includes numerical data identification and text extraction. A text recognition technique (OCR techniques) is used to inspect each bounding box which including text. In addition, tick labels are matched according to the coordinate locations to predict the relationship of coordinate-value. These relationships provide the vital features combining with an input image chart and incorporate other graphical components in the process.

### 2.4.2 Rule based method.

Feature-based algorithms have been the primary approach to addressing the challenge of extracting chart elements. These algorithms employ edge extraction and color continuity searching to locate the raw materials, followed by the elimination of incorrect candidates using pre-defined rules. But these rule-based methods mainly rely on predefined characteristics and developer-made rules [1].

While effective for data with specific styles, these algorithms lack generalizability across other types of data. For instance, a criterion designed for extracting vertically aligned bars may struggle to identify horizontally bars. Some methods suggest that users correct and reduce errors to mitigate this problem through user knowledge. However, even though incorporating user input may enhance performances, also significantly increases the time cost [1].

### 2.4.3 Extracting texts

The Tesseract OCR is used to identify numerical values and nominal both within each text object. Sometimes, text regions containing special characters like punctuation marks, empty spaces may be omitted from the output.

Tesseract OCR is well-regarded for its efficiency and effectiveness in text detection.

It carries out texts admirably with scanned images featuring a one-color background, uniform color and compatible typeface. On the other hand, Tesseract OCR does not achieve optimal performance when dealing with images that exhibit different font sizes, diverse color schemes, rotated, curved texts, or mixed languages. This may also confuse with image noises, interferences, other matters with the text, like less resolution, blur, variable lighting, out-of-focus segments, etc. These challenges possible to present in chart graphics including numeral characters, small text sizes, variations in formatting, and a blurry view.

## 2.5 Evaluation

All the revision existing systems, like extracting features from sunburst charts, need evaluation to judge algorithm’s efficiency and effectiveness. Evaluating these algorithms involves different metrics and methods that affect the conclusions. There are two main ways to evaluate: quantitative and qualitative.

Qualitative evaluations entail subjective perceivable evaluation by human experts, most of the times involving annotated images or studies by user. These methods offer valuable insights into the usability and user experience aspects of the algorithm.

Quantitative techniques provide objective evaluations using numbers. These evaluations use metrics like accuracy, precision, recall, F1-score, and confusion matrix to measure how well the algorithm identifies and extracts data from sunburst charts.

When evaluating data extraction from sunburst charts, certain parameters are essential. These may include the accuracy of extracting hierarchical data, the ability to handle different chart styles and complexities, and the efficiency of the extraction process.

## 2.6 Chapter Summary

This chapter used to identify the most effective research and developments for addressing the issue at hand. This associated with creating a graph to visualize potential solutions, studying existing systems to understand their strengths and limitations, and reviewing previous work to gather relevant findings that could inform the development of a new approach.

# CHAPTER 3

# TECHNOLGY ADOPTED IN CHART SUMMARIZATION SYSTEM

## 3.1 Introduction

To achieve the project's goals, a variety of innovative technologies had to be learned and adapted. One such technology is the Python-based OpenCV library, which was used for image processing. This chapter focuses on the technological requirements and evaluates the convenience and appropriateness of using these technologies to implement the proposed solutions.

## 3.2 Vision Transformer (ViT) Models

Transformers were introduced in 2017 in the well-known paper called “Attention is all you need”. Transformers change the entire landscape of NLP. In ViTs, we pre-process things before they are fed into the Vision Transformer. And rather than with Bert and other language Transformers that consume word or sub word tokens Vision Transformer consumes image patches. first, we split an image into image patches two we process those patches through a linear projection layer to get our initial patch embeddings then we pre-append something called a class embedding to those patch embeddings and finally we sum the patch embeddings, and it called positional embedding now there's a lot of parallels with this process.

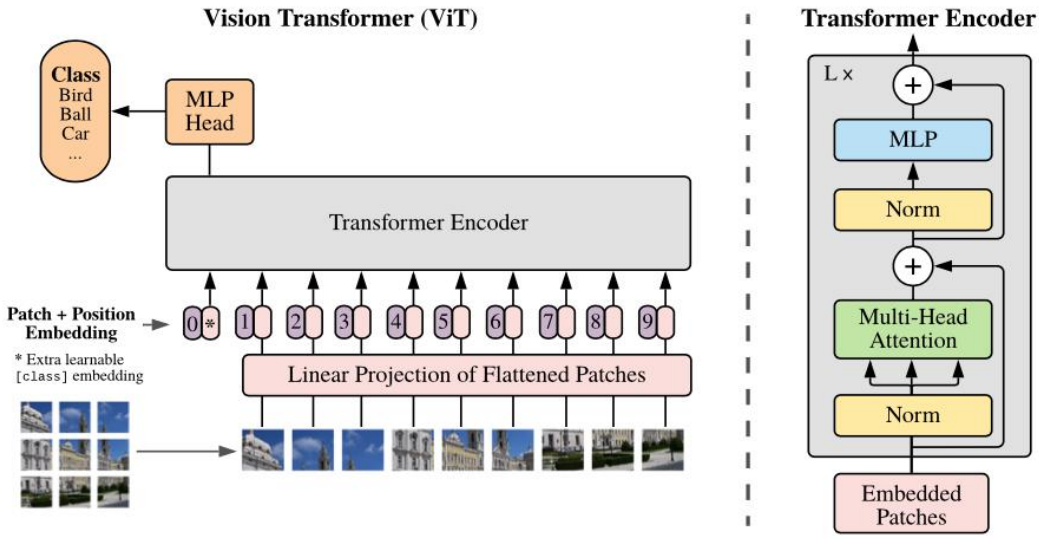



*Figure 3.1: Process of the ViT Model*

## 3.3 Optical Character Recognition (OCR)

OCR is a application in computer vision that identify and recognize text object in images. Those issues I explained in Literature review in details. Because of that, can’t use conventional OpenCV OCR techniques in our case. So, we want to overcome and increase the accuracy of those results. For that I chose RoboFlow OCR API is powered

by DocTR to get results in our case. That is a pre-built machine learning model to use OCR tasks.

## 3.4 Generative Pre-training Transformer (GPT)

GPT is a set of deep learning-based models we used for different NLP tasks such as question answering, text summarization, etc. It's a set of pre-built applications that we can use with minimum supervision. In our case the summarization task is achieve by Open AI, "gpt-3.5-turbo-instruct" model to achieve the summarization task at last.

## 3.5 Python

Python is an object-oriented scripting, interpreted, interactive and high-level language. English keywords are used most of the times whereas other programming languages use punctuation. It has very less syntactical constructions than other programming languages. As well as Python is mainly using programming language utilize for web developments, software development, etc. Main advantages of python are it is emphasis on code readability and programmers can express anything within few lines of codes. As well as OpenCV library has been written in Python and C++. Many neural network libraries have been written based on python. For an example TensorFlow which was introduced by Google was developed using python.

## 3.6 Version Controlling System

GIT was used as the version controlling system where private repositories were created in Bit bucket from Atlassian.

## 3.7 Open-Source Computer Vision Library (OpenCV)

Open-Source Computer Vision Library (OpenCV) is a image processing software and open-source computer vision library. In image processing projects this is mainly relying on image processing. It is an open-source computer vision library that can be perform in many different platforms(portable). OpenCV aids the research in performing various processing activities. The main reason behind selecting OpenCV over MATLAB was the efficiency in performance. The IDE used in carrying out the development of the system is Microsoft Visual Studio 2017.

## 3.8 Microsoft Visual Studio

Microsoft Visual Studio, created by Microsoft, is a comprehensive integrated development environment (IDE) used for programming various applications, including computer programs, web apps, websites, mobile apps, and web services. Visual Studio supports over 36 programming languages, enabling developers to write managed code and native code efficiently. The IDE features a robust code debugger and editor that supports nearly every programming language, along with language-specific services.

## 3.9 Chapter Summary

This chapter's goal to introduce the used technologies for addressing the issue. It useful for make an clear idea graph, moreover studying existing systems, and doing a review of previous works for indeed findings.

# CHAPTER 4

# APPROACH OF SUNBURST CHART SUMMARIZATION SYSTEM

## 4.1 Introduction

This chapter describes what type of approaches used by us in brief. This chapter discusses all basic modules and their all functionalities with input and output. This system consists of five sub modules which are appropriate for sunburst chart summarization main module.

## 4.2 Hypothesis

Here we hypothesize that ML, Image Processing and NLP can be used to summarize the sunburst charts. This inspiration for the hypothesis has come from the huge volume of literature evidence for the use of previous research and developments.

## 4.3 Input

Only the input here is the sunburst chart image from the user. The system supports all images in JPG/ PNG formats.

## 4.4 Output

Based on the input image stated above in the input, the system is generated a summary of the given sunburst chart image as a paragraph. That output describes the hierarchies and the contained segments, percentages of them.

## 4.5 Process

The system is designed to giving the solution for the existing problem is divided mainly for 4 modules.

- Image Preprocessing
- Circles Detection
- Title, Text and Segment Objects Detection
- Calculate Percentages and Colors
- Texts Extraction
- Summary Generation

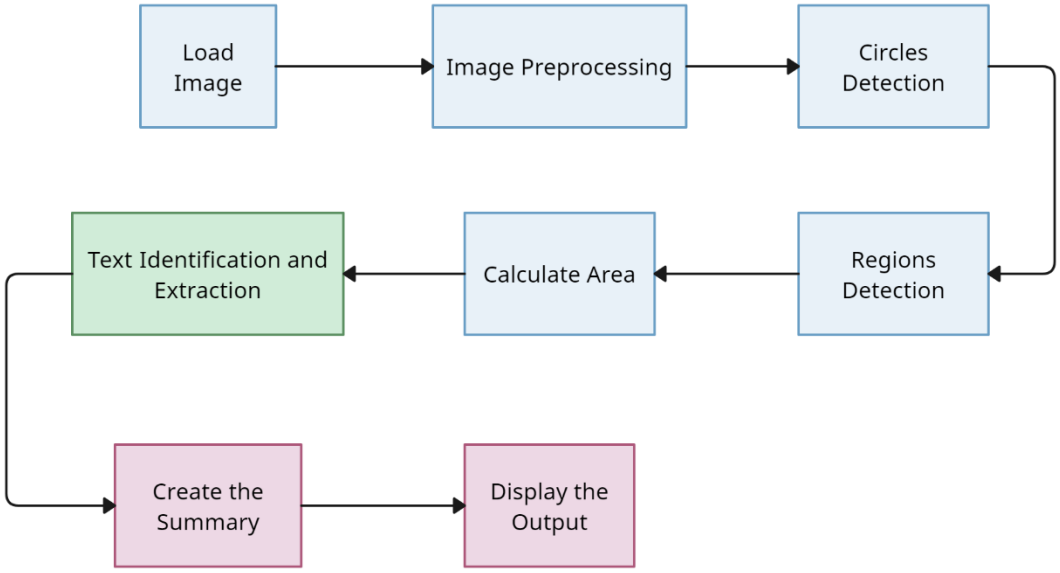



*Figure 4.2: Process of the sunburst chart summarization*

## 4.6 Users

Users of the research outcome come from different fields. The ultimate end users of the industry will be the knowledge in statistics who are the business-related people and they will get profit in their businesses using this system.

## 4.7 Summary

This chapter discussed briefly on the module inputs, outputs, solution and the flow of the process we followed. Next chapter showing the design of the whole system along with the detailed explanation of the modules how implemented individually.

# CHAPTER 5

# DESIGN OF SUNBURST CHART SUMMARIZATION SYSTEM

## 5.1 Introduction

This section is giving a detailed and visual representation of the design of the system and architecture. It will include low-level, high-level diagrams to give a pure idea of the implemented system's design.

## 5.2 Design Goals

| Goal | Description |
|---|---|
| Performance | The system should also prioritize efficiency to ensure that users receive the results promptly without unnecessary delays. |
| Accuracy | The primary goal of this project is to develop a system that can accurately generate a summary by extracting data from sunburst charts. Since the accuracy of each extraction step directly impacts the final result, ensuring high accuracy is crucial. |
| Maintainability | The end product should be designed to be user-friendly and easy to maintain for both users and the development team. This includes providing clear documentation, well-organized code, and intuitive interfaces to facilitate ongoing maintenance and updates. |
| Reusability | Components of the project can be reused to facilitate future improvements, ensuring that valuable work is not duplicated and that the project can evolve time efficiently. |

*Table 5.1: Designing Goals*

## 5.3 System Architecture Design (High-level)

### 5.3.1 Design Diagram

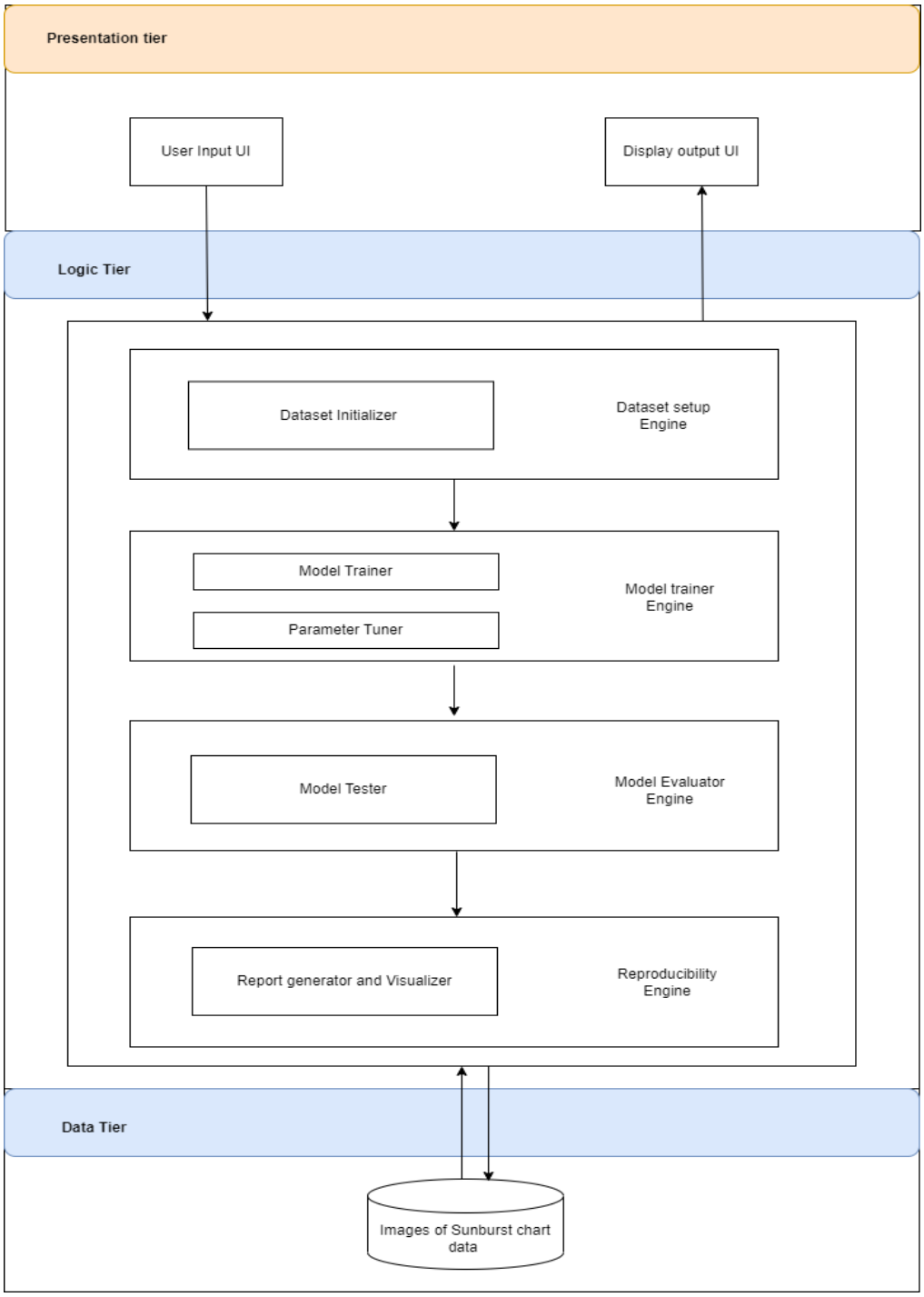



*Figure 5.1: Architecture Diagram*

In this process of summarizing data from sunburst chart images, a three layers design diagram could help organize the features as follows:

1. **Presentation Tier**: This layer would display the sunburst chart to users and handle interactions with the application layer for data retrieval and summary presentation.
2. **Logic Tier**: Responsible for extracting and processing data from the sunburst chart to generate a summary. It interacts with the data storage layer and prepares the dataset, trains models, and evaluates them.
3. **Data Tier**: Stores raw data from the sunburst chart and any additional data needed for the summary. It is use by the logic tier to extract necessary information.

## 5.4 System Design

### 5.4.1 Methodology of designed system

The system will use the Structured System Analysis and Design (SSAD) methodology instead of Object-Oriented Design (OOD). This choice is made because the system has many components that need to manage functions and methods effectively. SSAD is considered the best option because its principles come from objects, existing scenarios and making so easier to understand.

## 5.5 Design Diagrams

### 5.5.1 Activity Diagram

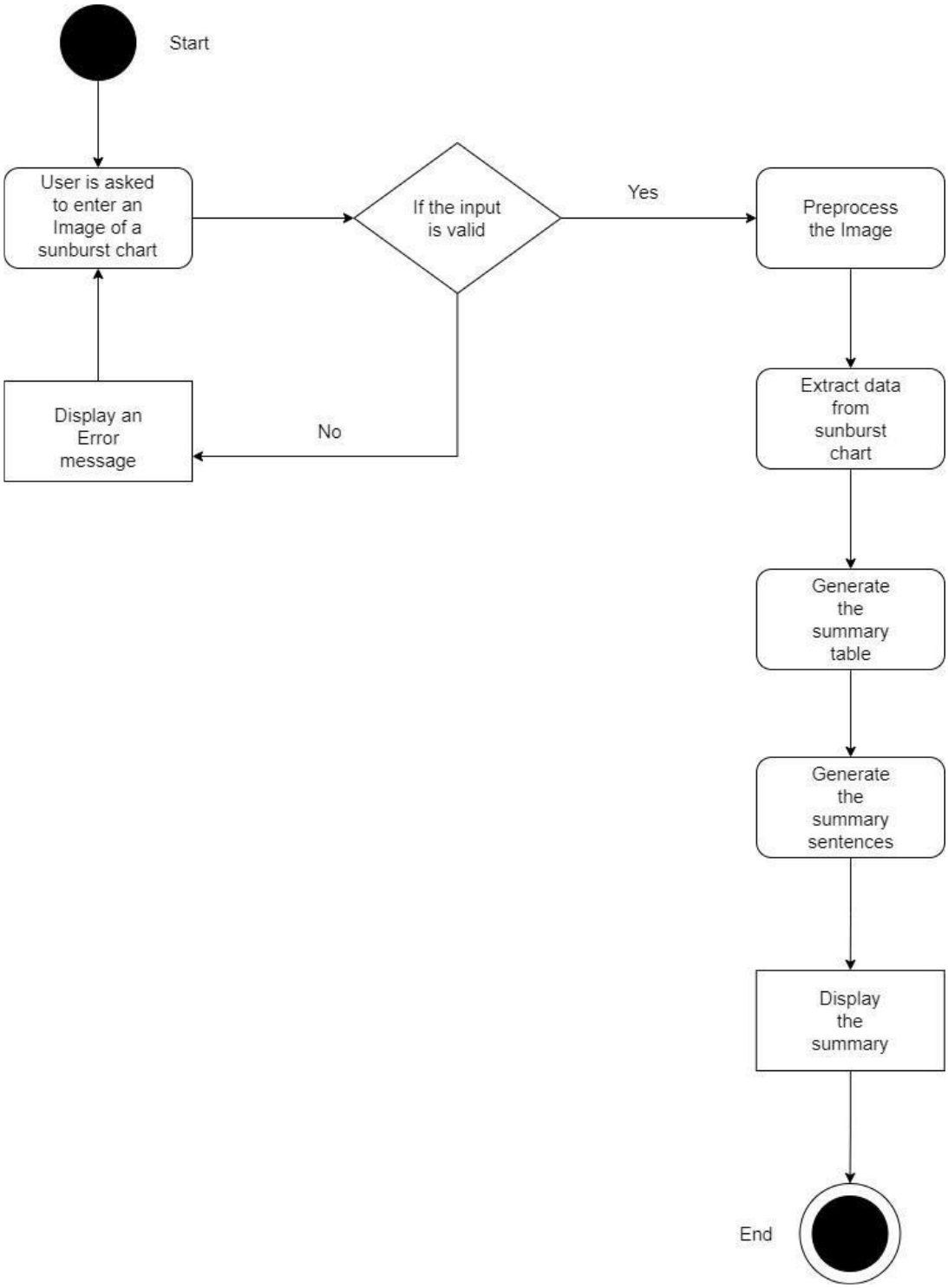



*Figure 5. 2 Activity diagram*

### 5.5.2 Data Flow Diagram

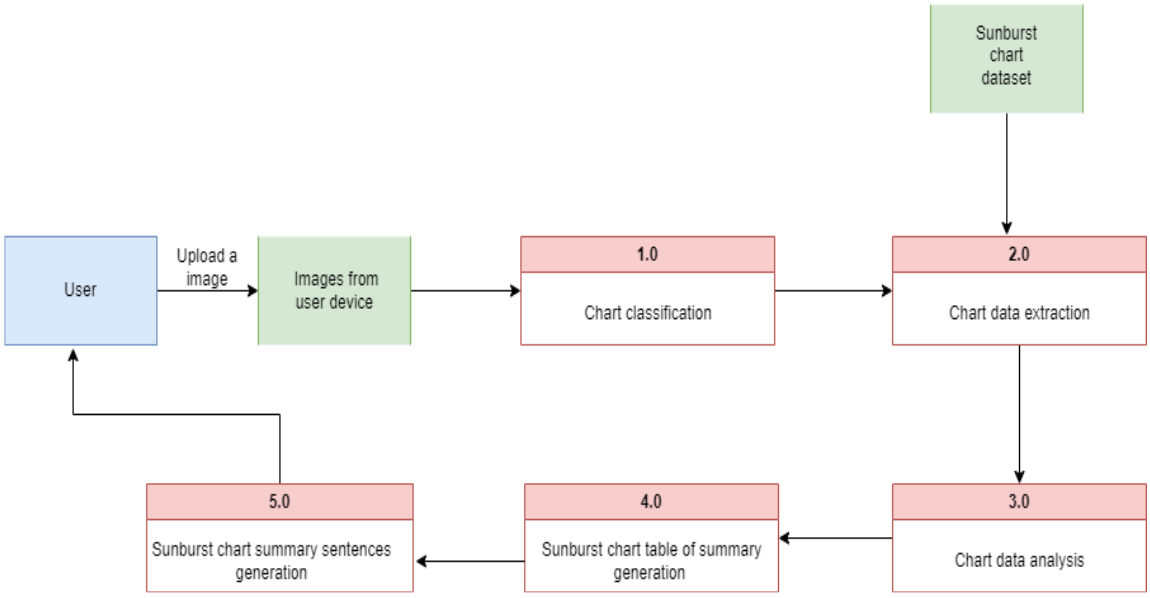



*Figure 6.3: Data flow diagram L1*

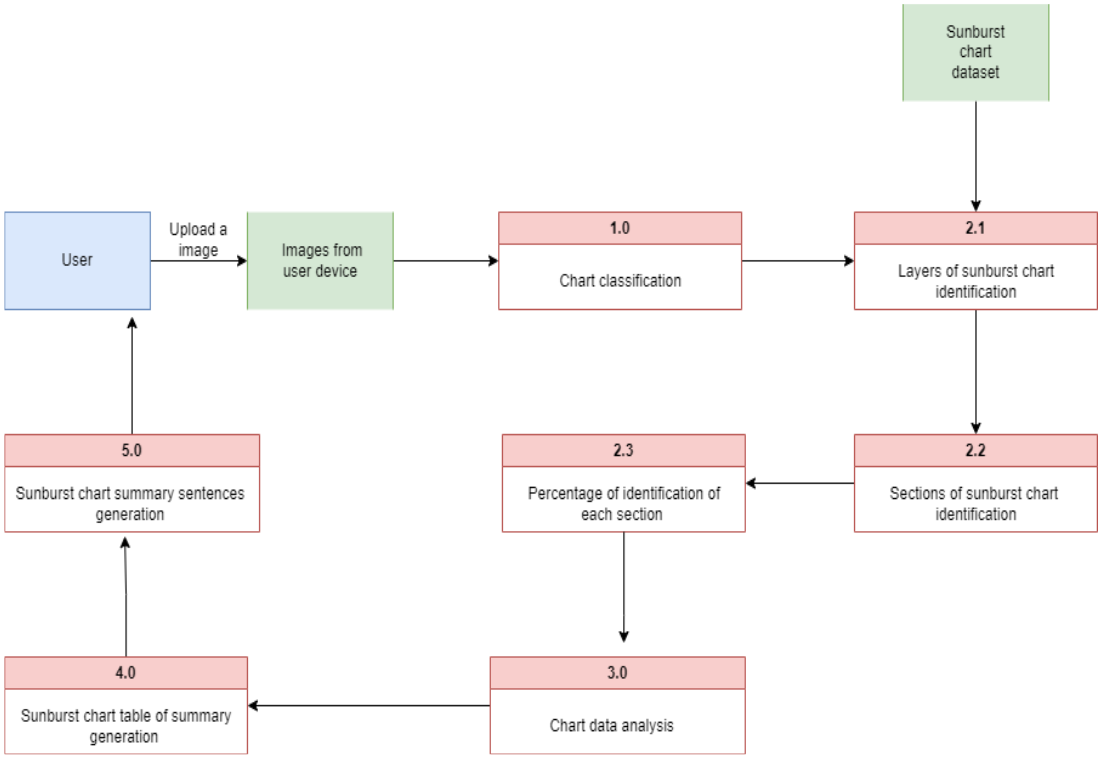



*Figure 6.4: Data flow diagram L2*

## 5.6 UI Wireframe Design

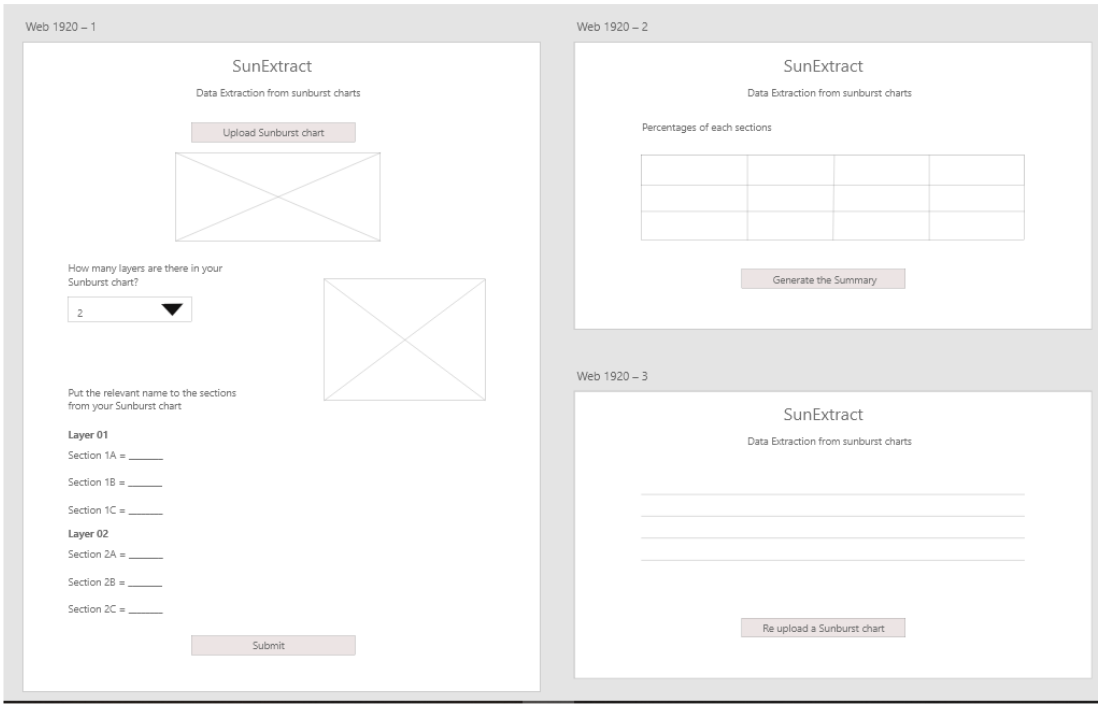



*Figure 6. 5: UI Wireframe*

## 5.7 Chapter Summary

During this project's design level, influenced to choose and use the System Analysis and Design designing method. I believe, A detailed overview was made to give the knowledge of the whole system. The design and architecture diagrams were presented to illustrate the tasks of the flow and how data moves within it. The next chapter will focus on implementing the system and further exploring the design aspect already discussed.

# CHAPTER 6

# IMPLEMENTATION OF SUNBURST CHART SUMMARIZATION SYSTEM

## 6.1 Introduction

This chapter provides an in-depth look at the central implementation of the project, covering the technologies, programming languages, and tools used in its development. It also explains the rationale behind each choice. The processes outlined in this chapter culminate in the results, discussing all implementations involved in these processes.

## 6.2 Implementation of sub modules

This chapter we use to describe the implementations part of the sub module of sunburst chart summary generation. These details are written down below. The results of each module also showing below.

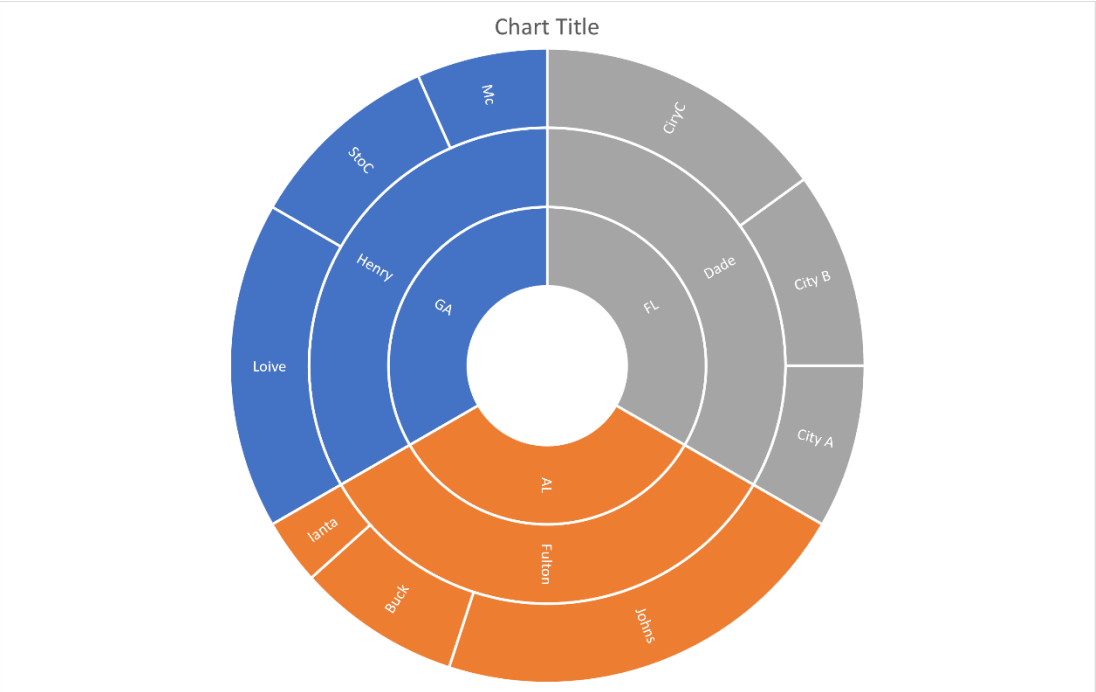
### 6.2.1 Image pre-processing

As we discussed in the designing part, image preprocessing will result for all the rest of the processes. So here we first upload the image to the system and resize image to get a image with a fixed size, this will reduce the size of image but not the quality of the image.

```
imgname = 'sun_3.png'
src1 = cv.imread(imgname)
src = cv.resize(src1, (0,0), fx=0.33, fy=0.33)
cv.imwrite("sun_3_processed.png", src)
```

Within the next steps, used some preprocessing steps, here before detect the circles of the image, I have added a preprocessing steps as follows.

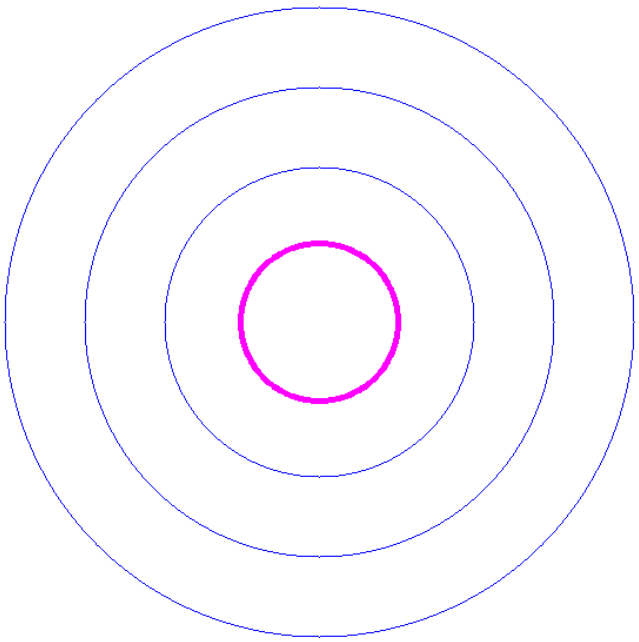
```
# preprocsessing
gray2 = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
gray2 =  cv.GaussianBlur(gray2,(3,3),0)
# wide = cv.Canny(gray, 10, 200)
mid = cv.Canny(gray2, 30, 150)
# tight = cv.Canny(gray, 240, 250)
gray1 = cv.Canny(gray2, 30, 150)
```




### 6.2.2 Circles Detection

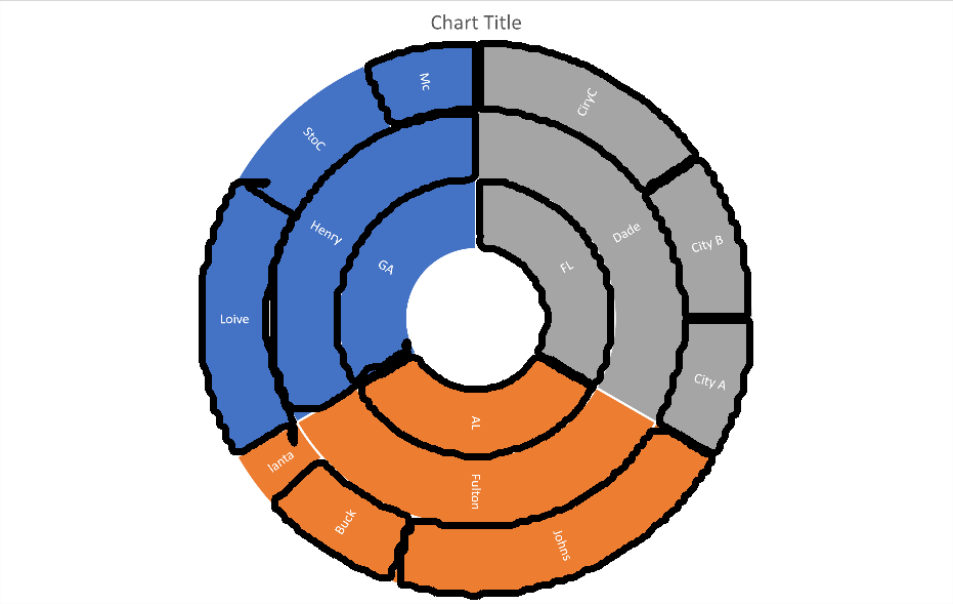
Preprocessed and enhanced image will use to detect the circles. So, detect all the circles using HoughCircles library and find the small circle first. Then find the center of image and find the next circles which having the specific gap between each circle. All the parameters and their values calculated after validated through all images in the dataset.

```
circles = cv.HoughCircles(gray, cv.HOUGH_GRADIENT, 1, rows / 8,
                          param1=100, param2=30,
                          minRadius=b, maxRadius=a)
```



### 6.2.3 Segments Detection

After we detected the circles, we want to detect the segments of each circle because we want to extract these features for chart summary generation. Therefore, we trained a dataset and developed a VIT model to extract these segments and texts. We want to know where the segments and texts is located exactly in the image, what are pixels belong to which segment and the circle, etc. In this case we want to be trained and developed the model using RoboFlow. Following results, we achieved through the training about 100 images of sunburst charts. So, the output of the model looks as follows.




### 6.2.4 Calculate Percentages and Colors

After detected the segment with their circle, we can calculate the area and percentage of each segment in that circle. Then I got the color of each segment and that will use to identify the same categories. So finally, I can get the result in JSON format as follows. Here the text also included, I will explain how to use OCR to identify the texts.


```
[{'color': (165, 165, 165), 'hierarchy': 1, 'area': 13584,
'percentage': 32.98108448928121, 'text': 'text_3'}, {'color':
(49, 125, 237), 'hierarchy': 1, 'area': 13794, 'percentage':
32.86002522068096, 'text': 'text_18'}, {'color': (196, 114, 68),
'hierarchy': 2, 'area': 23630, 'percentage': 32.00873425592527,
'text': 'text_5'}, {'color': (49, 125, 237), 'hierarchy': 3,
'area': 21026, 'percentage': 20.979078805920782, 'text':
'text_0'}, {'color': (165, 165, 165), 'hierarchy': 3, 'area':
9656, 'percentage': 9.644008930786406, 'text': 'text_1'},
{'color': (165, 165, 165), 'hierarchy': 3, 'area': 8067,
'percentage': 7.983957661457042, 'text': 'text_4'}, {'color':
(49, 125, 237), 'hierarchy': 3, 'area': 8062, 'percentage':
7.916770032250062, 'text': 'text_6'}, {'color': (196, 114, 68),
'hierarchy': 3, 'area': 16336, 'percentage': 16.16637724303316,
'text': 'text_9'}, {'color': (165, 165, 165), 'hierarchy': 3,
'area': 14045, 'percentage': 14.23447448937402, 'text':
```

'text_15'}, {'color': (196, 114, 68), 'hierarchy': 3, 'area': 6431, 'percentage': 6.359050690482097, 'text': 'text_17'}]

### 6.2.5 Text Detection and Extraction

As explained above, we can detect the texts as object as result of the VIT model we developed. Using those text objects, we can get the text using RoboFlow OCR API. Here we input the text object as an image, The API will detect the text inside it.

```
# Run OCR on the canvas with the resized text segment
ocr_result = CLIENT.ocr_image(inference_input=resized_text_segment_path)
print(f"Text Segment {i} OCR Result:")
print(ocr_result)
```

### 6.2.6 Summary Generation

The final module of the project is, generating the summary using the above features we extracted. For the Language Model (LLM) aspect of this, we leveraged the capabilities of GPT-2, a pre-trained model by OpenAI, to interpret and summarize the data extracted from sunburst charts. Here's a breakdown of the steps and the code functionality related to the LLM:

- **Data Preparation:** We structured the data extracted from the charts into a JSON format, encapsulating segments, hierarchies, and texts along with their confidence scores. This structured data serves as the input for generating textual descriptions.
- **Textual Description Generation:** Using the GPT-2 tokenizer and model, we transformed the structured information into a prompt that describes the chart's contents, including segments, hierarchies, and key texts, along with their confidence levels. An example description might detail the number of segments detected, their confidence scores, the hierarchy levels present, and the key financial metrics identified within the texts.
- **Summary Generation:** We employed the GPT-2 model to generate a summary based on the textual description created in the previous step. This involves encoding the description into tokens, feeding these tokens into the model, and then decoding the output to produce a coherent summary. The generated summary encapsulates the essence of the chart, highlighting key insights and relationships between different data points, such as revenue, expenses, and other significant metrics.

This approach allowed us to bridge the gap between visual data representation and textual interpretation, enabling the extraction of meaningful insights from complex sunburst charts through the use of advanced NLP techniques. This is the final outcome of the system.

The chart's main categories are labeled with "text_3" and "text_18," each occupying 32.98% and 32.86% of the total area, respectively. Within "text_3," a subcategory labeled "text_5" occupies 32.01% of the area. In "text_18," a subcategory labeled "text_0" occupies 20.98% of the area.

Further breakdowns include "text_1," "text_4," and "text_15" in the gray category, each with different percentages of the total area. In the blue category, "text_6" and "text_17" are included, each with distinct area percentages. This hierarchical representation helps visualize how each text label contributes to the overall area distribution.

---

## 6.2 Chapter Summary

There are various functions happening inside the sunburst chart summarization System, most of them are described in a detailed manner using the flow charts and algorithms as above. Next chapter contain the discussion about the current progress of the project and further work needs to be addressed.

# CHAPTER 7

# EVALUATION OF CHART SUMMARIZATION SYSTEM FOR SUNBURST CHARTS

## 7.1 Introduction

This chapter is used to show the evaluation parts of each sub module. Here we will analyze whether the implemented solution addresses the defined problems and with the degree of accuracy and other evaluation models. The evaluation process of sunburst chart summary generation goes through as four modules.

- Circles Detection.
- Segments Detection
- Calculate Percentages and Colors
- Texts Extraction

These modules evaluated using quantitative and qualitative manner.

### 7.1.1 Circles Detection

When considering the project, after initial input image, circles detection is the first module. Finally, after this module, the system can detect the other features as desired. The evaluation of this module was done by running the program and checking whether the circles are clearer to extract. The accuracy percentage was calculated using error-rate.

Accuracy of the module can be obtained by taking the average of the accuracy of each individual result. Following results obtained by the circles detection algorithm. An average accuracy of 64% was obtained by this method.

### 7.1.2 Segments Detection

The second main task is segments detection. After the circle's detection task, we can find the input image is a sunburst chart image or not. If it's a sunburst chart or else we can detect circles correctly, the next step is segments detection.

Here we are using a ViT model to detect the segments and get the position details of them. So, I calculated manually, how many segments the model output can given as the result. It's 60% of all segments tested.

### 7.1.3 Calculate Percentages and Colors

After identified the segments, we had found the percentage of each segment in that circle, for that we should get the color of the segment, so this process is depending on the previous two sub modules.

Accuracy of the module can be obtained by taking the average of the accuracy of each individual result. Following results obtained by the calculating percentage and colors. An average accuracy of 80% was obtained by this method.

### 7.1.4 Texts Extraction

The accuracy of text extraction module is dependent on the API endpoint we call. Also in our case, the text is curved or rotated. So, with these issues, the accuracy of text extraction is very low. As we discussed in the literature review, we faced those issues here. So, I calculated the text extraction accuracy as same as above in other modules and it's 55%.

### 7.1.5 Summary Generation

This is the last output of the project, so the accuracy of this module is equal to the accuracy of the whole process. Also, summary generation is based on how fine-tuned the GPT model and the quality of our prompting. So, we can handle it by handling either one or two course of actions. However, the main factor is that we detect and extract the all-chart components correctly. The evaluation of summary generation is qualitative. We have compared both the developed system generated summary and the summary written by an expert in a field that relevant tot the chart. Such as salesperson of a company, marketing employee, HR officers, statisticians likewise. So, I added that as a future work.

## 7.4 Conclusion

When evaluating the implemented solution, it is observed that the designed sunburst chart summary extraction provides reasonably precise outputs with respect to expertise knowledge. The evaluation stage is found to be challenging to analyze the outputs logically by comparing with an actual scenario. The next chapter will conclude the thesis along with mentioned limitations and recommendations for further development of the design.

# CHAPTER 8

# CONCLUSION, LIMITATIONS AND FURTHER WORKS

## 8.1 Introduction

This is the final chapter of the report which summarizes the findings and concludes the research. It will also discuss the limitations of each research component along with the possible improvements in the future.

## 8.2 Conclusion

The implemented system for sunburst chart summarization consists of five main sub modules and which are as follows:

- Preprocessing and Circles Detection.
- Segments Detection
- Calculate Percentages and Colors
- Texts Extraction
- Summary Generation

In preprocessing module and in the circle's detection modules, the proposed algorithms are used to detect the number of circles and their center and the radius of each circle. It has been tested on 1000 images. The implemented algorithm has been able to obtain circles extraction 65% of cases.

As the conclusion, the accuracy of segment detection is dependent on the VIT model that we developed. It gives 98% of accuracy when detecting the segments, title, and texts.

When the segments detection accuracy is 98%, it will directly help to make the accuracy of calculating percentage and colors is 98%. Because those processes giving exact results. But the next part of, extracting texts will reduce the overall accuracy because some of the limitations of pre-trained OCR model.

Finally, the summary generation is in high level of results, but there cannot find the accuracy directly. But the level of giving high accuracy results of GPT-3.5, It has been with very high level of accuracy.

## 8.3 Limitations

### 8.3.1 Preprocessing and Circles Detection

There are limitations that can be there in image enhancing. Due to the salt and paper noise in the images and most images were not in a good condition. So, more attention would have been given for the image enhancement and proper filtering method would have suggested for more accurate results. So, we would enhance the circle detection's results.

Also there have so many varieties in sunburst charts. Here we limited to the clearer and excel generated sunburst charts only. And we can see that, some sunburst charts are very complex, so in that case, hard to detect the circles. So when we detect the

number of circles and their positions correctly, it will help to detect the segments, colors, and percentages precisely.

### 8.3.2 Texts Extraction

The next limitation and challenge are detecting texts. Most of the time the title in on the top of each chart but sometimes it's change to the bottom, left or right sides. As well as the texts' position inside the segments also change but most of the time it's centered in the segment.

### 8.3.5 Summary Generation

Summary Generation also has limitations because it's depended on GPT-2 libraries. It will need to tune up until we get a quality result.

As I believe, those are the main limitations, and they will need to, and they can address in the future.

## 8.2 Further Works

The sunburst chart summarization system should have some enhancements in the future. The solution is generated using sunburst charts which generated using excel sheets as considering the most general case. And the system is failed to be considered percentage to detect the circles, this will affect the overall results, so this should enhance. Finally, the quality of summary generation should tune-up until giving a quality result, and it's depend on GPT-3 model. The time will consume for an image should reduce because the current performances not enough.

# REFERENCES


[1] Luo, J., Li, Z., Wang, J., Lin, C.-Y., 2021. ChartOCR: Data Extraction from Charts Images via a Deep Hybrid Framework, in: 2021 IEEE Winter Conference on Applications of Computer Vision (WACV). Presented at the 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, Waikoloa, HI, USA, pp. 1916–1924. https://doi.org/10.1109/WACV48630.2021.00196

[2] Jung, D., Kim, W., Song, H., Hwang, J., Lee, B., Kim, B., Seo, J., 2017. ChartSense: Interactive Data Extraction from Chart Images, in: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems.

[3] Bajić, F., Job, J., 2022. Data Extraction of Circular-Shaped and Grid-like Chart Images. J. Imaging 8, 136. https://doi.org/10.3390/jimaging8050136

[4] Mishra, P., Kumar, S., Chaube, M.K., 2021. ChartFuse: a novel fusion method for chart classification using heterogeneous microstructures. Multimed Tools Appl 80, 10417–10439. https://doi.org/10.1007/s11042-020-10186-z

[5] Al-Zaidy, R.A., Giles, C.L., 2015. Automatic Extraction of Data from Bar Charts, in: Proceedings of the 8th International Conference on Knowledge Capture. Presented at the K-CAP 2015: Knowledge Capture Conference, ACM, Palisades NY USA, pp. 1–4. https://doi.org/10.1145/2815833.2816956

[6] Dadhich, K., Daggubati, S., Sreevalsan-Nair, J., 2021. BarChartAnalyzer: Digitizing Images of Bar Charts:, in: Proceedings of the International Conference on Image Processing and Vision Engineering. Presented at the International Conference on Image Processing and Vision Engineering, SCITEPRESS - Science and Technology Publications, Online Streaming, --- Select a Country ---, pp. 17–28. https://doi.org/10.5220/0010408300170028

[7] Decatur, D., Krishnan, S., 2021. VizExtract: Automatic Relation Extraction from Data Visualizations.

[8] Dyomin, V.V., Kamenev, D.V., 2016. Evaluation of Algorithms for Automatic Data Extraction from Digital Holographic Images of Particles. Russ Phys J 58, 1467–1474. https://doi.org/10.1007/s11182-016-0669-z

[9] Gao, J., Zhou, Y., Barner, K.E., 2012. View: Visual Information Extraction Widget for improving chart images accessibility, in: 2012 19th IEEE International Conference on Image Processing. Presented at the 2012 19th IEEE International Conference on Image Processing (ICIP 2012), IEEE, Orlando, FL, USA, pp. 2865–2868. https://doi.org/10.1109/ICIP.2012.6467497

[10] Al-Zaidy, R.A., Choudhury, S.R., Giles, C.L., n.d. Automatic Summary Generation for Scientific Data Charts 6.

[11] Presented at the CHI '17: CHI Conference on Human Factors in Computing Systems, ACM, Denver Colorado USA, pp. 6706–6717. https://doi.org/10.1145/3025453.3025957

[12] Kafle, K., Price, B., Cohen, S., Kanan, C., 2018. DVQA: Understanding Data Visualizations via Question Answering, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Presented at the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Salt Lake City, UT, pp. 5648–5656. https://doi.org/10.1109/CVPR.2018.00592

[13] Kantharaj, S., Leong, R.T., Lin, X., Masry, A., Thakkar, M., Hoque, E., Joty, S., 2022. Chart-to-Text: A Large-Scale Benchmark for Chart Summarization, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Presented at the Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, pp. 4005–4023. https://doi.org/10.18653/v1/2022.acl-long.277

[14] Kim, D.H., Hoque, E., Agrawala, M., 2020. Answering Questions about Charts and Generating Visual Explanations, in: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. Presented at the CHI '20: CHI Conference on Human Factors in Computing Systems, ACM, Honolulu HI USA, pp. 1–13. https://doi.org/10.1145/3313831.3376467

[15] Liu, X., Klabjan, D., NBless, P., 2019. Data Extraction from Charts via Single Deep Neural Network.

[16] Dyomin, V.V., Kamenev, D.V., 2016. Evaluation of Algorithms for Automatic Data Extraction from Digital Holographic Images of Particles. Russ Phys J 58, 1467–1474. https://doi.org/10.1007/s11182-016-0669-z

[17] Masry, A., Long, D., Tan, J.Q., Joty, S., Hoque, E., 2022. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning, in: Findings of the Association for Computational Linguistics: ACL 2022. Presented at the Findings of the Association for Computational Linguistics: ACL 2022, Association for Computational Linguistics, Dublin, Ireland, pp. 2263–2279. https://doi.org/10.18653/v1/2022.findings-acl.177

[18] Masry, A., Prince, E.H., n.d. Integrating Image Data Extraction and Table Parsing Methods for Chart Question answering 5.

[19] Methani, N., Ganguly, P., Khapra, M.M., Kumar, P., 2020. PlotQA: Reasoning over Scientific Plots, in: 2020 IEEE Winter Conference on Applications of Computer Vision (WACV). Presented at the 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, Snowmass Village, CO, USA, pp. 1516–1525. https://doi.org/10.1109/WACV45572.2020.9093523

[20] Mishchenko, A., Vassilieva, N., 2011. Model-Based Chart Image Classification, in: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Wang, S., Kyungnam, K., Benes, B., Moreland, K., Borst, C., DiVerdi, S., Yi-Jen, C., Ming, J. (Eds.), Advances in Visual Computing, Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 476–485. https://doi.org/10.1007/978-3-642-24031-7_48

[21] Araújo, T., Chagas, P., Alves, J., Santos, C., Sousa Santos, B., Serique Meiguins, B., 2020. A Real-World Approach on the Problem of Chart Recognition

Using Classification, Detection and Perspective Correction. Sensors 20, 4370. https://doi.org/10.3390/s20164370

[22] Mishra, P., Kumar, S., Chaube, M.K., Shrawankar, U., 2022. ChartVi: Charts summarizer for visually impaired. Journal of Computer Languages 69, 101107. https://doi.org/10.1016/j.cola.2022.101107

[23] Oyama, S., Baba, Y., Ohmukai, I., Dokoshi, H., Kashima, H., 2016. Crowdsourcing chart digitizer: task design and quality control for making legacy open data machine-readable. Int J Data Sci Anal 2, 45–60. https://doi.org/10.1007/s41060-016-0025-y

[24] Paliwal, S.S., D, V., Rahul, R., Sharma, M., Vig, L., 2019. TableNet: Deep Learning Model for End-to-end Table Detection and Tabular Data Extraction from Scanned Document Images, in: 2019 International Conference on Document Analysis and Recognition (ICDAR). Presented at the 2019 International Conference on Document Analysis and Recognition (ICDAR), IEEE, Sydney, Australia, pp. 128–133. https://doi.org/10.1109/ICDAR.2019.00029

[25] Saket, B., Endert, A., Demiralp, C., 2019. Task-Based Effectiveness of Basic Visualizations. IEEE Trans. Visual. Comput. Graphics 25, 2505–2512. https://doi.org/10.1109/TVCG.2018.2829750

# APPENDIX A – SOURCE CODE

```
!pip install inference inference-sdk supervision
# import the inference-sdk
from inference_sdk import InferenceHTTPClient
import cv2 as cv

# initialize the client
# CLIENT = InferenceHTTPClient(
#     api_url="https://detect.roboflow.com",
#     api_key="icWwcIwXkZDEdEmz2KdS"
# )

# infer on a local image
imgname = 'sun_3.png'
src1 = cv.imread(imgname)
src = cv.resize(src1, (0,0), fx=0.33, fy=0.33)
cv.imwrite("sun_3_processed.png", src)
# result = CLIENT.infer("sun_3_processed.png",
model_id="sunburst-charts-im5uv/1")
# Assuming 'result' is your JSON object with predictions
import cv2 as cv
import numpy as np
from scipy import stats

# Assuming 'result' is your JSON object with predictions and
'image' is your loaded image
image_path = "sun_3_processed.png"
image = cv.imread(image_path)
for i, prediction in enumerate(result['predictions']):
    if prediction['class'] == 'segment':
        points = prediction['points']
        pts = np.array([[int(point['x']), int(point['y'])] for
point in points], np.int32)
        pts = pts.reshape((-1, 1, 2))

        # Create a mask for the segment
        mask = np.zeros_like(image)
        cv.fillPoly(mask, [pts], (255, 255, 255))

        # Extract the segment and calculate its main color
        masked_image = cv.bitwise_and(image, mask)
        masked_image_flattened = masked_image.reshape(-1, 3)

        # Remove all black ([0,0,0]) background pixels
        non_black_pixels_mask = np.all(masked_image_flattened !=
[0, 0, 0], axis=1)
        non_black_pixels =
masked_image_flattened[non_black_pixels_mask]

        # Get the most frequent color using a dictionary to count
occurrences
        color_counts = {}
        for color in non_black_pixels:
            color_tuple = tuple(color)
            if color_tuple in color_counts:
                color_counts[color_tuple] += 1
```

```
            else:
                color_counts[color_tuple] = 1

        # Find the most frequent color
        mode_color = max(color_counts, key=color_counts.get)

        # Convert the most frequent color to integers
        mode_color_int = (int(mode_color[0]), int(mode_color[1]),
int(mode_color[2]))

        # Print the main color
        print(f"Segment {i}: Main Color (B, G, R) =
{mode_color_int}")

        # Draw the segment on the original image
        cv.polylines(image, [pts], isClosed=True, color=(0,0,0),
thickness=5)

# Save the image with the drawn segments
cv.imwrite("output_segments.png", image)
import sys
import cv2 as cv
import numpy as np
import os

# Load the original image
imgname = 'sun_3_processed.png'

print(imgname)
src = cv.imread(imgname)
original1 = src.copy()
original2 = src.copy()
original = src.copy()
print(original.shape[2])
height = original.shape[0]
width = original.shape[1]


# preprocsessing
gray2 = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
gray2 =  cv.GaussianBlur(gray2,(3,3),0)
# wide = cv.Canny(gray, 10, 200)
mid = cv.Canny(gray2, 30, 150)
# tight = cv.Canny(gray, 240, 250)
gray1 = cv.Canny(gray2, 30, 150)

# show the output Canny edge maps
# cv.imshow("Mid Edge Map", mid)
# cv.waitKey(0)


contours, hierarchy = cv.findContours(mid, cv.RETR_TREE,
cv.CHAIN_APPROX_SIMPLE)
contour_list = []
contour_list_bad = []
cnt = sorted(contours, key=cv.contourArea, reverse=True)

for contour in cnt:
```

```
    approx =
cv.approxPolyDP(contour,0.01*cv.arcLength(contour,True),True)
    area = cv.contourArea(contour)
    # if ((len(approx) > 8) & (len(approx) < 23) & (area > 200)
):
    if (area > 200):
        contour_list.append(contour)
        cv.drawContours(src, [contour], -1, (255,0,0), 2)
        M = cv.moments(contour)
        if M['m00'] != 0:
            cx = int(M['m10']/M['m00'])
            cy = int(M['m01']/M['m00'])
    else:
        contour_list_bad.append(contour)
        cv.drawContours(src, [contour], -1, (255,255,255), 2)

# cv.drawContours(src, contour_list_bad, -1, (255,255,255), 2)
# cv.drawContours(src, contour_list,  -1, (255,0,0), 2)


for contour_bad in contour_list_bad:
    M = cv.moments(contour_bad)
    if M['m00'] != 0:
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
    checked = False
    # print(str(cx)+" "+str(cy))
    for pix in range(cx,cx+60,1) :
        try:
            if(original[pix,cy] != (255, 255, 255)).all():

                cv.drawContours(src, [contour_bad], -1,
(int(original[pix,cy][0]),int(original[pix,cy][1]),int(original[p
ix,cy][2])), 2)
                checked = True
                break
        except:
            pass
    if(checked != True):
        for pix in range(cy,cy+50,1) :
            try:
                if(original[cx,pix] != (255, 255, 255)).all():

                    cv.drawContours(src, [contour_bad], -1,
(int(original[cx,pix][0]),int(original[cx,pix][1]),int(original[c
x,pix][2])), 2)
                    checked = True
                    break
            except:
                pass


# cv.imshow('Objects Detected',src)
# cv.waitKey(0)

gray = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
gray =  cv.GaussianBlur(gray,(3,3),0)
# cv.imshow('Obje',gray)
```

```
# cv.waitKey(0)

rows = gray.shape[0]
b=40
count=0
centerXX = 0
centerYY = 0
center = ()

for a in range(48,int(gray.shape[1]/2),8) :
    circles = cv.HoughCircles(gray, cv.HOUGH_GRADIENT, 1, rows /
8,
                              param1=100, param2=30,
                              minRadius=b, maxRadius=a)

    if circles is not None:
        print(len(circles[0,:]))

    try:
        if len(circles[0,:])==1:
            count = count+1
            b=a+8
            circles = np.uint16(np.around(circles))
            for i in circles[0, :]:
                center = (i[0], i[1])
                centerXX=i[0]
                centerYY=i[1]
                # circle center1
                cv.circle(gray1, center, 3, (0, 100, 100), 3)
                # circle outline
                radius = i[2]
                cv.circle(gray1, center, radius, (255, 0, 255),
3)
            break
    except:
        print("exception 1")

print("XX", str(centerXX))
print("YY", str(centerYY))
# cv.imshow('Objec',gray1)
# cv.waitKey(0)

# Get the white Backgroud Image
white = src.copy()
for w in range (width):
    for h in range (height):
        white[h,w] = (255,255,255)

# Draw the Mid Circle on the white Backgroud Image
try:
    cv.circle(white, center, radius, (255, 0, 255), 3)
except:
    print("No circle Detected")
    exit()

# Find the Line Color
different_colors = []
# get the 1st segment and count color types and pixels--------->4
```

```
for x in range(height):
    for y in range(width):
        if (white[x,y] == [255, 0, 255]).all():
            checked = False
            for a in different_colors:
                if (a[0] == original[x,y]).all():
                    a[1] = a[1] + 1
                    checked = True
                    break
            if not checked:
                different_colors.append([original[x,y], 1])

different_colors_sorted = sorted(different_colors, key=lambda x:
x[1], reverse=True)
line_color = different_colors_sorted[0][0]

# cv.imshow('White',white)
# cv.waitKey(0)

for contour in cnt:
    cv.drawContours(original, [contour], -1,
(int(line_color[0]),int(line_color[1]),int(line_color[2])), 2)


radius = []
for x in range(centerXX,width-1,1):
    if(original[centerYY+4,x] ==
(line_color[0],line_color[1],line_color[2])).all():
        # cv.circle(white, center, (x - centerXX), (255, 0, 0),
1)
        if(original[centerYY+4,x-1] !=
(line_color[0],line_color[1],line_color[2])).all():
            cv.circle(white, center, (x - centerXX), (255, 0, 0),
1)
            radius.append(x - centerXX)

# cv.imshow('White with Line',white)
# cv.waitKey(0)
segmented_parts = []

# Initialize the OCR client
# CLIENT = InferenceHTTPClient(
#     api_url="https://infer.roboflow.com",
#     api_key="icWwcIwXkZDEdEmz2KdS"
# )

def segments_in_segment_circle(imagez, heirarchy_area, idx):
      for i, prediction in enumerate(result['predictions']):
          if prediction['class'] == 'title':
            title = "Title"
            points1 = pred['points']
            pts1 = np.array([[int(point['x']), int(point['y'])]
for point in points1], np.int32)
            pts1 = pts1.reshape((-1, 1, 2))
            image_text_black = np.full_like(imagez,
fill_value=255, dtype=np.uint8)
            cv.fillPoly(image_text_black, [pts1], color=(0, 0,
0))
```

```
            rect = cv.boundingRect(pts1)
            x, y, w, h = rect

            # Extract the text segment from the original image
            text_segment = original2[y:y+h, x:x+w]

            # Resize the text segment by 10 times
            resized_text_segment = cv.resize(text_segment, None,
fx=10, fy=10, interpolation=cv.INTER_LANCZOS4)

            # Create a blank canvas with the same size as the
original image
            canvas = np.zeros_like(original2)

            # Calculate the new position and size for the resized
text segment
            new_w, new_h = resized_text_segment.shape[0],
resized_text_segment.shape[1]
            new_x = min(x, original2.shape[0] - new_w)
            new_y = min(y, original2.shape[1] - new_h)

            # Place the resized text segment onto the canvas
within the bounds of the original image
            canvas[ new_x:new_x+new_w, new_y:new_y+new_h] =
resized_text_segment

            # Save the canvas with the resized text segment
            resized_text_segment_path =
f"resized_text_segment_{i}.png"
            cv.imwrite(resized_text_segment_path, canvas)

            # Run OCR on the canvas with the resized text segment
            # title =
CLIENT.ocr_image(inference_input=resized_text_segment_path)
            print(f"Text Title {i} OCR Result:")
            print(title)

            segmented_parts.append({
                'title': title
            })

          if prediction['class'] == 'segment':
            points = prediction['points']
            pts = []
            for point in points:
              pts.append([int(point['x']), int(point['y'])])

            pts = np.array(pts, np.int32)
            pts = pts.reshape((-1, 1, 2))

            ######################## Find Color
################################
            # Create a mask for the segment
            mask = np.zeros_like(original2)
            cv.fillPoly(mask, [pts], (255, 255, 255))

            # Extract the segment and calculate its main color
            masked_image = cv.bitwise_and(image, mask)
```

```
            masked_image_flattened = masked_image.reshape(-1, 3)

            # Remove all black ([0,0,0]) background pixels
            non_black_pixels_mask = np.all(masked_image_flattened
!= [0, 0, 0], axis=1)
            non_black_pixels =
masked_image_flattened[non_black_pixels_mask]

            # Get the most frequent color using a dictionary to
count occurrences
            color_counts = {}
            for color in non_black_pixels:
                color_tuple = tuple(color)
                if color_tuple in color_counts:
                    color_counts[color_tuple] += 1
                else:
                    color_counts[color_tuple] = 1

            # Find the most frequent color
            mode_color = max(color_counts, key=color_counts.get)

            # Convert the most frequent color to integers
            mode_color_int = (int(mode_color[0]),
int(mode_color[1]), int(mode_color[2]))
            ######################## Find Color
##################################


            image_black = np.full_like(imagez, fill_value=255,
dtype=np.uint8)
            cv.fillPoly(image_black, [pts], color=(0, 0, 0))

            ######################## Find Text
##################################
            ocr_result = 'Other'
            for j, pred in enumerate(result['predictions']):
              if pred['class'] == 'text':
                  points1 = pred['points']
                  pts1 = np.array([[int(point['x']),
int(point['y'])] for point in points1], np.int32)
                  pts1 = pts1.reshape((-1, 1, 2))
                  image_text_black = np.full_like(imagez,
fill_value=255, dtype=np.uint8)
                  cv.fillPoly(image_text_black, [pts1], color=(0,
0, 0))

                  pas = 0
                  tot = 0
                  flag = True
                  for w in range(image_text_black.shape[0]):
                      for h in range(image_text_black.shape[1]):
                          if (image_text_black[w, h] == (0, 0,
0)).all():

                              if (image_black[w, h] != (0, 0,
0)).all():
                                flag = False
                                break;
```

```
                    if(flag):
                        rect = cv.boundingRect(pts1)
                        x, y, w, h = rect

                        # Extract the text segment from the original
image
                        text_segment = original2[y:y+h, x:x+w]

                        # Resize the text segment by 10 times
                        resized_text_segment =
cv.resize(text_segment, None, fx=10, fy=10,
interpolation=cv.INTER_LANCZOS4)

                        # Create a blank canvas with the same size as
the original image
                        canvas = np.zeros_like(original2)

                        # Calculate the new position and size for the
resized text segment
                        new_w, new_h = resized_text_segment.shape[0],
resized_text_segment.shape[1]
                        new_x = min(x, original2.shape[0] - new_w)
                        new_y = min(y, original2.shape[1] - new_h)

                        # Place the resized text segment onto the
canvas within the bounds of the original image
                        canvas[ new_x:new_x+new_w, new_y:new_y+new_h]
= resized_text_segment

                        # Save the canvas with the resized text
segment
                        resized_text_segment_path =
f"resized_text_segment_{i}.png"
                        cv.imwrite(resized_text_segment_path, canvas)

                        # Run OCR on the canvas with the resized text
segment
                        # ocr_result =
CLIENT.ocr_image(inference_input=resized_text_segment_path)
                        print(f"Text Segment {i} OCR Result:")
                        print(ocr_result)
                ######################## Find Text
###################################

                pas = 0
                tot = 0
                circle_tot = 0
                for w in range(image_black.shape[0]):
                    for h in range(image_black.shape[1]):
                        if (original2[w, h] == imagez[w, h]).all()
and (original2[w, h] != (255, 255, 255)).all():
                            circle_tot +=1
                        if (image_black[w, h] == (0, 0, 0)).all():
                            tot += 1
                            if (original2[w, h] == imagez[w,
h]).all() and (original2[w, h] != (255, 255, 255)).all():
                                pas += 1
```

```
            print("heirarchy_area ",heirarchy_area)
            if tot > 0 and pas / tot > 0.8:
                segmented_parts.append({
                      'color': mode_color_int,
                      'hierarchy': idx,
                      'area': tot,
                      'percentage' : (pas/heirarchy_area)*100,
                      'text': ocr_result
                })
                cv.polylines(imagez, [pts], isClosed=True,
color=(0, 255, 0), thickness=5)

            # Draw the polygon

            # Save the image with drawn segments

      cv.imwrite(f'segment_with_drawn_parts_{i}_{idx}.png',
imagez)
      print(segmented_parts)

def segment_circle(image, center, radii):
    # Convert the image to grayscale
    # gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Iterate over the radii
    for i, radius in enumerate(radii):
        # Create a mask with the same dimensions as the image
        mask = np.full_like(image, fill_value=255,
dtype=np.uint8)

        # Draw a filled circle on the mask
        # cv.circle(mask, center, radius, (255, 100, 100), -1)
        cv.circle(mask, center, radius, (0, 0, 0), -1)

        # If it's not the first circle, remove the inner circle
        if i > 0:
            cv.circle(mask, center, radii[i-1], (255, 255, 255),
-1)

        # Apply the mask to the image
        masked_image = cv.bitwise_or(image, mask)
        heirarchy_area = np.count_nonzero(masked_image[:,:,0] !=
255)

        # Save the segmented image
        cv.imwrite(f'segmented_image_{i+1}.png', masked_image)
        segments_in_segment_circle(masked_image, heirarchy_area,
i+1)

segment_circle(original1, center, radius)
filename = str(imgname+'.png')
# # print(filename)
cv.imwrite(f"{filename}_circles.png", white)
print(radius)
```

```
v import openai
import json
```

```
def prepare_sunburst_prompt(data):
    prompt = "I have the following data of a sunburst chart, can
you generate a summary of 200-300 words? \n"
    # prompt = "Generate a paragraph describing the following
sunburst chart data:\n"
    for item in data:
        if 'title' in item:
            prompt += f"- Title: {item['title']}\n"
        else:
            prompt += f"- Segments:\n"
            prompt += f"  - Segment {data.index(item) + 1}:\n"
            prompt += f"    - Text: {item['text']}\n"
            prompt += f"    - Hierarchy: {item['hierarchy']}\n"
            prompt += f"    - Color: {item['color']}\n"
            prompt += f"    - Area: {item['area']}\n"
            prompt += f"    - Percentage:
{item['percentage']}%\n"
    return prompt

# Sample JSON data
json_data = [
    {'color': (165, 165, 165), 'hierarchy': 1, 'area': 13584,
'percentage': 32.98108448928121, 'text': 'Other'},
    {'title': 'Title'},
    {'color': (49, 125, 237), 'hierarchy': 1, 'area': 13794,
'percentage': 32.86002522068096, 'text': 'Other'},
    {'color': (196, 114, 68), 'hierarchy': 2, 'area': 23630,
'percentage': 32.00873425592527, 'text': 'Other'},
    {'title': 'Title'},
    {'color': (49, 125, 237), 'hierarchy': 3, 'area': 21026,
'percentage': 20.979078805920782, 'text': 'Other'},
    {'color': (165, 165, 165), 'hierarchy': 3, 'area': 9656,
'percentage': 9.644008930786406, 'text': 'Other'},
    {'color': (165, 165, 165), 'hierarchy': 3, 'area': 8067,
'percentage': 7.983957661457042, 'text': 'Other'},
    {'color': (49, 125, 237), 'hierarchy': 3, 'area': 8062,
'percentage': 7.916770032250062, 'text': 'Other'},
    {'color': (196, 114, 68), 'hierarchy': 3, 'area': 16336,
'percentage': 16.16637724303316, 'text': 'Other'},
    {'color': (165, 165, 165), 'hierarchy': 3, 'area': 14045,
'percentage': 14.23447448937402, 'text': 'Other'},
    {'title': 'Title'},
    {'color': (196, 114, 68), 'hierarchy': 3, 'area': 6431,
'percentage': 6.359050690482097, 'text': 'Other'}
]

# Prepare the prompt
prompt = prepare_sunburst_prompt(json_data)
# print(prompt)

# Set your API key
openai.api_key = 'sk-proj-
5fA15ADbFd3zf9gwso9ST3BlbkFJSjzLKrrxnDkPglJKb0oP'

# # Use the openai module to make API requests
# response = openai.completions.create(
#   model="gpt-3.5-turbo-instruct",
#   prompt="Write a tagline for an ice cream shop."
```

```
# )

# # Now you can work with the response
# print(response)

openai.sandbox = True

response = openai.completions.create(
  model="gpt-3.5-turbo-instruct",
  prompt= prompt,
  max_tokens=300
)

print(response)
print(response.choices[0].text)
```