# Augmenting Inputs using a Novel Figure-to-Text Pipeline to Assist Visual Language Models in Answering Scientific Domain Queries

by

Sejal Gupta

B.S. Computer Science & Engineering, MIT, 2023

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024



Authored by: Sejal Gupta
Department of Electrical Engineering and Computer Science
May 17, 2024

Certified by: Michael Cafarella
Principal Research Scientist, CSAIL, Thesis Supervisor

Accepted by: Katrina LaCurts
Chair
Master of Engineering Thesis Committee

# Augmenting Inputs using a Novel Figure-to-Text Pipeline to Assist Visual Language Models in Answering Scientific Domain Queries

by

Sejal Gupta

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

## ABSTRACT

Recent advancements in visual language models (VLMs) have transformed the way we interpret and interact with digital imagery, bridging the gap between visual and textual data. However, these models, like Bard, GPT4-v, and LLava, often struggle with specialized fields, particularly when processing scientific imagery such as plots and graphs in scientific literature.

In this thesis, we discuss the development of a pioneering reconstruction pipeline to extract metadata, regenerate plot data, and filter out extraneous noise like legends from plot images. Ultimately, the collected information is presented to the VLM in structured, textual manner to assist in answering domain specific queries. The efficacy of this pipeline is evaluated using a novel dataset comprised of scientific plots extracted from battery domain literature, alongside the existing benchmark datasets including PlotQA and ChartQA. Results about the component accuracy, task accuracy, and question-answering with augmented inputs to a VLM show promise in the future capabilities of this work.

By assisting VLMs with scientific imagery, we aim to not only enhance the capabilities of VLMs in specialized scientific areas but also to transform the performance of VLMs in domain specific areas as a whole. This thesis provides a detailed overview of the work, encompassing a literature review, methodology, results, and recommendations for future work.

Thesis supervisor: Michael Cafarella
Title: Principal Research Scientist, CSAIL

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In recent years, there has been remarkable advancement in visual language models (VLMs), revolutionizing the way we interpret and interact with images. Beginning with OpenAI's CLIP model, VLMs bridge the gap between visual and textual information. By tokenizing and understanding visual data, the models open a realm of multi-modal inputs to enhance the performance of machine learning models.

While the recent developments with Bard, GPT4-v, LLava, and many others have progressed the visual language model domain, their effectiveness is often contingent on the nature of the images they process. Predominantly trained on everyday objects for common tasks, these models excel in routine contexts but falter in specialized fields.

A notable shortcoming arises in the realm of scientific imagery, particularly in the interpretation of plots and graphs within scientific literature. Images from these domains can be challenging for the model, as they require common background knowledge, domain knowledge, and interpretation of the diagram.

As such, we developed a sophisticated scientific diagram extraction process, designed to augment VLMs' understanding of scientific plots. The core of our project lies in the development of a reconstruction pipeline that extracts the metadata, generates the plot data, and removes the noise to provide to the VLM in a structured format. Initially, a novel dataset with plots extracted from literature about lithium batteries is created; lithium battery literature has diverse plots that convey similar information, providing a clear area for comparison. With the utilization of diverse scientific plot datasets, this pipeline was evaluated using various metrics to analyze the performance of fine-tuned extraction model, the reconstructed image, and the VLM's ability to answer domain-specific questions.

By addressing the current limitations in processing scientific imagery, this project represents a significant step forward in the realm of VLMs, not only in scientific domains but also to open new avenues for knowledge dissemination and understanding for large multi-modal models.

In chapter 2, there is a review of previous literature regarding scientific diagram extraction and visual question answering. To introduce the problem and goals of the thesis, Chapter 3 describes the larger vision of the project, describing the parameters for success. Chapter 4 provides an overview of the work with a focus on the datasets used (4.1), the reconstruction pipeline (4.2), and the experimentation procedure (4.3). Chapter 5 outlines the metrics used to evaluate the pipeline's performance, specifically the task accuracy and the visual question

answering accuracy. Providing a glimpse of the project, Chapter 6 highlights the results for various parts of the pipeline. Finally, Chapter 7 provides a conclusion for the thesis.

The code for the project can be found at https://github.com/mitdbg/meng_code/. The data can be found at https://github.com/sejalgupta/scientific_figure_dataset. The annotation model can be accessed here https://tinyurl.com/annotation-detection.

# Chapter 2

# Related Works

This chapter presents an overview of the significant advancements and tools in the fields of visual language models, datasets of scientific figures, and scientific figure analysis tools.

## 2.1 Visual Language Models

Visual Language Models (VLMs) are multimodal models that combine computer vision and natural language processing into a single task. These models are trained on large datasets of images and text annotation pairs, allowing the model to learn to associate visual features with textual expressions. Through different training techniques, these models can learn to perform tasks such as Visual Question Answering (VQA), image captioning, and text-to-image search.

In recent years, there has been a significant surge in the popularity of VLMs due to their remarkable ability to forge connections between images and text. Notable among these models is CLIP [1], which has been influential in interpreting the connections between visual and textual data. CLIP is trained using a contrastive learning objective, where it learns to match images with their corresponding captions and distinguish them from mismatched pairs, enhancing its capability to understand and relate images and text.

While CLIP can provide captions to images and understand their content, more advanced VLMs like Flamingo [2] and ChatGPTv [3] bring new capabilities to the realm of visual language models. These models allow users to provide a prompt and image input, and the VLM will output a response by leveraging information from both the text and the image.

Flamingo, characterized by its unique architecture, processes images through a series of steps. First, the visual features are extracted using visual encoders, and the text is tokenized and processed through a pre-trained language model to create visual and textual embeddings. These visual and textual embeddings are then combined using gated cross-attention layers, which ensure that relevant information is highlighted and the modalities can interact effectively. Finally, the combined information is passed to a decoder, which generates an output sequence of tokens, forming a coherent response to the user.

These models perform exceedingly well on everyday objects, as they were trained on a plethora of everyday images. However, questions about domain-specific images, such as scientific diagrams, present more challenges for these models. Addressing these challenges

involves improving the model's ability to understand and interpret specialized visual information, which is critical for enhancing VLM performance in scientific and technical domains.

In the following sections, we will delve deeper into the different components of improving a VLM's performance on scientific diagrams, focusing on both the existing datasets and tools available for the task.

## 2.2 Datasets of Scientific Figures and Questions

The development of models that can analyze and interpret scientific figures has been bolstered by the availability of specialized datasets. ChartQA, PlotQA, and FigureQA are prime examples of such datasets. They provide a rich source of data for training and evaluating models designed to understand and interact with scientific charts and figures. These datasets have been pivotal in advancing research in the field of scientific figure analysis.

Researchers created ChartQA [4], a dataset of chart images paired with questions and answers about the data represented in these charts. Their hope was to assist models in understanding and interacting with various types of charts: bar charts, line graphs, and pie charts. Although ChartQA played a crucial role in establishing a range of questions from inquiries about individual data points to identifying trends, the charts in the dataset are uniform and basic, which does not accurately reflect the diverse imagery found in scientific publications.

Similar to ChartQA, PlotQA [5] focuses on plots typically found in scientific contexts, including more complex graph types, and FigureQA [6] focuses on broader range of scientific figures, beyond just charts and plots. However, all the figures are similar in color schemes, fonts, and overall design decisions to the rest of the figures in the dataset. Since these charts were mass generated, they have similar characteristics which should not be learned by a model. Therefore, we will develop a dataset that incorporates actual figures from scientific papers and bases the questions on those figures.

A recent paper introduces a new benchmark dataset [7] with realistic plot images from the Reticular Chemistry domain. Their small dataset of approximately 6,000 images categories the plots in five categories: Nitrogen Isotherm, Power X-Ray Diffraction, Thermogravimetric Analysis, Crystal Structure or Topology 98.1, and Other Gas Sorption Isotherm. As such, the solution is contained to this domain and is not representative of other domains, specifically the battery domain. In the battery domain, most figures have annotations that are important for understanding the plot, but they do not contribute to the datasets. Existing solutions do not have a focus on the removal of these annotations. Thus, we plan to create a labelled dataset of scientific figures in the battery domain, specifically focusing on the annotations that are present in the chart area.

## 2.3 Scientific Figure Analysis Tools

Scientific plot analysis has evolved through the contributions of various models and tools, each addressing different aspects of chart and plot interpretation. Parsing Line Charts [8], ChartOCR [9], ChartReader [10], FigureSeer [11], and Classification-Regression for Chart

Comprehension [12] represent a diverse range of approaches and methodologies for analyzing scientific plots. Some emphasizing the parsing of the charts, while others are more oriented towards OCR capabilities or holistic chart analysis. Unfortunately, most of these tools only work on a specific dataset, have limited abilities, or cannot accurately answer questions. Thus, the proposed method focuses primarily on extraction, so current-state-of-the-art VLMs can best interpret the data to make accurate claims on complex scientific figures.

Like the proposed method, tools, like MaTcha [13], focus on the pre-training of models for data extraction and figure creation through code. MaTcha beat all the existing competitors on the PlotQA and ChartQA datasets, but they attribute over 43% of their errors to data extraction. Whenever they could not extract the correct data series, the question-answering regarding the plots was quite inaccurate. Therefore, our project will focus its efforts on improving the data extraction pipeline to improve the question answering accuracy.

Furthermore, DeepMind's Deplot [14] claims to be able to extract charts into datasets in CSV form to assist the LLM. While they attempt to decompose the plot into text, Deplot struggles to perform well on complex plots, specifically multiple line plots. This may be due to the fact that all layout information such as orientation and color of the visual elements/objects is not captured within the model. Similar to Deplot, this project will focus on the conversion of plot images to text to pass into a VLM, but the layout information and color will play a large role in the data extraction pipeline.

Recently, researchers at Berkeley [7] released a new tool to convert scientific diagrams in the Reticular Chemistry domain to CSV files. Their innovative approach involves labelling the contents of the diagrams with GPT4v and utilizing a semiautomatic tool called WebPlotDigitizer [15] to obtain the dataset. However, since WebPlotDigitizer requires manual entry of a few points and the axes, their solution cannot extract the data and differentiate between data series without human intervention. Therefore, our solution will similarly leverage GPT4v to extract metadata components in the figures but will focus on an automated data extraction solution to retrieve the data series without human intervention.

This chapter has provided an overview of the key developments in visual language models, datasets of scientific figures, and scientific figure analysis tools. As seen through this comprehensive review of previous literature, an automated conversion tool from diagram to CSV tool is missing for realistic scientific figures. Existing works are analyzed on synthetic datasets that are uniform and simplistic in design, inaccurately describing the true performance of tools. Even though a new paper provides some realistic diagrams, most of the plots are structured in a similar way, making a general solution challenging. Therefore, our work bridges the gap by introducing a realistic dataset with figures from published work with annotations. Additionally, most of the plot-to-csv tools try to improve both the extraction and LLM component, or the tools involve manual extraction. Our work will leverage the advancements in VLMs and enhance the data extraction procedure to obtain the best results possible, a solution that has not been explored previously.

# Chapter 3

# Problem Statement

Visual Language Models (VLMs) have demonstrated difficulty in answering specific questions about scientific figures. To enhance their performance, data needs to be accurately extracted from these figures. Previous plot-to-CSV solutions often fall short, especially when dealing with complex figures, such as those found in battery literature. Among existing solutions, WebPlotDigitizer [15] stands out as the best, but its semi-automated nature requires a human-in-the-loop, which makes high-throughput extractions challenging. This thesis will focus on the automatic extraction of data from scientific figures to improve the accuracy of question answering by VLMs.

The input to our system is an image of a scientific figure, and the output is a CSV file containing the extracted data points. The system operates in two main stages: classification and decomposition, as shown in Figure 3.1.

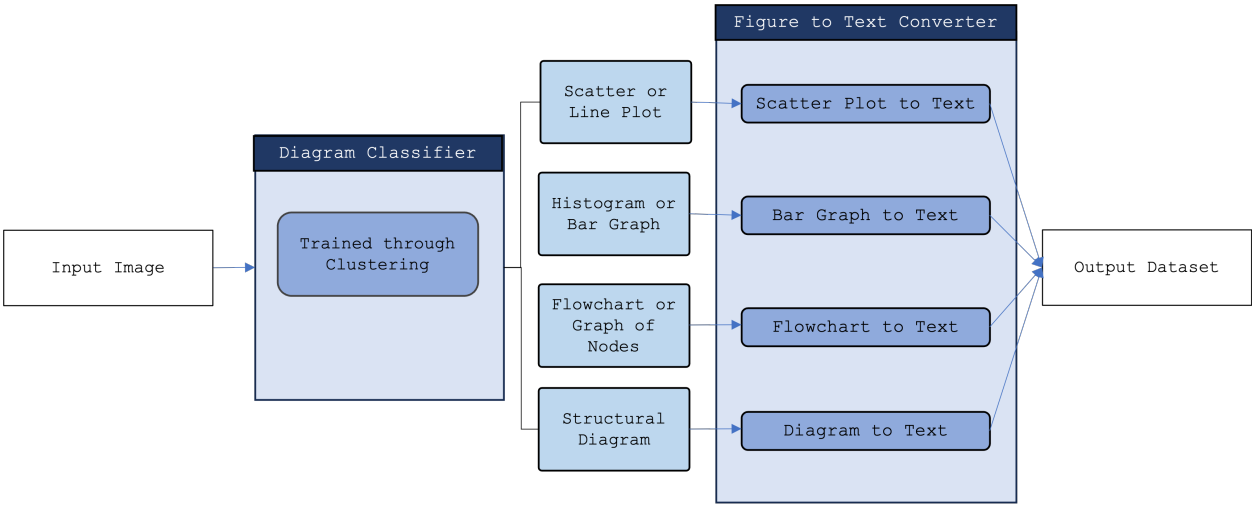



Figure 3.1: Architecture for Reconstruction of a Scientific Diagram

In the first stage, the system classifies the figure into a specific category based on its visual characteristics. This classification is crucial because different types of figures (e.g., bar graphs, scatter plots, line plots) require different extraction techniques. The classifier analyzes the figure and assigns it to a generic (non-domain specific) categorical label. For instance, bar graphs will be grouped in cluster 1, and scatter/line plots will be placed in

cluster 2. This process is essential for ensuring that the appropriate extraction pipeline is triggered for each figure type, as depicted in Figure 3.1.

Once the classifier assigns a label to an image, the figure undergoes a figure-to-text conversion process specific to its category. This stage involves decomposing the image into a structured data format, specifically a CSV file. The CSV file will contain all relevant data points extracted from the figure, organized in a tabular format. For the scope of this thesis, we focus on scatter plots and line plots, converting them into CSV format.

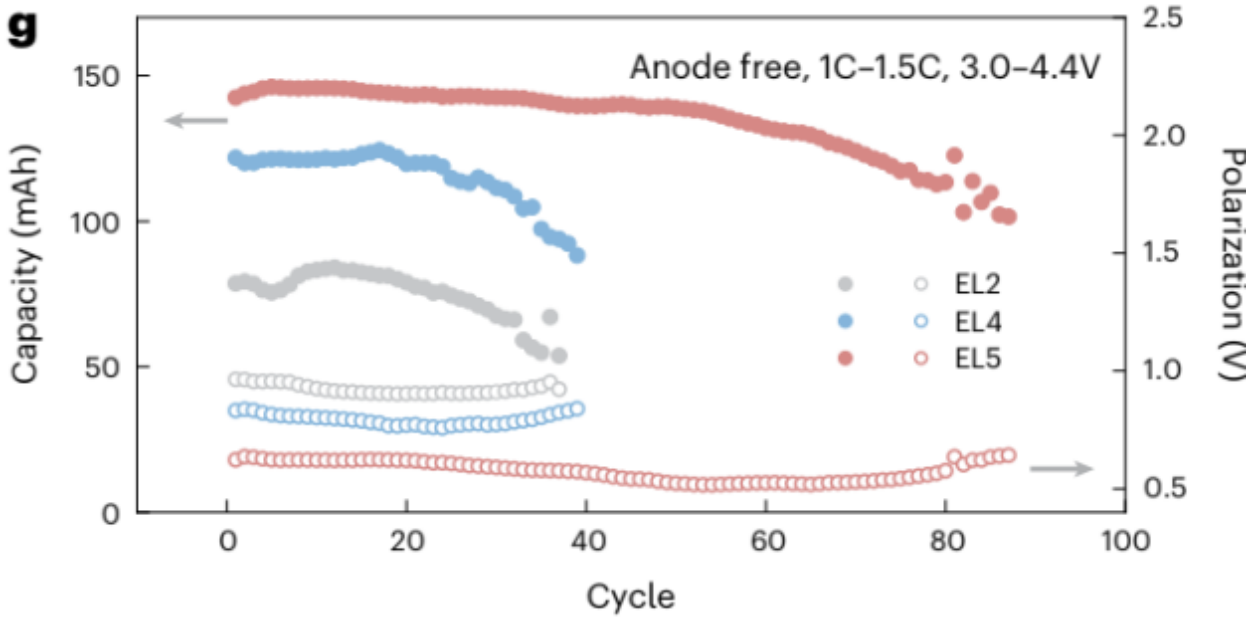



Figure 3.2: Example Input Image

Consider the image in Figure 3.2. The classifier assigns it the label "Scatter Plot." Based on this classification, the image is processed through a scatter/line plot reconstruction pipeline. This pipeline involves several steps:

- Detection: Identifying the axes (location, titles, and ranges), any legends or other annotations present in the plot, and the data series.
- Extraction: Extracting the coordinates of each data point and converting them into numerical values.
- Clustering: Group the data points by the legend item (if applicable).
- Structuring: Organizing the extracted data points into a CSV format, with each row representing a data point and each column representing a specific attribute (x-value, y-value, second y-value, legend item / data series title).

The output of this process is a CSV file containing the data points from the scatter plot. An example of such a CSV file is illustrated in Appendix D. This CSV file, along with the related question, is then used to prompt the VLM.

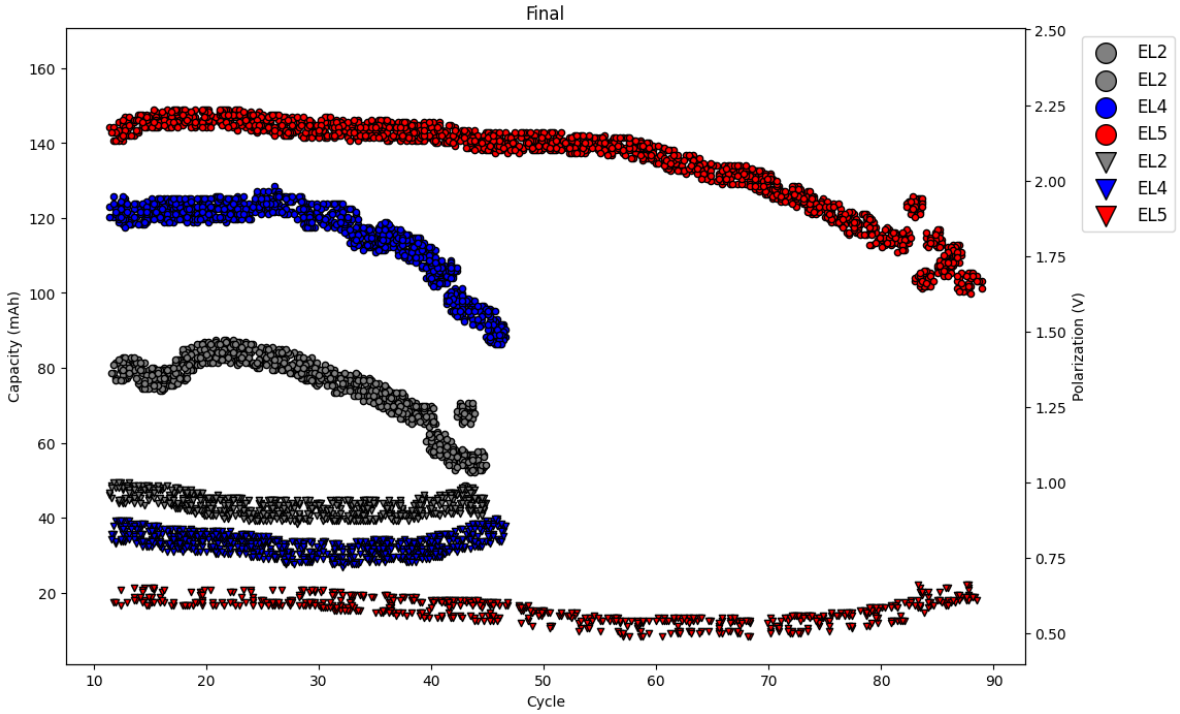



Figure 3.3: Assignment of Data Clusters to Axes for Image $I_0$ Where Circle Correspond to the Left Axis and Triangle Corresponds to the Right Axis

The successful implementation of this reconstruction pipeline involves taking an input image of a scatter or line plot and outputting a CSV file of the points. To measure the accuracy of this system, we will use various metrics:

- Completeness: The extent to which all data points in the figure are accurately extracted and included in the CSV file. For Figure 3.2, the pipeline should extract all the data points in EL2 closed circle, EL2 open circle, EL4 closed circle, EL4 open circle, EL5 closed circle, and EL5 open circle. The pipeline should not extract and points outside of those data series.

- Assignment: Correctly assigning extracted data points to the appropriate legend items (if applicable). For Figure 3.2, each of the data series (EL2 closed circle, EL2 open circle, EL4 closed circle, EL4 open circle, EL5 closed circle, and EL5 open circle) should be clustered in independently. All the closed circle data series should be assigned to the y-axis (on the left), but all the open circle data series should assigned to the second y-axis (on the right).

- Correctness: The accuracy of the extracted data points, ensuring that their values match the original figure. Each of the data points in Figure 3.2 should be in terms of axis ranges defined in the plot. For Figure 3.2, the data points should scaled to the x-axis range of approximately [-10, 100], the y-axis range of approximately [0, 165], and the second y-axis range of approximately [0.4, 2.5].

For Figure 3.2, a successful reconstruction will look like Figure 3.3 if the output CSV file was replotted.

Ultimately, the effectiveness of the system will be evaluated based on the percentage of questions answered correctly by the VLM when using the extracted data. This metric will demonstrate the practical impact of our system on improving the VLM's performance in interpreting and answering questions about scientific figures.

# Chapter 4

# Methodology

Given the limitations of previous work in scientific plot question answering, we propose a novel extraction process to assist visual language models (VLMs). Initially, two datasets will be hand created for the tasks of reconstruction and visual question answering. From there, the reconstruction pipeline is constructed. Finally, various experiments will be carried out to test the accuracy of the reconstruction pipeline on the data extraction and visual question answering tasks.

## 4.1 Dataset

The development of our pipeline necessitates the use of various datasets, both existing and newly generated. Our focus is currently on the creation of datasets for scatter and line plots.

### 4.1.1 Scientific Figure Dataset

While previous datasets of charts are quite large, the plots are all relatively similar and do not reflect the complexity of true scientific plots. Therefore, we will create a new benchmark dataset to represent the figures in the battery literature domain.

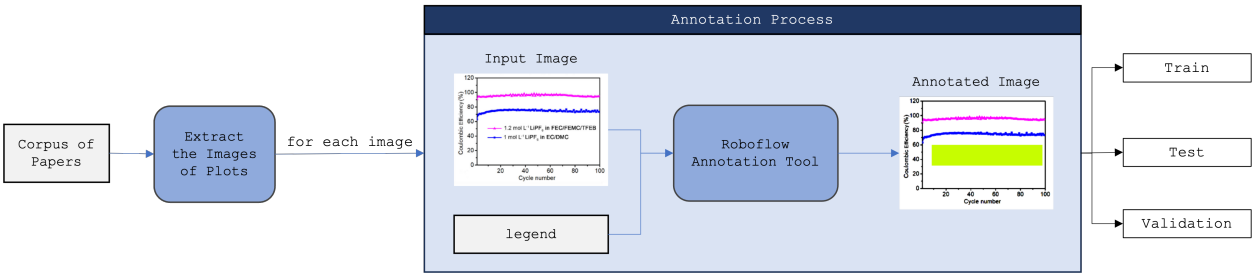



Figure 4.1: Scientific Figure Dataset Creation Pipeline

To create the dataset, we compile a large corpus of papers in the battery domain that have similar figures, as seen in Figure 4.1. From these papers, we extract several scatter and line plots to create a collection of plot images representative of those the tool will process during runtime. Each image in the dataset will be manually annotated using Roboflow's

tool, focusing on elements such as legends and arrows. After all the images are annotated, the images will be split into train, validation, and test datasets.

The train and validation datasets will be crucial in the creation of the annotation detection model. However, the evaluation will be solely done on the test dataset. In the complete dataset, each image will be paired with a list of the bounding boxes of the annotations in the image.

### 4.1.2 Scientific Figure Visual Question Answering Dataset

Using the images from the test dataset of the Scientific Figure Dataset, we randomly sample 10 of the images. These images are all from the battery papers and require similar domain knowledge. While they can be interpreted using background knowledge about batteries, many questions regarding the graphical structure are still applicable.

For each of the images, there will be 3 question-answer pairs generated by hand about the plot information provided.

## 4.2 Scientific Figure Reconstruction Pipeline

The reconstruction pipeline consists of several modules, including Image Augmentation, Annotation Mask Extraction, Bounding Box Mask Extraction, Metadata Extraction, Color Mask Extraction, Axis Recalculation, Data Point Extraction, Dual Axis Partition of Data, and Assignment of Data Clusters to Axes.

In the following subsections, each of the aforementioned modules are described in detail with an example image input, Figure 3.2. The original image will be referred as $I_0$.

### 4.2.1 Bounding Box Mask Extraction

The image, $I_0$, is first passed through the bounding box extractor module. To isolate the chart elements such as titles and data series, a bounding box around the chart area must be identified. A bounding box is defined as

$$[(x_{min}, y_{min}), (x_{max}, y_{max})]$$

where $(x_{min}, y_{min})$ is the top left corner of the bounding box and $(x_{max}, y_{max})$ is the bottom right corner.

As a majority of images of scientific plots have a border around the chart area, a traditional, computer vision approach will be utilized as an initial attempt, to obtain the bounding box corners. The input image, $I_0$, will be converted to grayscale and passed through Canny Edge Detection to define all the edges in the image. These procedures are done in preparation for the Hough Line Transform which detects all the straight lines in a given image. After the transform is applied, the following procedure is used to identify the bounding box:

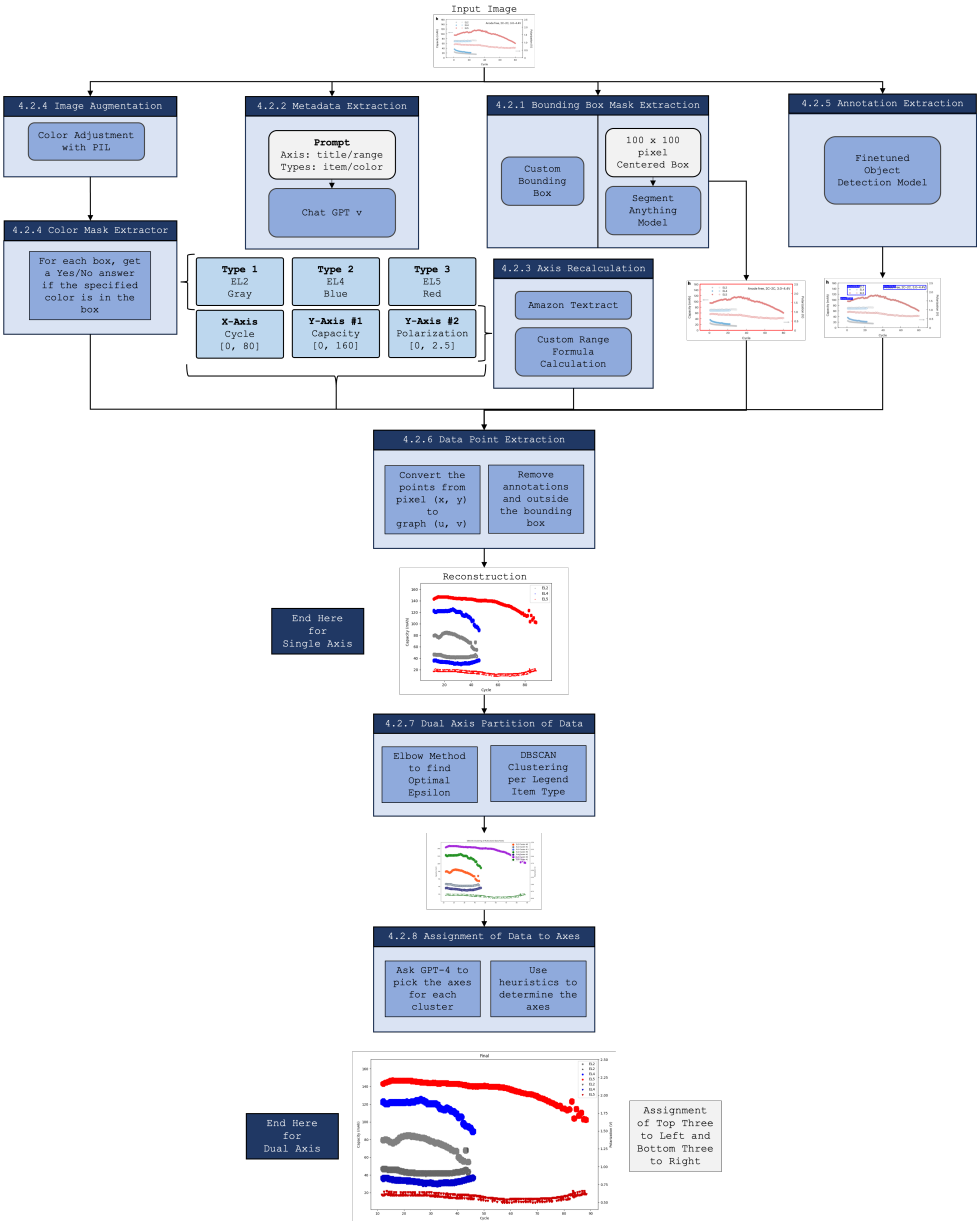



Figure 4.2: Pipeline for Reconstruction for a Scatter Plot using Color Analysis

1. Identify all the horizontal lines that are within 5° of a perfectly horizontal line.
2. Combine horizontal lines that are at the same $y$ to get the longest continuous line at a specific $y$.
3. Sort the resulting combined lines and get the two longest lines.
4. Extend those lines to span the entire image in the $y$ direction.
5. Repeat steps 1 - 4 for vertical lines.
6. Mark all the intersection points between the lines.

The procedure forms a bounding box, as seen in Figure **??**. If the area of the bounding box is greater than 50% of the area of the image, the bounding box is accepted. However, in the case where the bounding box is not determined or the area is less than 50% of the image, the input image, $I_0$, is passed through an alternate approach. Even though the bounding box in Figure **??** would be accepted, for the purposes of explanation, the alternate approach will be shown with $I_0$.

The input points are defined as

$$\left(\frac{w}{2} - 50, \frac{h}{2} - 50\right)$$
$$\left(\frac{w}{2} + 50, \frac{h}{2} - 50\right)$$
$$\left(\frac{w}{2} - 50, \frac{h}{2} + 50\right)$$
$$\left(\frac{w}{2} + 50, \frac{h}{2} + 50\right)$$

where $w$ is the width of the image and $h$ is the height of the image. We assume that the chart area overlaps with the center of the image as such these input points form a 100 x 100-pixel centered box on $I_0$. The input points and $I_0$ are passed to Meta's Segment Anything Model [16] to mask the chart area, as seen in Figure 4.3. Once the binary mask around the chart area is retrieved from the model, the mask is applied and a bounding box is constructed from the outline of the mask.

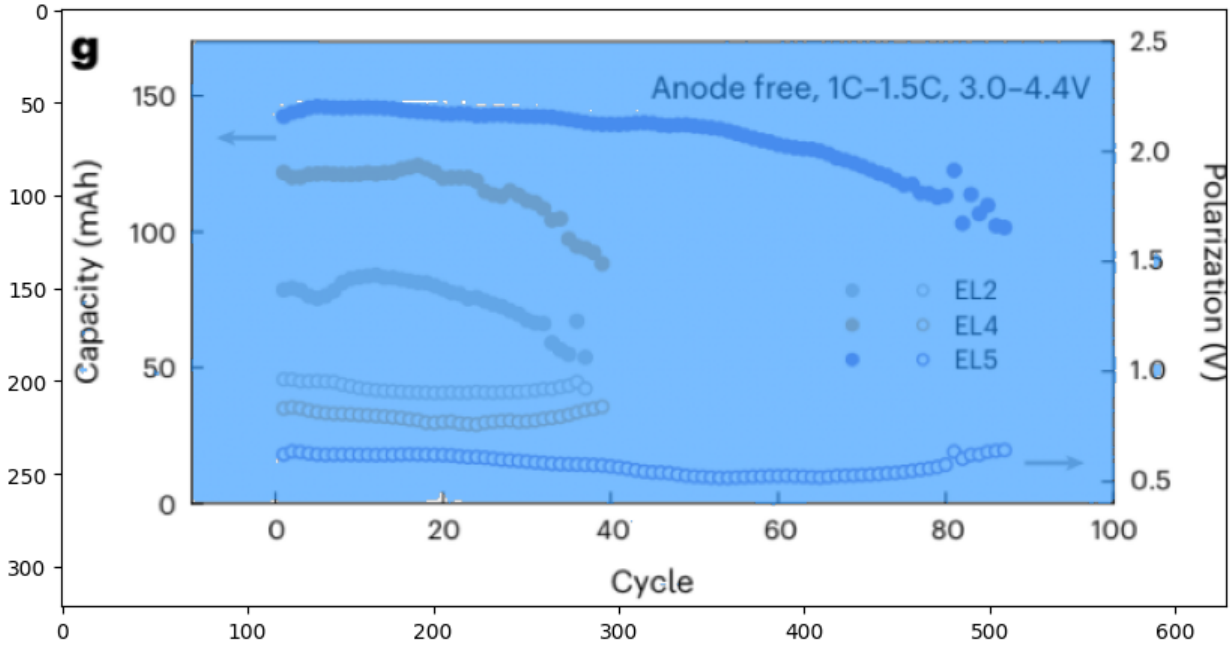



Figure 4.3: Segment Anything Chart Area Mask on Input Image $I_0$

As a result of either method, a bounding box of the chart area will be extracted. For $I_0$, we expect a bounding box of [(78, 14), (574, 266)].

### 4.2.2 Metadata Extraction

Simultaneously to the Bounding Box Mask Extraction, the input image and a prompt are sent to ChatGPT's vision model [3]. The specific prompt sent to ChatGPT is available in Appendix A.

ChatGPT extracts metadata from the input image, including titles and ranges for each axis, as well as legend items with corresponding marker colors. Due to the lack of accuracy with ChatGPT's color recognition, the marker colors are restricted to be one of the following: black, white, red, purple, green, yellow, blue, pink, orange, and grey.

For example, if ChatGPT was provided the prompt and $I_0$, the following information should be extraction in the expected JSON format outlined in the prompt:

- X-axis Title: Cycle
- X-axis Range: [0, 100]
- Y-axis Title: Capacity (mAh)
- Y-axis Range: [0, 150]
- Second Y-axis Title: Polarization (V)
- Second Y-axis Range: [0.5, 2.5]
- List of Legend Items with the Marker Color and Legend Item Title: [(EL2, Gray), (EL4, Blue), (EL5, Red)]

### 4.2.3 Axis Recalculation

The metadata extraction provides a baseline for the axis ranges. However, ChatGPT is only able to extract the values from the image provided. This can pose issues when the entire axis is not properly defined from the beginning to the end. For instance, in image $I_0$, the 0 tick mark on the x-axis is not directly aligned with the y-axis. As such, the extracted x-axis from ChatGPT is not accurate.

To rectify the issues with the ranges, the true ranges for each of the axes must be recalculated. First, Optical Character Recognition (OCR) must be performed to get the bounding boxes of the text in the image. A variety of tools exist for OCR such as Keras, Pytesseract, and Amazon Textract. The methods will be tested to find the optimal tool. The results from Amazon Textract on image $I_0$ are provided in Figure 4.4.

The custom algorithm iterates through OCR results to find OCR detected text that is numerical, the shortest distance from the axis endpoints, and positioned in the correct spot for the particular axis. For the x-axis, the correct position is below the axis. For the y-axis, the correct position is left of the axis. For the second y-axis, the correct position is right of the axis. The recalculation requires two bounding boxes on the axis.

Using a similar method to the authors in ChartOCR, the axes are recalculated using the pixel locations and the bounding boxes. The formulas for the first y-axis are provided, but the axis recalculation formulas for the second y-axis and x-axis are similar.

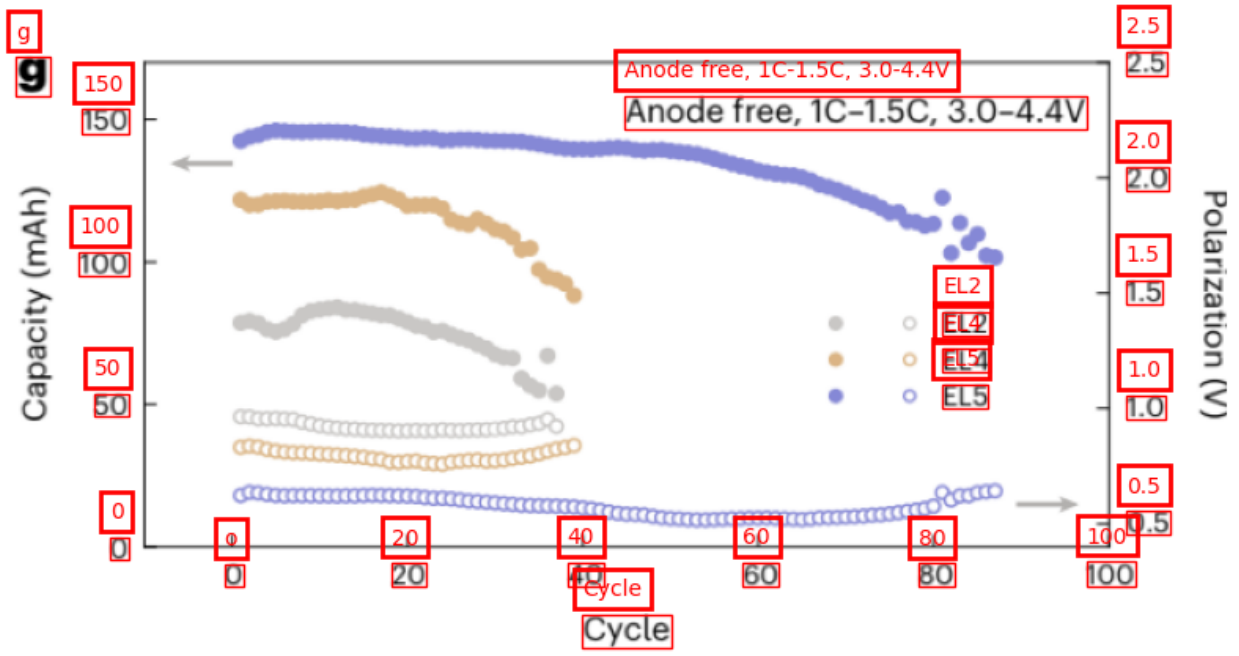



Figure 4.4: OCR Results on Example Input Image $I_0$

First, the pixels per unit is calculated by taking the ratio between the pixel difference between the middle of the two bounding boxes and the difference between the numerical values of the detected text.

$$ratio_{pixelsPerUnit} = \frac{Y_{p_{max}} - Y_{p_{min}}}{Y_{v_{max}} - Y_{v_{min}}}$$

To rescale the axis, the pixel length of the axis needs to be determined from the bounding box extracted earlier.

$$lengthOfYAxis = y_{max} \breve{} y_{min}$$

Therefore, the difference in the range of the axis is the ratio between the length of the axis in pixels and the pixels per unit calculated previously.

$$differenceTrueRange = \frac{lengthOfYAxis}{pixelsPerUnit}$$

Finally, the minimum and maximum of the range are calculated using the calculated values.

$$true_{y_{min}} = Y_{v_{min}} - \frac{(Y_{p_{min}} - y_{min})}{ratio_{pixelsPerUnit}}$$

$$true_{y_{max}} = true_{y_{min}} + differenceTrueRange$$

By default, all the ranges are from 0 to 100 in case the tick marks are not specified in the plot or the recalculation is not possible.

For image $I_0$,

- $ratio_{pixelsPerUnit} = -1.466$

- $lengthOfYAxis = 249$
- $differenceTrueRange = 169.849$

Ultimately, for image $I_0$, the true y range is [0.937, 170.798]. Similarly, the x range is [2.393, 99.865], and the second y range is [0.398, 2.506]. While these values are not perfectly accurate, the range is closer to complete accuracy.

### 4.2.4 Image Augmentation and Color Mask Extractor

Each of the legend items corresponds to a marker color. With the marker colors from the metadata extraction and the input image, masks for each of the colors can be created. Unfortunately, the marker shape and hue will not be differentiated through this method.

The input image is passed through an Image Augmentation module. In the module, image is altered through a series of transformations like sharpen, contrast, and filter. To determine the best set of transformations, experiments on the test dataset will reveal the ideal transformation.

Using the legend information and the transformed image, a color analysis tool will generate a mask of the existence of the colors in a plot. Various color analysis methods will be explored to determine the best color extraction method.

#### Color Analysis Methods

Each pixel in the image has numerical value for each of the red, green, and blue channels. In the metadata extraction, each item in the legend is paired with a specific color like "red," "green," or "blue". Originally, we attempted to request ChatGPT [3] to provide the colors in RGB format or HEX format. However, this method was unsuccessful because ChatGPT [3] struggles to map specific pixels to the larger objects, resulting in inaccurate color schemes. Additionally, ChatGPT can only accurately provide the names of simple colors. For instance, it will identify the color firebrick as red. Therefore, a color analysis method must be utilized to convert the RGB pixel values to ChatGPT colors.

Looking at Figure 4.5 from GeeksforGeeks [17], to convert the RGB values to a simple color, we must employ a clustering mechanism, since the color scheme can be visualized as a cube and the vertices are all different "simple" colors. Three potential methods for color conversion are proposed: linear search using Euclidean distance, nearest neighbor clustering using cosine similarity, and KDtree search for nearest color.

Linear search using Euclidean distance finds the distance between the pixel value and a predefined set of colors with their respective RGB values. To determine the closest neighbor, for each predefined color, the distance is calculated using the following equation:

$$d = \sqrt{(r_c - r_p)^2 + (g_c - g_p)^2 + (b_c - b_p)^2}$$

where $(r_c, g_c, b_c)$ are the RGB values for the centroid and $(r_p, g_p, b_p)$ are the RGB values for the pixel. The color with the minimum distance is returned. Since colors in grayscale (black, white, and gray) are harder to defined without disrupting the other colors, we expect challenges with picking the specific centroids (predefined RGB values of the colors).

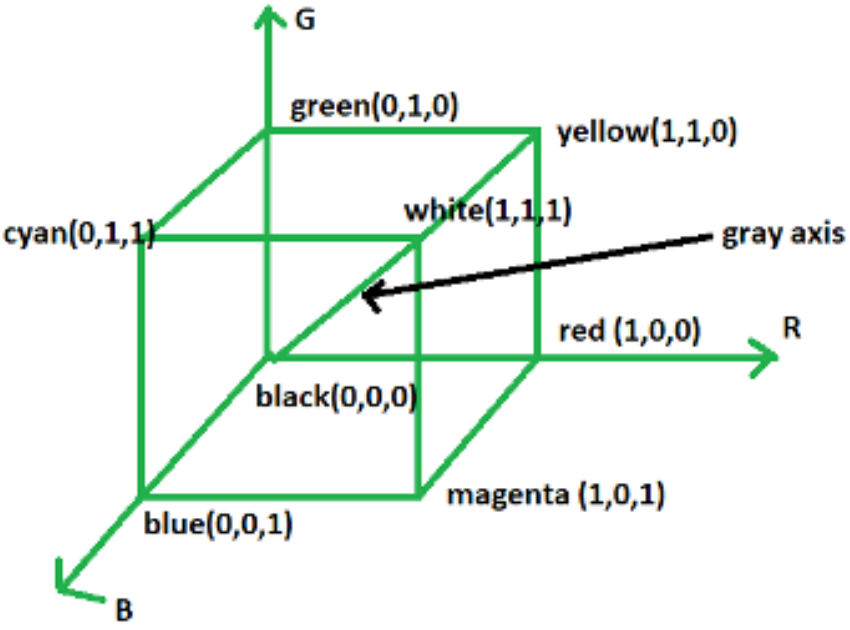



Figure 4.5: Color Space

Linear Search using cosine similarity finds the cosine of the angle between the RGB vector of the pixel and the predefined colors. To determine the closest neighbor, for each predefined color, the similarity is calculated using the following equation:

$$similarity = \frac{\vec{A} * \vec{B}}{||\vec{A}|| * ||\vec{B}||}$$

where $\vec{A}$ and $\vec{B}$ are the RGB vectors of the predefined color and the pixel, respectively. Rather than the importance of magnitude for Euclidean distance, cosine similarity focuses more on the direction, an indication of hue, of the colors. The color with the highest similarity (closest to 1) is the most similar color. We expect that this method may be more effective in distinguishing colors with similar intensities but different hues rather than different colors.

A KDtree (k-dimensional tree) uses a space-partitioning data structure for organizing the predefined set of colors in the RGB space. Once the tree is built, KDtree Search for Nearest Color can be used to efficiently find the nearest neighbor to a given pixel's RGB value by searching for the closest point (color) in the tree, using Euclidean distance. This method is computationally more efficient than a linear search, especially for large sets of predefined colors, and allows for more precision with colors. Using the KDtree, we can utilize the larger dictionary of CSS3 [18] and hand map the large set of colors to a smaller set of simple colors.

While there are advantages and disadvantages to each of the methods, the experiments will show which method is the most effective for reconstruction.

### 4.2.5 Annotation Extraction

Most of the images of the plots have quite a bit of noise due to the presence of annotations. Annotations are any markings on the image that lie within the chart area. The most com-

mon examples of these annotations are legends, arrows, and free text. The color extractor mentioned earlier will pick up all the pixels that have color. To filter out the pixels that are annotations, an annotation extractor must be employed on the input image.

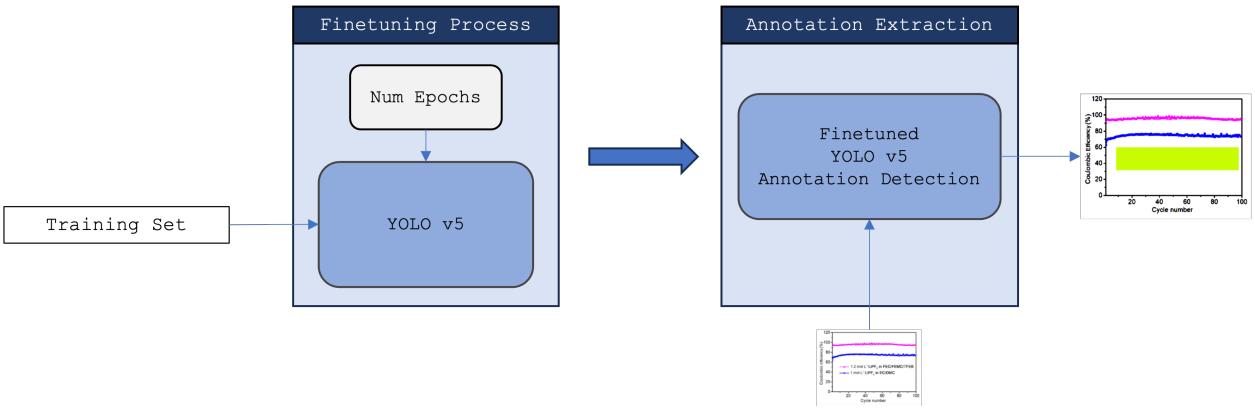



Figure 4.6: Fine-tuning Procedure for the Annotation Detection Model

Since existing object detection models cannot accurately find all the annotations in an image of a plot, a small object detection model will be fine-tuned to detect the annotations in the image. We will fine-tune a YOLOv5 object detection model using the training dataset from the Scientific Figure Dataset4.1.1, deploying it via Roboflow for inference. The fine tuning process is depicted in Figure 4.6.

The input image will be passed into the Annotation Detection Model, resulting in a mask of all the bounding boxes of the annotations that are detection. Figure 4.7 depicts the expected mask for input image $I_0$ from the Annotation Detection Model.

### 4.2.6 Data Point Extraction

The final step for a single y-axis plot image is to convert points from the masks into the true axes. First, the colors masks from the Color Mask Extractor are pruned to only contain pixels within the bounding box extracted by the Bounding Box Mask Extraction process and exclude the pixels inside of the annotations found by the Annotation Detector Model. Next, the true axis ranges for the x and y axis are passed in from the Axis Recalculation step through a graph reconstruction tool that converts the pixel coordinates into Cartesian points of the original graph. This conversion tool uses the following formulas to convert the x and y values:

$$lengthOfYAxis = y_{max} - y_{min}$$

$$lengthOfXAxis = x_{max} - x_{min}$$

$$plot_x = \frac{(p_x - x_{min}) * (true_{x_{max}} - true_{x_{min}})}{lengthOfXAxis} + true_{x_{min}}$$

$$plot_y = \frac{(y_{max} - p_y) * (true_{y_{max}} - true_{y_{min}})}{lengthOfYAxis} + true_{y_{min}}$$

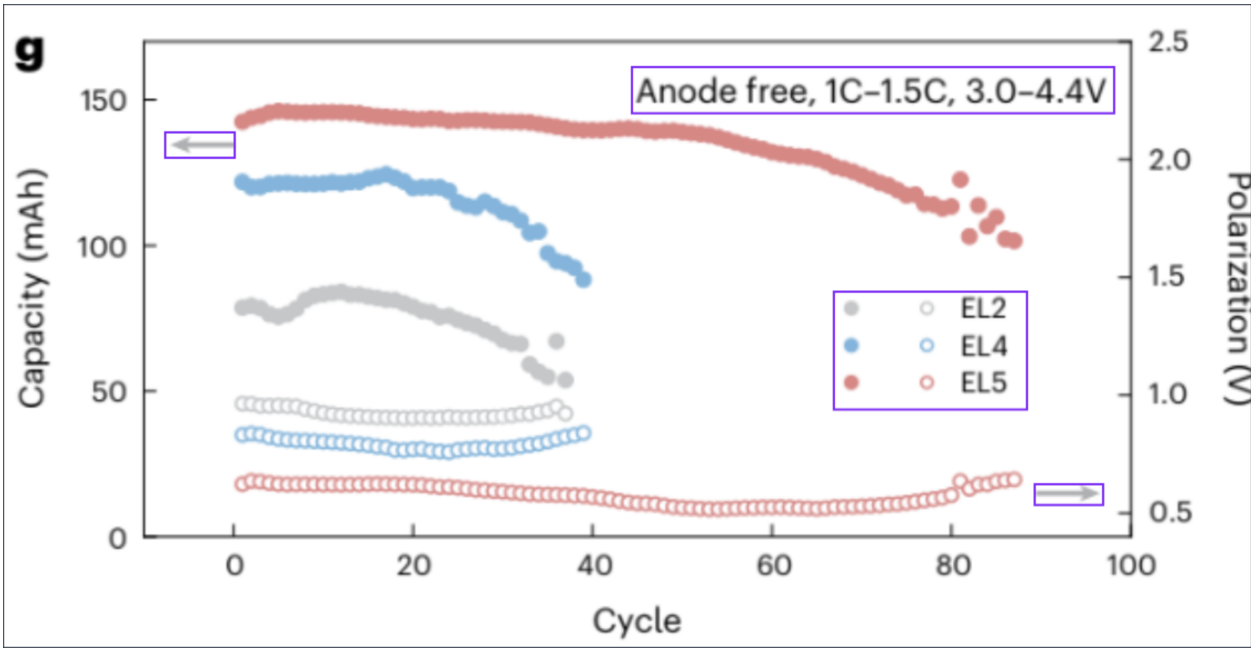



Figure 4.7: Annotations on Image $I_0$

where $p_x$ is the $x$ value of the pixel, $p_y$ is the $y$ value of the pixel, $x_{min}$ is the $x$ coordinate of the chart area's top left point, $y_{max}$ is the $y$ coordinate of the chart area's bottom most point, $true_{x_{min}}$ is the start of the range on the $x$ axis, $true_{x_{max}}$ is the end of the range on the $x$ axis, $true_{y_{min}}$ is the start of the range on the $y$ axis, $true_{y_{max}}$ is the end of the range on the $y$ axis, $plot_x$ is the $x$ coordinate of the pixel on the actual plot, and $plot_y$ is the $y$ coordinate of the pixel on the actual plot.

With the metadata retrieved from the Metadata Extraction module, the data series and chart elements are labelled to match the original image. For image $I_0$, the final single axis reconstruction is illustrated in Figure 4.8. If the image has a single axis, this is the final step of the pipeline. In this case, image $I_0$ has a second y-axis, so the image will continue down the pipeline.

### 4.2.7 Dual Axis Partition of Data

As seen in Figure 4.8, the two data series for each color are assigned automatically to the left axis. To correctly assign the data to an axis, the data series need to be partitioned between the left and right axis. Since the data is already partitioned into the legend items, the data for each legend item can be clustered independently.

Many unsupervised clustering algorithms exist like KMeans and density-based clustering non-parametric algorithm, DBSCAN. Since (1) the data is prone to outliers, (2) the exact shape is not known before, and (3) the data will most likely not be in a spherical format, KMeans is not the ideal algorithm for this use case, even though the number of clusters can be defined to two. To show the issues, Figure 4.9 provides the clusters created on image $I_0$. The gray clusters are incorrect due to the location of the centroids, so the outliers really impact the formation.

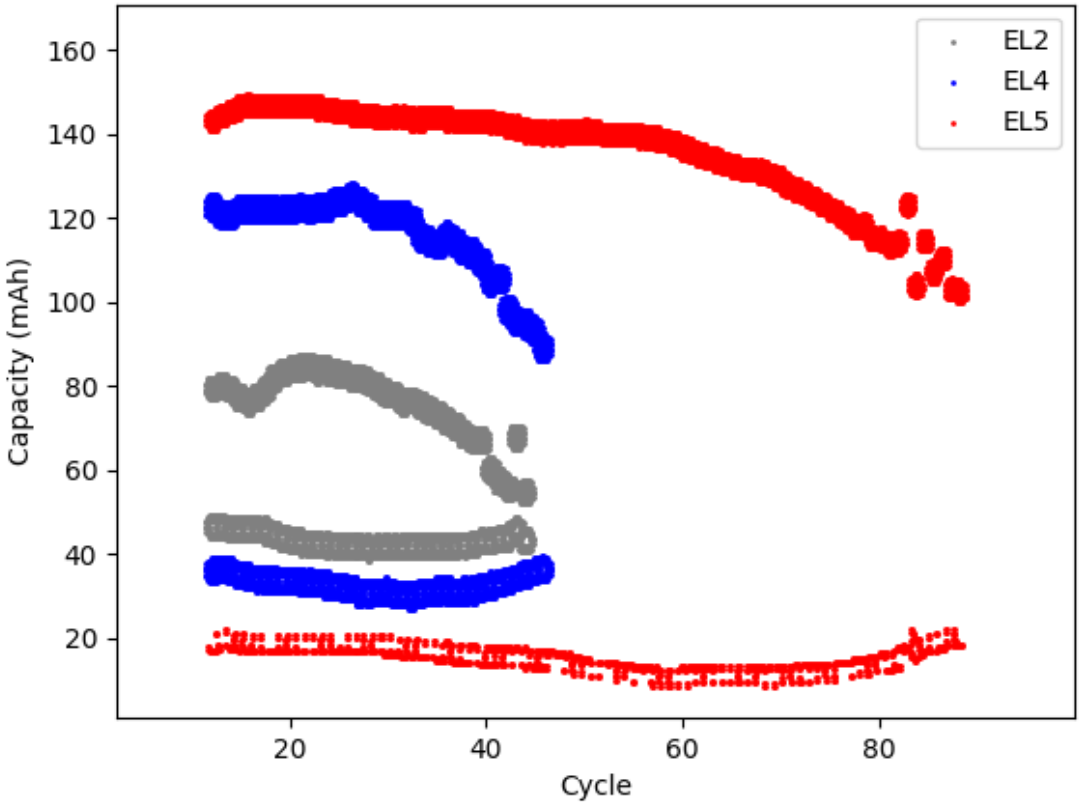
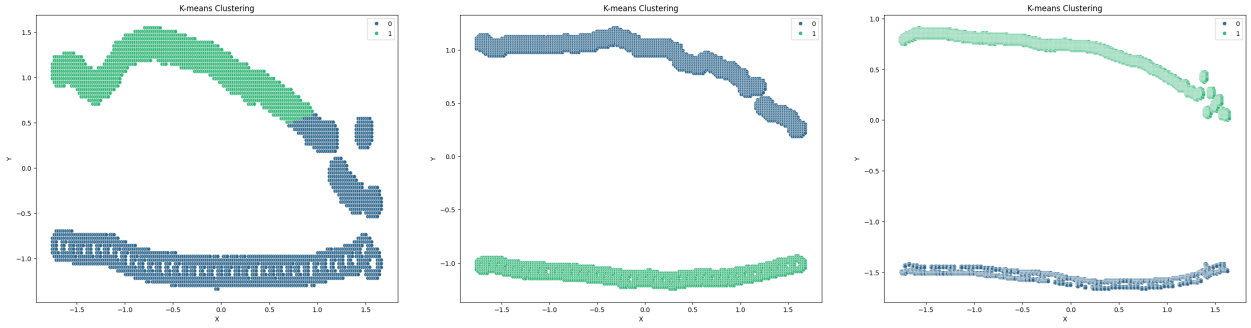


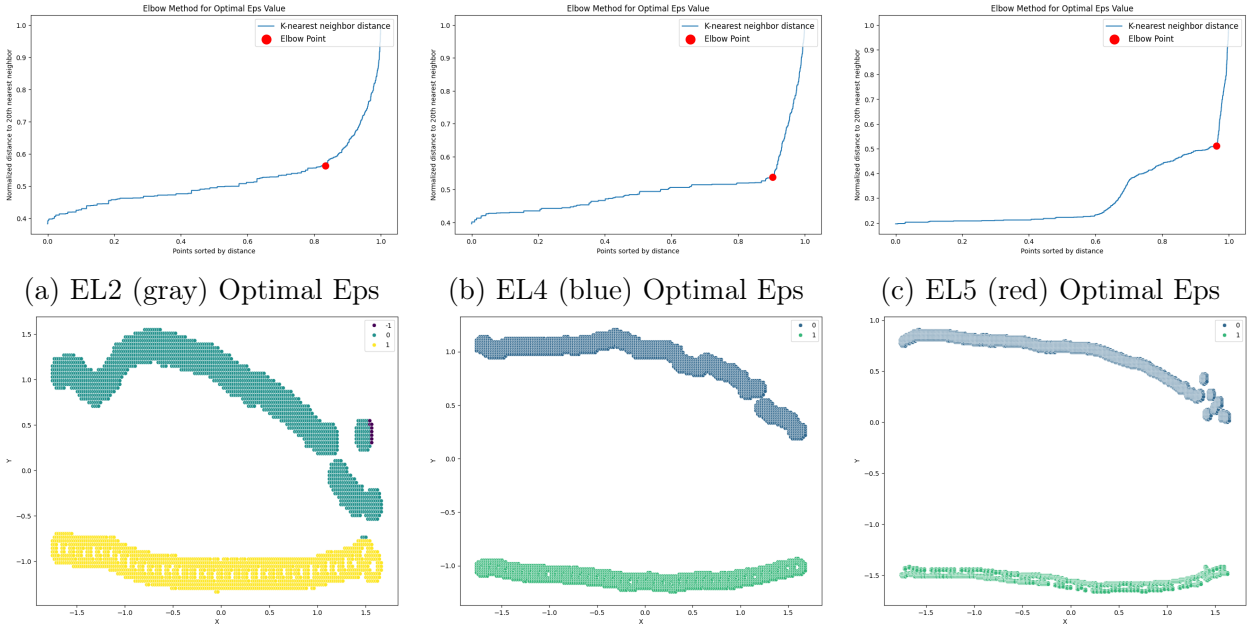

Figure 4.8: Single Axis Reconstruction on Image $I_0$




(a) EL2 (gray) Clusters (b) EL4 (blue) Clusters (c) EL5 (red) Clusters

Figure 4.9: Partition of Data into Clusters for Image $I_0$ Using KMeans Clustering

Accordingly, the clustering method of choice will be DBSCAN clustering, where each point of a cluster has a neighborhood of at least a minimum number of points that are defined by the user. DBSCAN requires two parameters: epsilon and minimum number of points. Epsilon is the maximum distance between two points required to be considered neighbors. Minimum number of neighbors (data points) is the required number of points for a cluster to be created.




(a) EL2 (gray) Optimal Eps (b) EL4 (blue) Optimal Eps (c) EL5 (red) Optimal Eps



(d) EL2 (gray) Clusters (e) EL4 (blue) Clusters (f) EL5 (red) Clusters

Figure 4.10: Dual Axis Partition of Data into Clusters for Image $I_0$

The minimum number of points was selected to be 200 based on similar literature. However, to determine the optimal values of epsilon, the Nearest Neighbors Algorithm is used to calculate the average distance between each point and its nearest neighbors (the same value used for minimum number of points). These values are sorted to produce a k-distance elbow plot to determine the ideal epsilon value [19]. Figure 4.10 provides the k-distance elbow plots

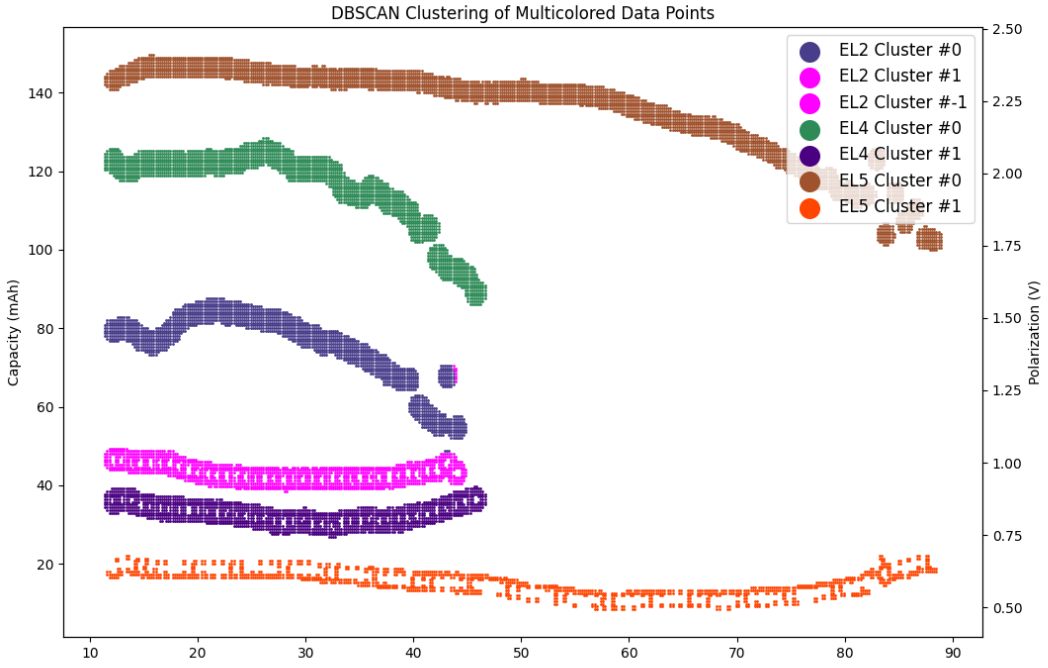



Figure 4.11: Clustering Output for Image $I_0$

for each of the legend items and the corresponding clusters using that epsilon value.

Ultimately, the clustering algorithm should yield two clusters for each legend item. Figure 4.11 provides the final image of the clusters generated through the DBSCAN clustering for Image $I_0$. Clusters labelled with "-1" are the outliers.

### 4.2.8 Assignment of Data to Axes

With the clusters of the data series, the clusters need to be assigned to the left and right y axes. The specific prompt sent to ChatGPT is available in Appendix E.

Using ChatGPT, the clusters in the reconstruction are assignment based on the input image. For each cluster, ChatGPT must assign the cluster to either the left or right axis. In the case that ChatGPT doesn't assign the cluster to the left or right, the default is left.

For example, if ChatGPT was provided the prompt and $I_0$, the following information should be extraction in the expected JSON format outlined in the prompt:

- EL2 Cluster #0: left
- EL2 Cluster #1: right
- EL4 Cluster #0: left
- EL4 Cluster #1: right

- EL5 Cluster #0: left
- EL5 Cluster #0: right

Since all of the data is currently in terms of the left y axis, all the clusters that are assigned to the right axis will be converted to the second axis through the methods described in Section 4.2.6. To decrease the number of points generated, each cluster will be randomly downsampled to 40% of the original size.

When plotting, all the points for the left axis will be denoted with a circle, and all the points for the right axis will be denoted with an inverted triangle. The final reconstruction for Image $I_0$ is shown in Figure 3.3. A subset of the sample CSV file for Image $I_0$ can be found in Appendix D.

## 4.3 Experimentation Process

To test the validity of the reconstruction pipeline, a variety of experiments will be carried out.

During the pipeline construction, smaller experiments outlined in Section 4.2 will be conducted to refine module decisions. For comparison, 28 images from the test set will be randomly sampled to establish a baseline. Specifically, the Image Augmentations, Color Analysis methods, and OCR tools will be evaluated on the sampled dataset.

Upon completing the pipeline, it will be tested on the entire test set. Each plot image's bounding box coordinates, annotations, metadata, single-axis reconstruction, clustering metadata, and dual-axis reconstruction will be saved and manually evaluated for accuracy.

From there, to test the impact on VLM performance, 10 of the plots will be randomly sampled from the test data set. Each plot will be provided to the pipeline to retrieve the CSV data file. Questions (from the VQA dataset), the original images, and the corresponding plot data from the reconstruction will be provided to ChatGPT with a prompt like in Appendix G. To provide a baseline, the question and the original image will be provided to ChatGPT with a prompt like in Appendix F. The ground truth answer will be hand annotated and compared to the results from the baseline and the experimental group.

# Chapter 5

# Evaluation Metrics

## 5.1 Pipeline Development Decisions

The development of the pipeline involves making several decisions based on both qualitative and quantitative evaluations from experiments. Specifically, the Image Augmentation and OCR tools will be chosen based on the quality of the visual results.

Due to the complexity of visually assessing color analysis methods, three established metrics will be used for quantitative assessment: Peak Signal-to-Noise Ratio (PSNR), Mean Squared Error (MSE), and Structural Similarity Index (SSIM). PSNR measures the quality of reconstructed images by comparing the original and reconstructed images, indicating how closely the reconstruction maintains the integrity of the original plot's data points and background. MSE calculates the average squared difference between pixel values of the original and reconstructed images, with lower values indicating minimal deviation in data point location and value representation. SSIM evaluates the perceived structural change in images, ensuring that key features such as axes, data labels, and markers are accurately preserved during reconstruction.

## 5.2 Complete Reconstruction Method

In the domain of scientific plot reconstruction, where the goal is to transform an image of a plot into a reconstructed digital version, accurately evaluating the quality of the reconstruction is crucial. The reconstruction process should ideally result in an output image that closely matches the original in terms of data representation and visual fidelity. The evaluation will focus on individual pipeline components, overall task quality, and complete reconstruction accuracy.

### 5.2.1 Components of the Pipeline

Since total accuracy does not reveal the contributions of each independent component of the pipeline, the accuracy of the individual components needs to be evaluated.

Each of the metadata, ground truth bounding boxes, and true axis ranges will be annotated by hand. For the metadata extraction, the percentage of correct values from GPT4

in comparison to the hand labelled dataset will be evaluated. Bounding box extraction will be evaluated using average precision at 75% Intersection over Union (AP75) to determine the accuracy of predicted bounding boxes overlapping with ground truth bounding box. Lastly, for the axis recalculation, accuracy will be measured using Root Mean Squared Error (RMSE) for each axis independently.

The fine-tuned object detection model for annotation extraction will be evaluated using precision, recall, mean Average Precision at 50% Intersection over Union (mAP50), and mAP50-95 across different IoU thresholds. First, the model's precision, measuring the accuracy of the model's detection, will be calculated. With precision, the model's recall will be determined to measure how many of the actual objects in the images were correctly detected by the model. Afterwards, the mean Average Precision at 50 percent Intersection over Union (mAP50) will be calculated to determine the level of accuracy in terms of how well the model's predicted bounding boxes overlap with the ground truth boxes at this threshold. Similarly, the model's average of the mean Average Precision (mAP50-95) will be calculated at different IoU thresholds, from 50 to 95 percent to evaluate the strictness in bounding box overlap. These metrics will help to assess the annotation extraction process within the pipeline.

Clustering performance of the data series with color analysis and DBSCAN will be evaluated based on the percentage of fully correct clusters. For a cluster to be completely correct, all the data points of a particular data series must be grouped under a single cluster. As this is the strictest calculation of accuracy of the clusters, this will only reward complete accuracy without any subjectivity. For the axis assignment of the clusters, the percentage of correctly assigned clusters to the appropriate axis will be reported to depict the performance of GPT4's assignment of the clusters.

### 5.2.2 Task Accuracy

Performance of various tasks will be evaluated based on average accuracy, with each plot's accuracy hand-annotated and the final reporting as the mean accuracy across the dataset.

For each of the axes, the accuracy will be the average of the correctness of the title, minimum of the range, maximum of the range, and the scale of the axis, as these are the main facets of the axis. The accuracy of the features of the axis will be reported with a Bernoulli random variable, a 0 or 1, to emphasize complete accuracy.

For the legend, the reported accuracy will be the average percentage of fully correct items for the number of legend items, the titles of the legend items, the colors of the legend items, and the pairs of legend items. We decompose the legend extraction task to understand which aspects of the legend are most challenging for the pipeline.

To evaluate the data series, the accuracy of the clustering of data series, axis assignment of data series, and complete correctness of the data series will be reported. Again, the decomposed task splits the clustering and axis assignment tasks to evaluate the performance on subtasks.

### 5.2.3 Overall Accuracy

Overall accuracy will be determined by an annotator, with each reconstruction assigned a Bernoulli variable (0 or 1) to represent correctness, where the percentage of fully correct charts are reported.

## 5.3 Visual Question Answering Performance

This section outlines the evaluation criteria and metrics for assessing the reconstruction pipeline's performance in assisting VLMs in answering questions about scientific plots. Accuracy will be used as the primary metric, calculated by comparing the model-generated answers to ground truth answers, with the percentage of correct answers reported.

# Chapter 6

# Results

This chapter presents the outcomes of our study, detailing the performance and insights gained from our experiments. The results are organized into several sections to provide a comprehensive view of our dataset distribution, pipeline development decisions, individual component evaluations, and overall pipeline accuracy.

## 6.1 Dataset Distribution

### 6.1.1 Scientific Figure Dataset

The scientific figure dataset consists of 500 images split into training, validation, and testing data sets. All of the images are sourced from scientific journals in the battery domain literature. Every image is classified as a scatter or line plot.

In total, there are 260 images in the training dataset, 140 images in the validation dataset, and 100 images in the testing dataset.

The 260 images in the training dataset are transformed using crop up to 20%, rotation between $-10°$ and $+10°$, grayscale on 25% of images, and blur up to 2.5px to create 3 images per training image. In the end, there are 780 training images.

The data distribution of the testing dataset is depicted in Table 6.1. All the evaluations of the pipeline are done using the testing dataset, as the annotation extraction dataset was trained using the training and validation datasets.

### 6.1.2 Visual Question Answering Dataset

To evaluate the ability of our model to answer questions from scientific figures, we constructed a dataset comprising of 10 randomly sampled images from the test set of the Scientific Figure Dataset, specifically focusing on figures from battery-related papers. For each figure, three question-answer pairs were manually generated, focusing on extracting plot information and requiring domain-specific knowledge about batteries. These questions were categorized into five types: Specific Data Points, Minimum/Maximum, Range, Comparison, and Average. The distribution of these questions is summarized in Table 6.2.

Table 6.1: Scientific Figure Test Data Distribution

| Category | Specific Task | Type | Percentage of Images (%) |
|---|---|---|---|
| Axis | Number of Y Axes | One | 87 |
| Axis | Number of Y Axes | Two | 13 |
| Axis | Range | Break | 2 |
| Axis | Range | Not Provided | 7 |
| Axis | Range | Continuous | 91 |
| Axis | Scale | Logarithm | 95 |
| Axis | Scale | Ratio | 1 |
| Axis | Scale | Linear | 4 |
| Legend | Colors | Different | 97 |
| Legend | Colors | Variations of Same Color | 3 |
| Legend | Legend Items | Explicit | 76 |
| Legend | Legend Items | Implicit | 24 |
| Legend | Location | In the Chart Area | 75 |
| Annotations | Arrows | In the Chart Area | 12 |
| Annotations | Free Text | In the Chart Area | 55 |
| Annotations | Small Chart | In the Chart Area | 7 |
| Annotations | Other Annotations | In the Chart Area | 10 |

## 6.2 Pipeline Development Decisions

### 6.2.1 Image Augmentation

To determine the best image augmentation, the reconstruction pipeline was tested using various augmentations. For example, the input image, Figure 6.3a, depicts a relatively faded plot from a battery paper that involves various colors. If we just reconstruct the plot using the original image, we get an incomplete plot, as seen in Figure 6.1b.

To ensure that all the data is reconstructed, we must augment the original image to enhance the contours in the image. Testing a 50% level and 200% level, the original image is augmented using brightness, contrast, and sharpness. By just a quick glance, we can rule out more brightness (Figure 6.1f), less contrast (Figure 6.1d), and less sharpness (Figure 6.1e) because of the inability to reconstruct the basic features in the image. We notice that the more contrast (Figure 6.1g) performs the best out of the image augmentations, as it can recreate all the major plot lines. Figure 6.1 depicts all the reconstruction outputs.

Similar results were seen across the sample of 28 images. Therefore, more contrast will be the choice of image augmentation. Combinations of the transformations were not tested due to the poor quality of the independent transformations.

Table 6.2: Visual Question Answering Dataset Overview

| Question | Number of Questions |
|---|---|
| Specific Data Points | 14 |
| Minimum / Maximum | 5 |
| Range | 4 |
| Comparison | 6 |
| Average | 1 |
| Total | 30 |

### 6.2.2 OCR Tools

In the pipeline, OCR is used to determine the available values on the axis that can be used to perform the axis recalculation. A variety of OCR Tools has been released to extract text from images. To determine the best OCR Tool to detect the numerical data, a sample of 28 images from the test set were used to decide the best OCR Tool for the pipeline.

Figure 6.2 showcases the differences in extraction between Keras, Pytesseract, and Amazon Textract on a sample input image. Similar results were seen across the sample of 28 images.

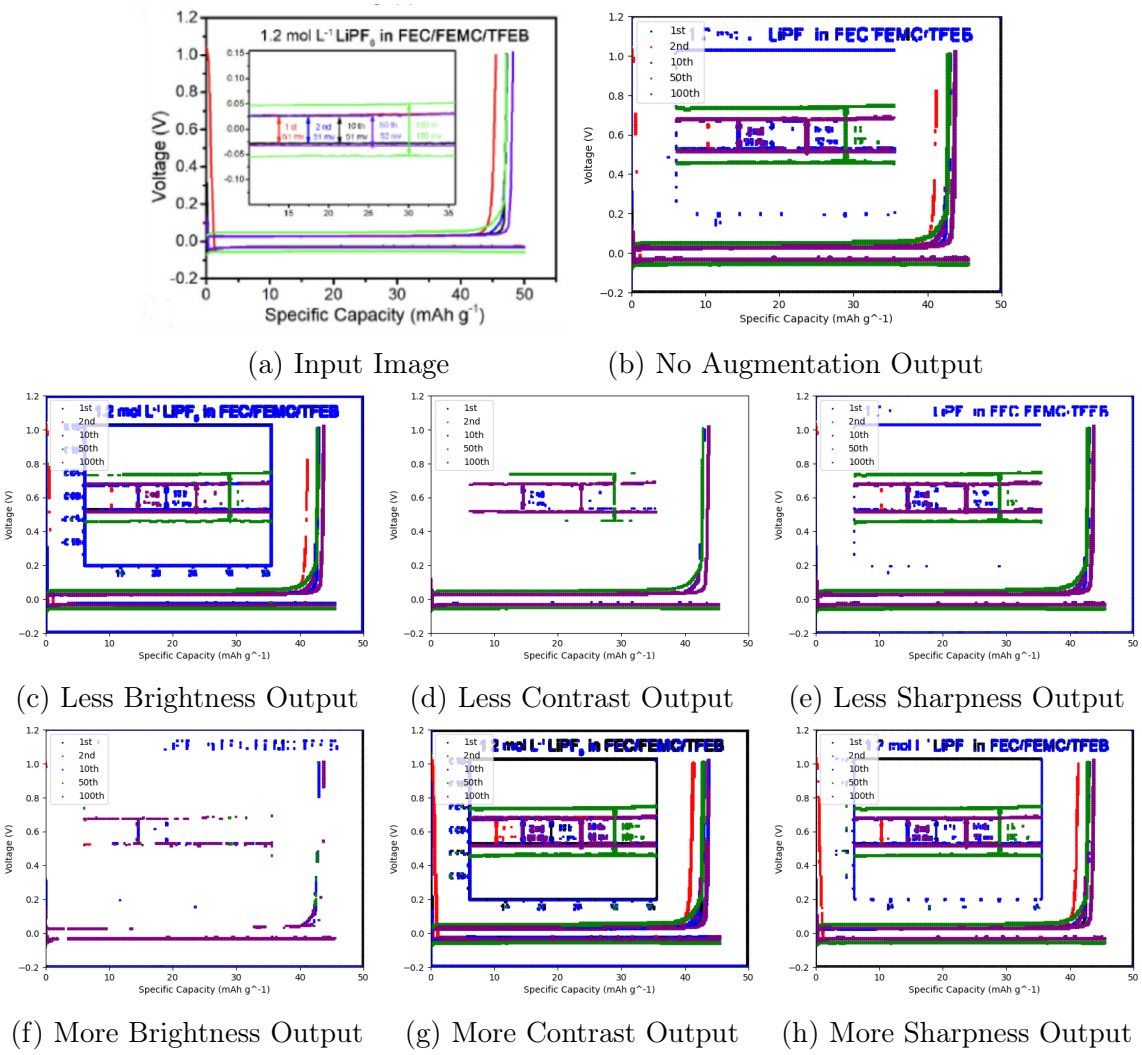
Keras was able to extract a great amount of text; however, the OCR results were really poor on numerical values with commas, decimals, and negative signs. On the other hand, Pytesseract allowed for the inclusion and exclusion of certain characters, but the tool did not have many extractions, making recalculation really challenging. Thus, Amazon Textract was the OCR Tool of choice, as it was able to extract most numerical values accurately.

### 6.2.3 Color Analysis Method

To determine the best color analysis method, a sample of 28 images from the test set were tested with each of the color clustering methods: linear search with euclidean distance, linear search with cosine similarity, and kdtree search. As stated in the methods section, since the run time for linear search is much larger, the centroids for each of the simple colors must be defined. The colors and the RGB values for their centroids are listed in Appendix C. Also, the mapping of the webcolors to simple colors can be found in Appendix B.

In Figure 6.3, a sample input image and the reconstructions using various color analysis tools is provided. While there is still noise present at the top of the reconstructed image, the KDtree search method outperforms the other two by reconstructing the pink plot line, as seen in Figure 6.3d. Since the centroids are empirically found, in most of the input images, the plot lines colored with a color other than blue, green, or red were usually not found. Therefore, either the centroids are not be representative of the entire color or the centroid clustering method does not work in this three-dimensional space due to the irregularity of the colors.

To aggregate the results, Table 6.3 provides the PSNR, MSE, and SSIM scores between the input image and the reconstructed image for the various color analysis methods. While the euclidean performs marginally better than the KDtree Search, the reconstructions are




(a) Input Image (b) No Augmentation Output

(c) Less Brightness Output (d) Less Contrast Output (e) Less Sharpness Output

(f) More Brightness Output (g) More Contrast Output (h) More Sharpness Output

Figure 6.1: Image Reconstructions with Input Image Augmentations

more accurate in the KDtree Search because of the color scheme. Therefore, the KDtree Search will be the method used for reconstructing the images.

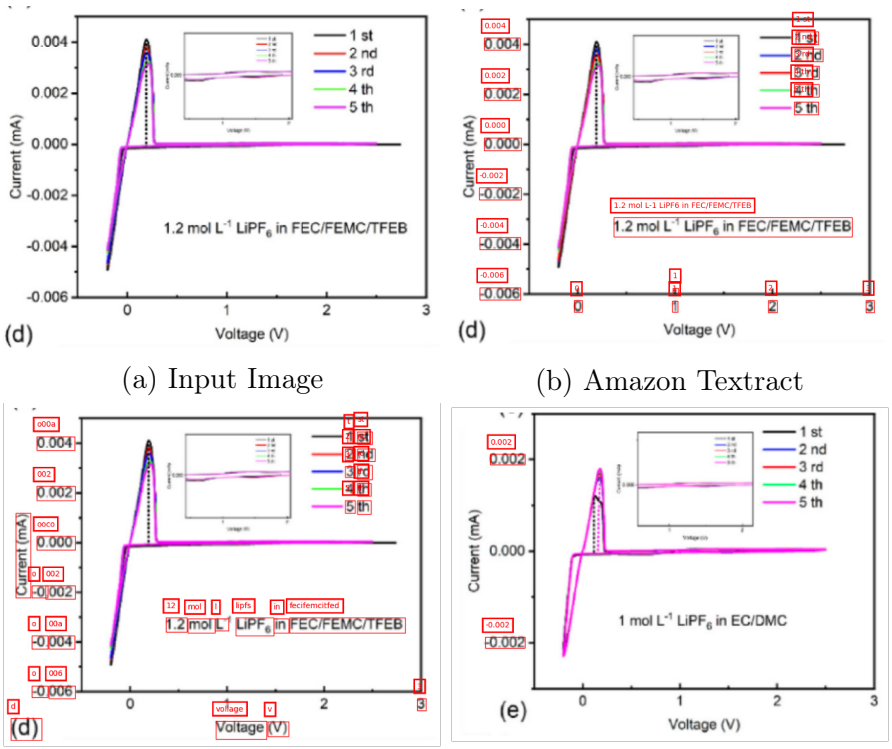
## 6.3 Complete Reconstruction Pipeline

### 6.3.1 Components of the Pipeline

The accuracy of the individual components of the pipeline was evaluated on the testing dataset.

#### Metadata Extraction

Metadata extraction is completed using ChatGPT's vision model. Unfortunately, the model is just a black box for customers, so the evaluation is primarily completed using the empirical




(a) Input Image

(b) Amazon Textract

(c) Keras

(d) Pytesseract

Figure 6.2: Output of Various OCR Tools on a Sample Image

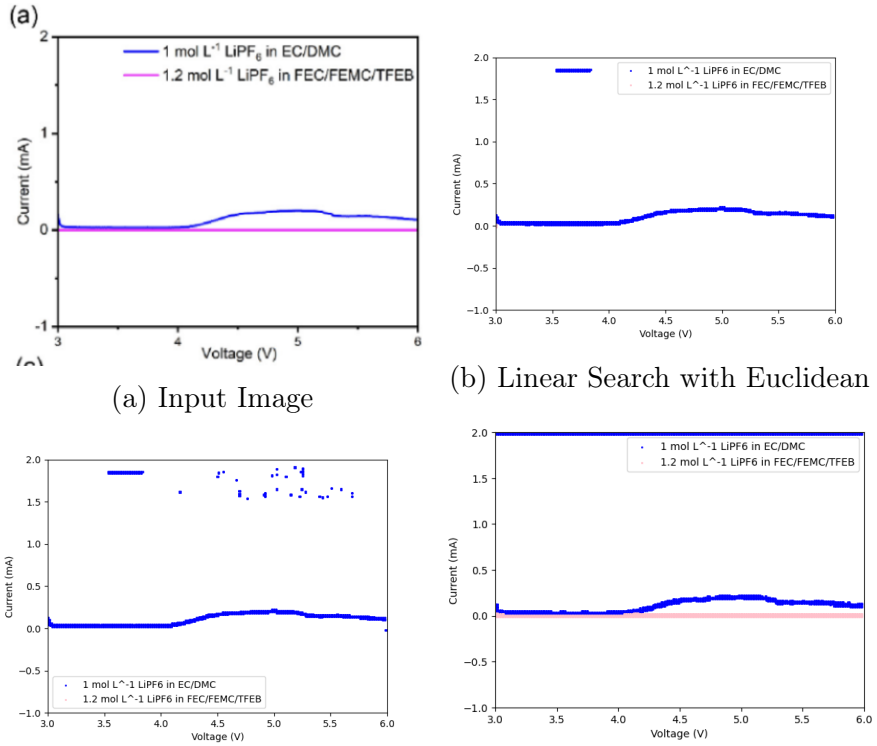
Table 6.3: Color Decision - Results

| Color Clustering Method | PSNR | MSE | SSIM |
|---|---|---|---|
| Euclidean Distance | 34.884 | 11298.300 | 0.565 |
| Cosine Similarity | 35.097 | 25825.421 | 0.546 |
| KDtree Search | 34.678 | 12677.668 | 0.550 |

results for the task.

On the testing dataset, the prompt and input image was sent to GPT4. To achieve accuracy, the model had to output the exact information written in the plot. For example, in Figure 3.2, we would expect GPT4 to output a range of 0 to 100 for the x-axis because 0 is the minimum value of the tick marks and 100 is the maximum values of the tick marks even though the actual range is approximately [-10, 100]. This is to maintain accuracy across the inferences such that GPT4 was only using the information provided. As such, the axis recalculation will adjust the values of the range later to get the true range.

All the results are presented in Table 6.4. From the results, GPT4 performs well with the exception of the Legend Colors and Legend Pairs. Specifically, the extraction of the colors is a challenging task for GPT4. Adding in the task of matching colors to text, the vision model struggles to accurately pair the two together, leading to an accuracy of only 0.791.




(a) Input Image

(b) Linear Search with Euclidean

(c) Linear Search with Cosine

(d) KDtree Search

Figure 6.3: Image Reconstructions with Various Color Analysis Tools

### Bounding Box Extraction

The evaluation of the bounding box extraction method using the average precision at a 75% IoU threshold yielded notable results. Detections with IoU values equal to or greater than this threshold were considered true positives, while those below were classified as false positives.

The analysis revealed a precision rate of 95%, indicating that 95% of the predicted bounding boxes that met or exceeded the IoU threshold were accurate when compared to the ground truth annotations. Furthermore, the recall rate was found to be 100%, suggesting that all ground truth bounding boxes were correctly identified by the model at the specified IoU threshold. This perfect recall score highlights the effectiveness of the model in detecting all relevant objects within the test dataset without missing any true positives. The overall accuracy of the bounding box predictions, measured as the mean IoU across all predictions, was approximately 96.05%. This accuracy metric reflects the average extent to which the predicted bounding boxes overlapped with the ground truth across all samples.

Overall, the bounding box extraction method has high precision, recall, and accuracy at identifying chart areas in images.

Table 6.4: Error Analysis of Metadata Extraction from GPT4

| Category | Sub Category | Sub Category Accuracy | Overall Accuracy |
|---|---|---|---|
| X Axis | | | 0.882 |
| | Title | 0.9 | |
| | Minimum Range | 0.888 | |
| | Maximum Range | 0.857 | |
| Y Axis | | | 0.865 |
| | Title | 0.9 | |
| | Minimum Range | 0.832 | |
| | Maximum Range | 0.863 | |
| Second Y Axis | | | 0.917 |
| | Title | 1 | |
| | Minimum Range | 0.75 | |
| | Maximum Range | 1 | |
| Legend | | | 0.848 |
| | Number of Legend Titles | 0.882 | |
| | Legend Titles | 0.89 | |
| | Legend Colors | 0.831 | |
| | Legend Pairs | 0.791 | |

**Axis Recalculation**

The axis recalculation process was rigorously implemented through three critical steps: (1) employing Optical Character Recognition (OCR) to extract text from images, (2) selecting two specific interval marks on each axis, and (3) recalculating the axis scales using the extracted data.

The accuracy of the axis recalculation pipeline is provided in Table 6.5. The perfect OCR accuracy (1.00) observed on the Second Y Axis, which highlights the robustness of the text extraction component in ideal conditions. However, the consistent drop in accuracy during the 'Choose two correct values' and 'Correct range calculation' steps across all axes suggests a potential area for improvement, particularly in the algorithms that select the interval marks. The overall accuracy also reflects this trend, with the Second Y Axis achieving the highest overall performance at 0.897, suggesting better isolation or clearer marking in these types of graphs.

Table 6.6 presents an error analysis. "Other Axis Values are Interfering" refers to the values incorrectly detected in an axis range that are actually present in another axis. For instance, if the recovery is for the x-axis, there might be nearby y-axis values that are mistakenly recovered to be part of the x-axis range. "No Range" refers to plots without a defined range. Currently, the system will assign and display a default minimum and maximum range which is not correct in this situation.

The table helps identify the most frequent challenges encountered during the recalculation process. Particularly concerning is the high prevalence of 'Bounding Box is Too Large' and 'Other Axis Values are Interfering' errors on the X Axis, each occurring in 32.2% and 25.8% of

Table 6.5: Accuracy of Axis Recalculation Pipeline

| Axis | Pipeline Part | Pipeline Accuracy | Overall Accuracy |
|---|---|---|---|
| X Axis | | | 0.76 |
| | OCR has at least two values | 0.86 | |
| | Choose two correct values | 0.73 | |
| | Correct range calculation | 0.69 | |
| Y Axis | | | 0.83 |
| | OCR has at least two values | 0.86 | |
| | Choose two correct values | 0.82 | |
| | Correct range calculation | 0.82 | |
| Second Y Axis | | | 0.897 |
| | OCR has at least two values | 1 | |
| | Choose two correct values | 0.85 | |
| | Correct range calculation | 0.85 | |

cases. This indicates significant issues in spatial recognition and differentiation, particularly when dealing with complex graph layouts or closely positioned axes. In contrast, errors like 'No Range' and 'Scientific Notation' show varied prevalence, indicating that these errors are highly context-dependent and could be linked to the specific characteristics of the data or the graph's design.

Overall, the results underscore the complexity and variability inherent in the task of automated axis recalculation. While the OCR component shows high reliability, the subsequent steps reveal sensitivity to graph layout and markings. Continuous improvement in these areas is essential for enhancing the robustness and accuracy of the axis recalculation pipeline.

### Annotation Extraction

After successfully fine-tuning for 125 epochs (Figure 6.4), YOLOv5 was fine-tuned to detect the annotations in an image of a plot. The model's precision was 0.812 on the test set, indicating a decent accuracy of the model's detection. The model's recall was 0.715, demonstrating that a great percentage of the annotations in the images were correctly detected by the model. The model's mAP50 was 0.803, indicating a high level of accuracy in terms of how well the model's predicted bounding boxes overlap with the ground truth boxes at this threshold.

A common visualization utilized for object detection models is the confusion matrix. In Figure 6.5, the matrix indicates that the model accurately predicts the annotations 71% of the time. While this percentage is not perfect, the performance is quite good for a broad task. Figure 6.6 illustrates the performance of the annotation detector on a few images from the validation set; the high confidence on the bounding boxes indicates that the model is well suited for the task at hand.

Table 6.6: Error Analysis of Axis Recalculation Pipeline

| Axis | Reason for Error | Prevalence of Error |
|---|---|---|
| X Axis | | |
| | Bounding Box is Too Large | 0.322 |
| | Other Axis Values are Interfering | 0.258 |
| | X Range is Extracted Together | 0.129 |
| | No Range | 0.129 |
| | Nonlinear Scale | 0.097 |
| | Wrong Text is Extracted | 0.032 |
| | Scientific Notation | 0.032 |
| Y Axis | | |
| | No Range | 0.4 |
| | Bounding Box is Too Large | 0.35 |
| | Break in Axis | 0.1 |
| | Scientific Notation | 0.1 |
| | Wrong Detection | 0.05 |
| Second Y Axis | | |
| | Other Axis Values are Interfering | 0.5 |
| | Bounding Box is Too Large | 0.5 |

#### Clustering of the Data Series with Color Analysis and DBSCAN

Utilizing the test data set, the pipeline achieved a clustering accuracy of 77.2%. This metric indicates the effectiveness of our clustering algorithm in grouping similar data points correctly. The percentage reflects the proportion of data series that were 100% correct. Specifically, for the plots with a second axis, DBSCAN performed with 67.3% accuracy.

#### Assignment of the Clusters

For the assignment of clusters, the method achieved an accuracy of 39.74%. This performance measure was obtained by comparing the manually annotated expected results with the actual assignments produced by the pipeline. Unfortunately, GPT4 really struggled with the assignment of clusters. This may be due to the colors of the clusters and the stark difference in color from the original image.

### 6.3.2 Task Accuracy

The task accuracy for various components of our graphical reconstruction pipeline was rigorously evaluated using a comprehensive test set. The performance results are detailed in Table 6.7, and a visual demonstration of these outcomes is provided through annotated images in Figure 6.7.

High task accuracy was observed in the retrieval of axis and legend information, with all categories achieving greater than 85% accuracy. This high level of performance demonstrates

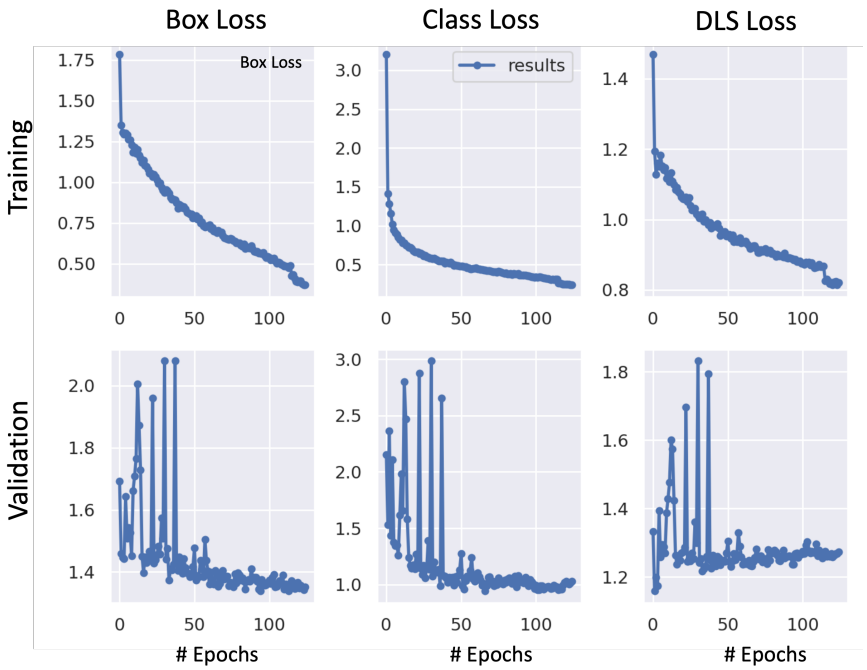



Figure 6.4: Training Results for the Annotation Detection Model

the robustness of Amazon Textract and the axis recalculation algorithm. However, it is important to note the limitations of the current pipeline, which is optimized for numerical and linear ranges. Accuracy decreases for graphs with non-present or exponential ranges. An example of a common error is shown in Figure 6.7, where the '0' in the bottom right-hand corner of the image interfered with the correct range retrieval, causing the x-axis range to default to [0, 100].

In contrast, the extraction and clustering of specific data points showed variable performance. The clustering accuracy stands at 0.772, indicating a high level of precision in data grouping. However, the overall accuracy for complete data series, integrating axes, legends, and data points, was lower at 0.562. This decrease can be attributed to the complexity of coordinating multiple elements, especially in dual-axis configurations, which only achieved a 0.423 accuracy rate.

The overall percentage of charts reconstructed correctly from the test dataset stands at only 34%, underscoring the challenges inherent in automated graphical data reconstruction and highlighting areas for potential improvement. While the accuracy of the complete pipeline is quite low, the current state of the art, WebPlotDigitizer [15], is a semi-automated solution that would not be able to do this task without a human. Therefore, the accuracy of 34% is an improvement from the existing systems for plot extraction.

Figure 6.8 provides a few successful examples of input images and their reconstructions. On the other hand, Figure 6.9 provides a few examples of failed reconstructions with the corresponding input images. Most of these are due to failed axis assignments of the clusters or lack of annotation extraction.

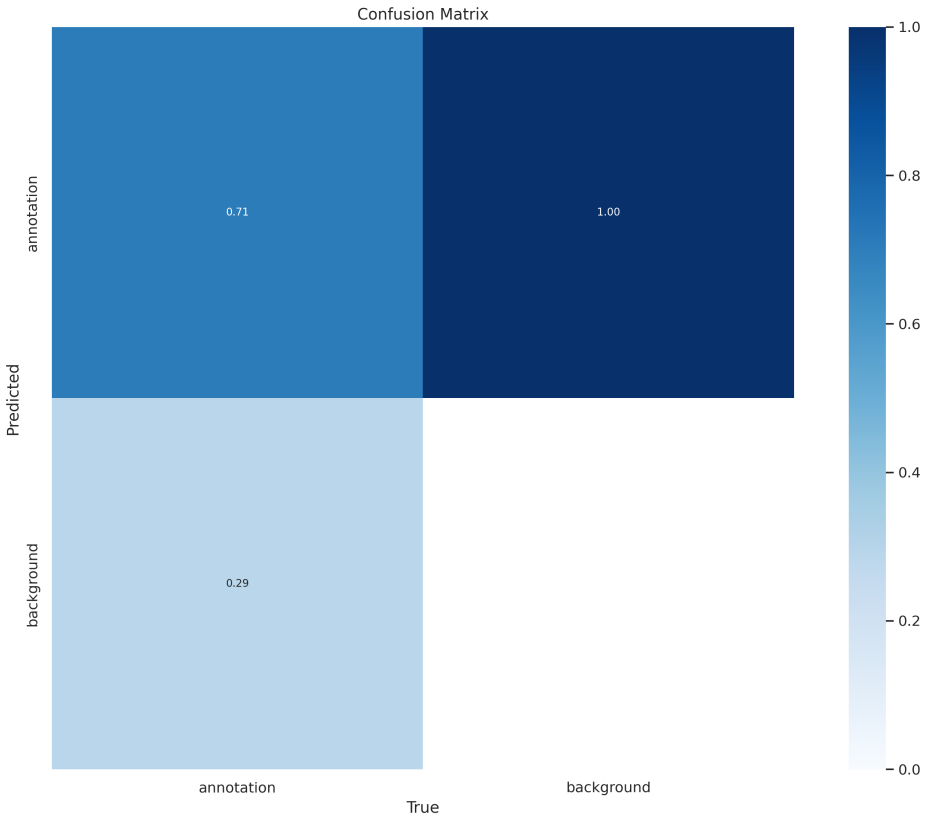


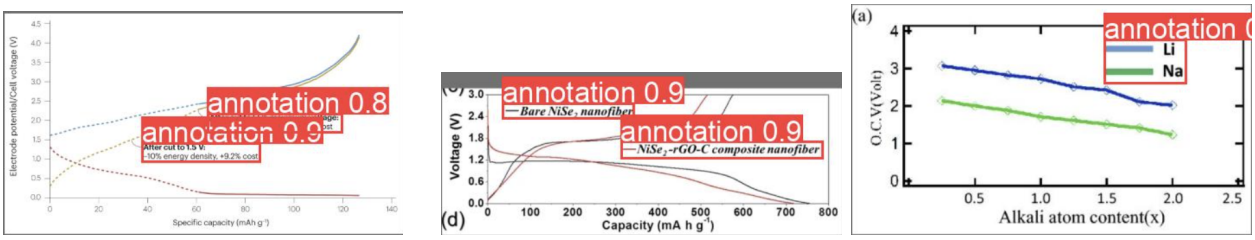

Figure 6.5: Confusion Matrix for the Legend Detection Model

## 6.4 Visual Question Answering Performance

We assessed the performance of our vision-language model (VLM) using a standard metric of accuracy, which measures the proportion of correct answers provided by the model against the ground truth. The experiment was conducted under two conditions: without the use of Comma-Separated Values (CSV) files and with the integration of CSV files containing exact data from the plots. This aimed to analyze the improvement in model performance when augmented with direct data access.

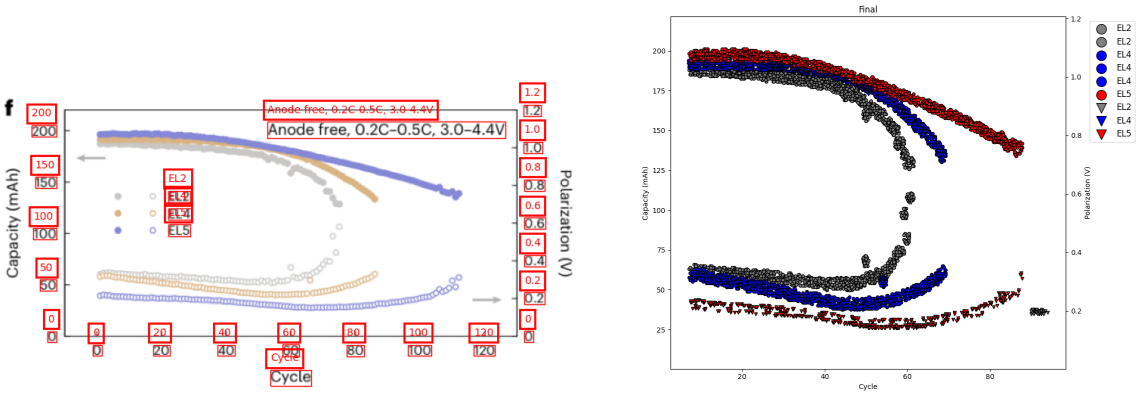
The results, presented in Table 6.8, show significant improvement across all question types when the model is augmented with CSV data. Specifically, the accuracy for 'Specific Data Points' increased from 45.2% to 85.1%, and for 'Minimum/Maximum' from 33.3% to 93.3%. Notably, the most challenging category without CSV, 'Range', saw an increase from 21.9% to 65.6%. The categories 'Comparison' and 'Average', which benefit from direct numerical comparisons, reached perfect accuracy with CSV augmentation.

Overall, the total accuracy improved from 44.6% to 87.4%, demonstrating the effectiveness of incorporating structured data into the VLM's processing pipeline for answering visual questions about scientific plots. This is really interesting because even though the extractions were not perfect, the VLM is able to utilize some of the data to get to a fine grained answer, resulting in better performance overall.




(a) Image 1 (b) Image 2 (c) Image 3

Figure 6.6: Annotation Detection on a Few Images of the Validation Set




(a) Amazon Textract Results on Sample Image (b) Reconstruction of Sample Image

Figure 6.7: Illustration of Axis Recalculation Error on a Sample Image

## 6.5 Limitations

This research developed a comprehensive approach to reconstructing scientific plots for assisting visual language models (VLMs) in interpreting and responding to queries about visual data. Despite its innovations, the study encountered several limitations. The pipeline demonstrated specific challenges in generalizing to different types of plots and scientific domains beyond battery research, indicating a need for additional features for broader plots and figures. Complex chart structures, such as those with dual axes or non-standard representations, also posed significant challenges, impacting the accuracy of data extraction and interpretation. Additionally, the annotation detection process, while advanced, did not achieve perfect precision and recall, leading to potential inaccuracies in the final data used by VLMs. The color analysis component, reliant on predefined color sets, sometimes struggled to accurately identify and match colors, particularly when faced with subtle variations not represented in the training set. Currently, the system cannot handle plots with variations of the same color marker and plots with nonlinear scales.

The limitations observed primarily relate to handling non-linear and non-numerical data ranges, as well as integrating multiple data components (axes, legends, and data points) in complex chart layouts. The performance disparity between single and dual-axis configurations further indicates difficulties in accurate axis assignment and the synchronization of multiple elements within the same chart. Future improvements should focus on enhancing

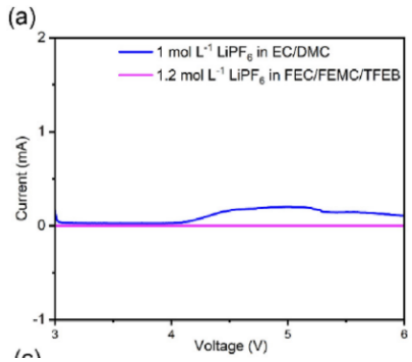
Table 6.7: Reconstruction Pipeline Task Accuracy

| Category | Sub Category | Sub Category Accuracy | Overall Accuracy |
|---|---|---|---|
| X Axis | | | 0.86 |
| | Title | 0.91 | |
| | Minimum Range | 0.82 | |
| | Maximum Range | 0.84 | |
| | Scale | 0.87 | |
| Y Axis | | | 0.895 |
| | Title | 0.92 | |
| | Minimum Range | 0.90 | |
| | Maximum Range | 0.89 | |
| | Scale | 0.87 | |
| Second Y Axis | | | 0.9375 |
| | Title | 1 | |
| | Minimum Range | 0.917 | |
| | Maximum Range | 0.917 | |
| | Scale | 0.917 | |
| Legend | | | 0.871 |
| | Number of Legend Titles | 0.87 | |
| | Legend Titles | 0.887 | |
| | Legend Colors | 0.912 | |
| | Legend Pairs | 0.815 | |
| Data Series | | | 0.639 |
| | Clustering of Data Series | 0.772 | |
| | Axis Assignment | 0.583 | |
| | Single Axis Data Series | 0.587 | |
| | Dual Axis Data Series | 0.423 | |
| | Data Series Correctness | 0.562 | |
| Total Accuracy | | | 0.34 |

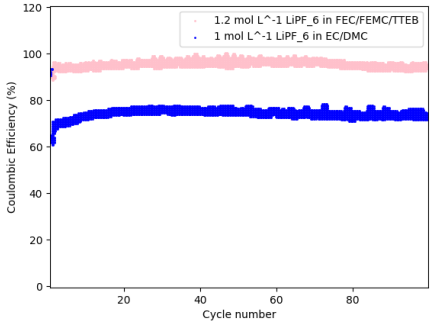
the algorithm's ability to assign datasets to axes and improve the integration process of different chart components to boost overall accuracy. Developing more sophisticated techniques for handling special cases, such as non-linear scales and overlapping data elements, will be crucial in advancing the capabilities of our reconstruction pipeline.

Table 6.8: Comparison of Results Without and With CSV

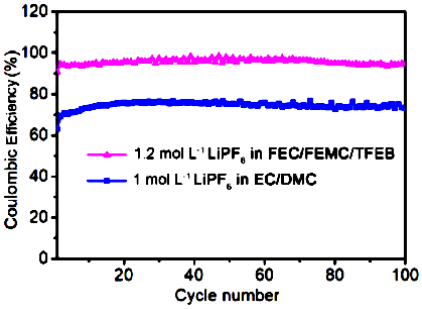
| Question | Without CSV | With CSV |
|---|---|---|
| Specific Data Points | 0.452 | 0.851 |
| Minimum / Maximum | 0.333 | 0.933 |
| Range | 0.219 | 0.656 |
| Comparison | 0.667 | 1.000 |
| Average | 0.500 | 1.000 |
| Total | 0.446 | 0.874 |



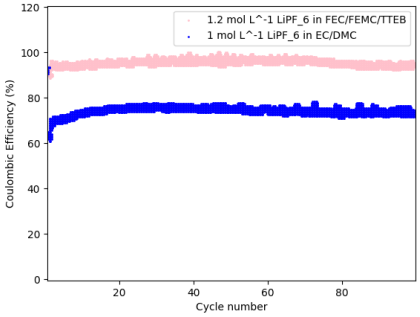

(a) Original Image #1



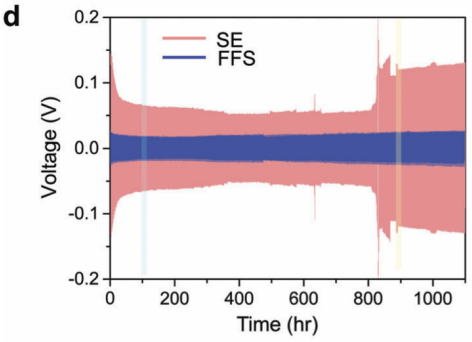

(b) Reconstruction of Image #1



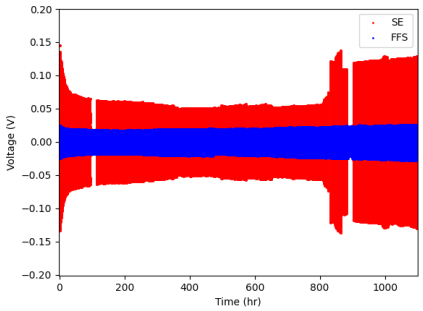

(c) Original Image #2



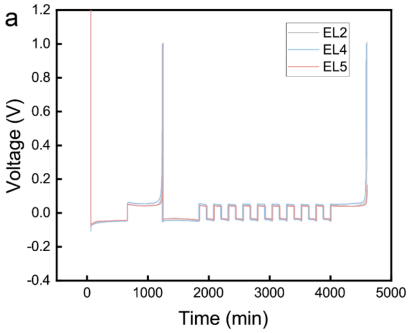

(d) Reconstruction of Image #2



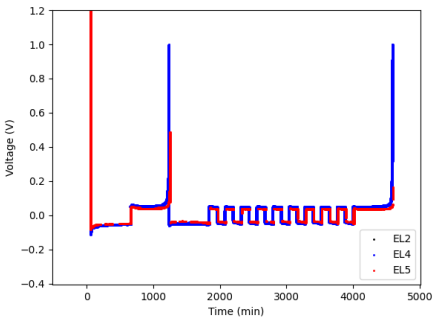

(e) Original Image #3




(f) Reconstruction of Image #3

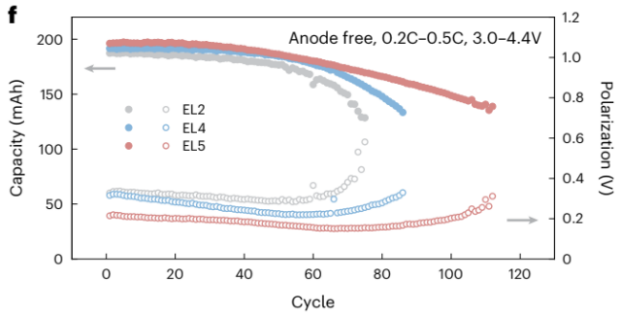



(g) Original Image #4

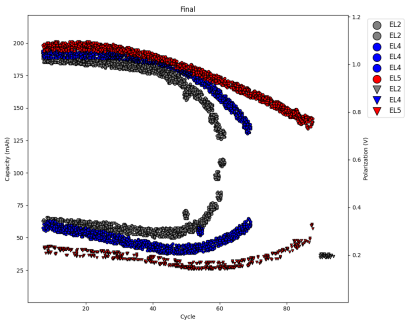



(h) Reconstruction of Image #4

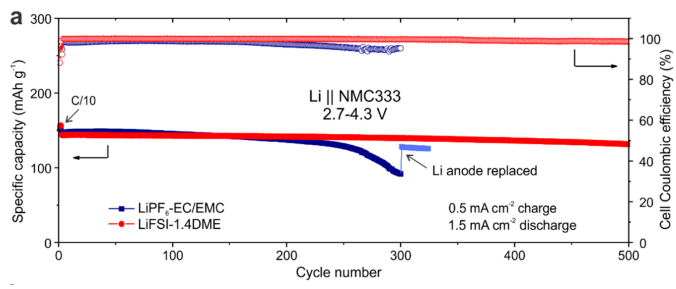
Figure 6.8: Successful Reconstructions of the Input Images



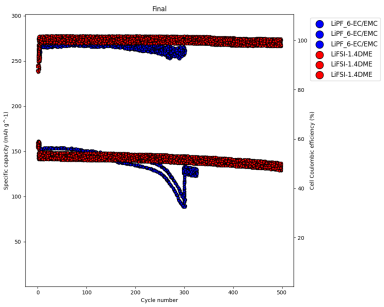

(a) Original Image #1



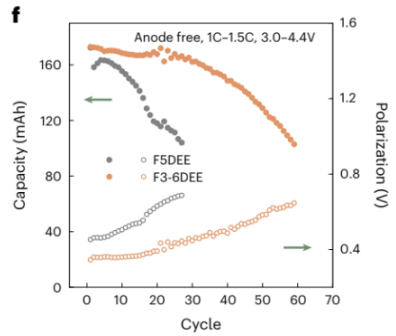
(b) Reconstruction of Image #1



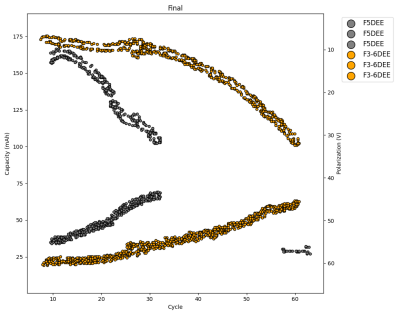

(c) Original Image #2



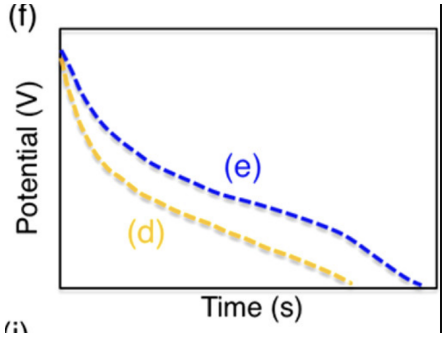
(d) Reconstruction of Image #2



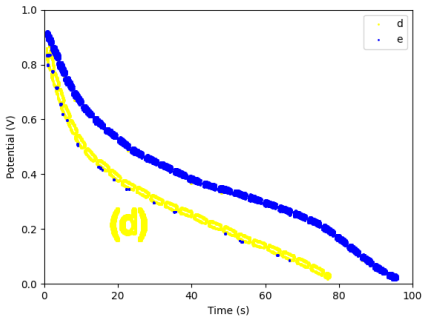

(e) Original Image #3



(f) Reconstruction of Image #3




(g) Original Image #4




(h) Reconstruction of Image #4

Figure 6.9: Failed Reconstructions of the Input Images

# Chapter 7

# Conclusion

In conclusion, this thesis has introduced a groundbreaking approach to enhance the capabilities of visual language models (VLMs) by developing a pipeline for the reconstruction of scientific plots. Focused primarily on scatter and line plots within the battery research domain, the pipeline leverages advanced data processing techniques to transform complex visual information into structured, machine-readable formats that are more easily interpreted by VLMs.

The core components of the pipeline include optical character recognition (OCR) for text extraction, color analysis to differentiate plot elements, and machine learning models for precise annotation detection. These modules work to convert images of scientific plots into detailed CSV files. Preliminary testing has shown that this structured approach not only improves the accuracy of the responses provided by VLMs but also enhances their ability to engage with complex scientific inquiries based on visual data.

Key results from the deployment of this pipeline have demonstrated its efficacy, particularly in extracting accurate metadata and plot data from a variety of scatter and line plots. The OCR component exhibited robust performance in text detection, achieving high accuracy rates in extracting axis values. Color analysis, while facing challenges with subtle color variations, successfully identified different plot elements in the majority of test cases. Annotation detection also showed promising results, with the fine-tuned YOLOv5 model achieving substantial precision and recall rates, though it also highlighted areas for potential improvement.

However, the pipeline's performance also illuminated several limitations. Challenges were particularly noted in handling plots with complex structures or dual axes, where the standard processing techniques struggled to maintain accuracy. Additionally, its current lack of generalization to other types of plots is a limitation for greater use.

Further testing and validation across diverse datasets will be crucial for optimizing the pipeline's capabilities. This will involve enhancing the color processing techniques to handle a wider spectrum of markers found in scientific plots and refining the annotation detection algorithms to improve their accuracy and reliability. Future work will focus on expanding the pipeline's applicability by incorporating a broader range of plot types.

# Acknowledgments

First and foremost, I would like to express my deepest gratitude to Dr. Michael Cafarella, my principal investigator, for his invaluable advice and guidance throughout this journey. His expertise and support have been instrumental in shaping this thesis.

I am deeply thankful to Chunwei Liu, a postdoctoral fellow in my lab, for his support on the pipeline development and project direction. His willingness to spend time debugging and helping me overcome hurdles has been greatly appreciated.

I would also like to extend my thanks to Oscar Moll Thomas, another postdoctoral fellow in my lab, for his assistance in testing methods with me specifically helping annotate the scientific figure dataset on Roboflow.

My gratitude goes to the Semantic Management group in the Data System organization in MIT CSAIL for their unwavering support and constructive feedback.

I am profoundly grateful to my advisor, Dr. Justin Solomon, for his unwavering support and advocacy throughout my MIT undergrad and MEng journey.

I am immensely grateful to the Boston University Yoga Teacher Training for providing me with an outlet outside of my classes and research throughout the year, offering a much-needed escape and balance in my life.

Thank you to my roommate, Natasha Maniar, for the countless late-night "I need your advice" talks and for being a sounding board for my ideas. Thanks to my other roommate, Angel Yang, for having me take much-needed breaks with your jokes and stories.

A heartfelt thanks to Zoe Pasetsky, Sara Modiano, Opalina Vetrichelvan, Madeline Hon, and all my other friends who have helped me grow in so many ways and spent the last four years with me at MIT.

A special thank you to Alex Zappi for keeping me motivated, proof reading my thesis, and providing company during the long days of coding, writing, and much more.

Most of all, I want to thank my grandparents, my parents, my brother Krishang, my cousin Pearl, my aunt and uncle, and my extended family, friends, and mentors here and in India. I would like to especially acknowledge my parents for their unwavering support, encouragement, and love over the years. I would not be where I am today without them. Also, thanks Krish for keeping a smile on my face through all your stories, making sure that the calls are never dull, and helping me check my math for my plot reconstructions. Thank you for always being there for me and for inspiring me to reach for the stars.

# Appendix A

# Metadata Prompt to GPT

Based on the plot:

- Describe the title of the x-axis.
- Describe the range of the x-axis.
- Describe the title of the first y-axis.
- Describe the range of the first y-axis.
- Describe the title of the second y-axis if there is one.
- Describe the range of the second y-axis if there is one.
- Describe the different types.

Please provide the answers to the aforementioned questions only in the following string format:

```
'{
    \"x-axis\":
    {
        \"title\": [INSERT TITLE HERE IN STRING FORMAT],
        \"range\": [ INSERT RANGE HERE IN [start, end] FORMAT ],
    },
    \"y-axis\":
    {
        \"title\": [INSERT TITLE HERE IN STRING FORMAT],
        \"range\":  INSERT RANGE HERE IN [start, end] FORMAT,
    },
    \"second-y-axis\":
    {
        \"title\": [INSERT TITLE HERE IN STRING FORMAT ELSE null],
        \"range\":  INSERT RANGE HERE IN [start, end] FORMAT ELSE
           null,
    },
```

```
    \"types\": [INSERT THE TYPES IN LIST OF LISTS OF LENGTH = 2 WHERE
        EACH LIST LOOKS LIKE
        [TYPE NAME, MARKER COLOR in ONLY THE FOLLOWING CHOICES \"
           black\", \"white\", \"red\",\"purple\",\"green\", \"yellow
           \", \"blue\", \"pink\", \"orange\", \"grey\"]
    ]
}'
```

# Appendix B

# Mapping of Webcolors to Simple Colors

```
complex_to_simple_color = {
    'aliceblue': 'white',
    'antiquewhite': 'white',
    'cyan': 'blue',
    'aquamarine': 'green',
    'azure': 'white',
    'beige': 'white',
    'bisque': 'white',
    'black': 'black',
    'blanchedalmond': 'white',
    'blue': 'blue',
    'blueviolet': 'purple',
    'brown': 'red',
    'burlywood': 'orange',
    'cadetblue': 'blue',
    'chartreuse': 'green',
    'chocolate': 'orange',
    'coral': 'orange',
    'cornflowerblue': 'blue',
    'cornsilk': 'white',
    'crimson': 'red',
    'darkblue': 'blue',
    'darkcyan': 'blue',
    'darkgoldenrod': 'yellow',
    'darkgray': 'grey',
    'darkgreen': 'green',
    'darkkhaki': 'yellow',
    'darkmagenta': 'purple',
    'darkolivegreen': 'green',
    'darkorange': 'orange',
    'darkorchid': 'purple',
    'darkred': 'red',
    'darksalmon': 'orange',
    'darkseagreen': 'green',
```

```
    'darkslateblue': 'purple',
    'darkslategray': 'blue',
    'darkturquoise': 'blue',
    'darkviolet': 'purple',
    'deeppink': 'pink',
    'deepskyblue': 'blue',
    'dimgray': 'grey',
    'dodgerblue': 'blue',
    'firebrick': 'red',
    'floralwhite': 'white',
    'forestgreen': 'green',
    'magenta': 'pink',
    'gainsboro': 'white',
    'ghostwhite': 'white',
    'gold': 'yellow',
    'goldenrod': 'yellow',
    'gray': 'grey',
    'green': 'green',
    'greenyellow': 'green',
    'honeydew': 'white',
    'hotpink': 'pink',
    'indianred': 'pink',
    'indigo': 'purple',
    'ivory': 'white',
    'khaki': 'yellow',
    'lavender': 'white',
    'lavenderblush': 'white',
    'lawngreen': 'green',
    'lemonchiffon': 'white',
    'lightblue': 'blue',
    'lightcoral': 'pink',
    'lightcyan': 'white',
    'lightgoldenrodyellow': 'white',
    'lightgray': 'grey',
    'lightgreen': 'green',
    'lightpink': 'pink',
    'lightsalmon': 'orange',
    'lightseagreen': 'blue',
    'lightskyblue': 'blue',
    'lightslategray': 'grey',
    'lightsteelblue': 'blue',
    'lightyellow': 'white',
    'lime': 'green',
    'limegreen': 'green',
    'linen': 'white',
    'maroon': 'red',
    'mediumaquamarine': 'green',
```

```
    'mediumblue': 'blue',
    'mediumorchid': 'purple',
    'mediumpurple': 'purple',
    'mediumseagreen': 'green',
    'mediumslateblue': 'purple',
    'mediumspringgreen': 'green',
    'mediumturquoise': 'blue',
    'mediumvioletred': 'pink',
    'midnightblue': 'blue',
    'mintcream': 'white',
    'mistyrose': 'white',
    'moccasin': 'yellow',
    'navajowhite': 'yellow',
    'navy': 'blue',
    'oldlace': 'white',
    'olive': 'green',
    'olivedrab': 'green',
    'orange': 'orange',
    'orangered': 'red',
    'orchid': 'purple',
    'palegoldenrod': 'yellow',
    'palegreen': 'green',
    'paleturquoise': 'blue',
    'palevioletred': 'pink',
    'papayawhip': 'white',
    'peachpuff':'orange',
    'peru': 'orange',
    'pink': 'pink',
    'plum': 'purple',
    'powderblue': 'blue',
    'purple': 'purple',
    'red': 'red',
    'rosybrown': 'pink',
    'royalblue': 'blue',
    'saddlebrown': 'orange',
    'salmon': 'pink',
    'sandybrown': 'orange',
    'seagreen': 'green',
    'seashell': 'white',
    'sienna': 'orange',
    'silver': 'grey',
    'skyblue': 'blue',
    'slateblue': 'purple',
    'slategray': 'gray',
    'snow': 'white',
    'springgreen': 'green',
    'steelblue': 'blue',
```

```
    'tan': 'orange',
    'teal': 'blue',
    'thistle': 'pink',
    'tomato': 'orange',
    'turquoise': 'blue',
    'violet': 'pink',
    'wheat': 'yellow',
    'white': 'white',
    'whitesmoke': 'white',
    'yellow': 'yellow',
    'yellowgreen': 'green'
}
```

# Appendix C

# Simple Color Centroids

| Color | Red | Blue | Green |
|---|---|---|---|
| Black | 0 | 0 | 0 |
| Grey | 128 | 128 | 128 |
| White | 255 | 255 | 255 |
| Red | 255 | 0 | 0 |
| Purple | 255 | 0 | 255 |
| Green | 0 | 255 | 0 |
| Yellow | 255 | 255 | 0 |
| Blue | 0 | 0 | 255 |
| Pink | 255 | 20 | 147 |
| Orange | 255 | 165 | 0 |

# Appendix D

# Sample CSV Format for ChatGPT

```
Cycle,Capacity (mAh),Polarization (V),Type
23.727161389997743,81.43332152995133,,EL2
34.68794142632996,74.6115764114762,,EL2
26.858812828949805,80.0689725062563,,EL2
42.1256135938411,52.78199203235573,,EL2
37.232408220478504,72.56505287593365,,EL2
19.22541244650416,84.16201957734138,,EL2
16.485217437421102,78.70462348256127,,EL2
32.33920284711591,,0.8803891280405052,EL2
30.773377127639883,,0.8888562024113731,EL2
18.638227801700644,,0.9735269461200518,EL2
18.83395601663515,,0.965059871749184,EL2
30.773377127639883,,0.8803891280405052,EL2
39.581146799692554,,0.8888562024113731,EL2
13.549294213403547,,0.9904610948617876,EL2
16.2894892224866,,0.9396586486365802,EL2
73.8335844132307,125.09249028819225,,EL5
36.05803893087148,118.27074516971712,,EL4
25.880171754277285,147.6042491791602,,EL5
67.17882510545758,133.96075894220994,,EL5
33.12211570685392,142.8290275962276,,EL5
50.73765505095926,142.8290275962276,,EL5
61.50270687235697,136.6894569896,,EL5
37.428136435413,142.1468530843801,,EL5
37.232408220478504,144.19337661992265,,EL5
68.7446508249336,128.50336284742983,,EL5
79.70543086126581,,0.5332390788349226,EL5
36.25376714580599,,0.6009756738018657,EL5
87.73028767358046,,0.6263768969144692,EL5
49.56328576135224,,0.5671073763183941,EL5
44.67008038798965,,0.6009756738018657,EL5
17.072402082224617,,0.5925085994309978,EL5
14.527935288076065,,0.6517781200270728,EL5
58.37105543340491,,0.516304930093187,EL5
```

# Appendix E

# Assignment Prompt to GPT

One of the images is the figure from the paper. The other one is a reconstruction of the image.

Based on the similarities:

- Provide the title of the cluster including the legend name and number.
- Provide which y-axis the cluster belongs to.
- Provide the title of the desired y-axis.

Please provide the answers to the aforementioned questions only in the following string format:

```
'{
    \"clusters\": [
        {
            \"title\": [ CLUSTER NAME AND NUMBER IN STRING FORMAT ],
            \"axis\": \"left\" OR \"right\",
            \"axis_title\": [ NAME OF THE RELEVANT AXIS ]
        },
        {
        # ADD MORE CLUSTERS HERE
        }
    ]
}'
```

# Appendix F

# Baseline Prompt to GPT

Using the image provided, answer the following questions:

- What is the current at 3 V for each of the types?
- What is the current at 6 V for each of the types?
- At what voltage, is the current highest for each of the types?

Please provide the answers to the aforementioned questions in a list of tuples with the type and the answer.

# Appendix G

# Reconstruction Prompt to GPT

Using the points from the CSV file and the image provided, answer the following questions:

- What is the current at 3 V for each of the types?
- What is the current at 6 V for each of the types?
- At what voltage, is the current highest for each of the types?

Please provide the answers to the aforementioned questions in a list of tuples with the type and the answer.

# References


[1] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning transferable visual models from natural language supervision," 2021. arXiv: 2103.00020 [cs.CV].

[2] J.-B. Alayrac, J. Donahue, P. Luc, *et al.*, "Flamingo: A visual language model for few-shot learning," 2022. arXiv: 2204.14198 [cs.CV].

[3] OpenAI. "Chatgpt api," OpenAI. (2023), [Online]. Available: https://openai.com/api/.

[4] A. Masry, D. X. Long, J. Q. Tan, S. Joty, and E. Hoque, *Chartqa: A benchmark for question answering about charts with visual and logical reasoning*, 2022. arXiv: 2203.10244 [cs.CL].

[5] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar, *Plotqa: Reasoning over scientific plots*, 2020. arXiv: 1909.00997 [cs.CV].

[6] S. E. Kahou, V. Michalski, A. Atkinson, A. Kadar, A. Trischler, and Y. Bengio, "Figureqa: An annotated figure dataset for visual reasoning," 2018. arXiv: 1710.07300 [cs.CV].

[7] Z. Zheng, Z. He, O. Khattab, N. Rampal, M. A. Zaharia, C. Borgs, J. T. Chayes, and O. M. Yaghi, "Image and data mining in reticular chemistry using gpt-4v," 2023. arXiv: 2312.05468 [cs.AI].

[8] H. Kato, M. Nakazawa, H.-K. Yang, M. Chen, and B. Stenger, "Parsing line chart images using linear programming," pp. 2553–2562, 2022. DOI: 10.1109/WACV51458.2022.00261.

[9] J. Luo, Z. Li, J. Wang, and C.-Y. Lin, "Chartocr: Data extraction from charts images via a deep hybrid framework," Jan. 2021. [Online]. Available: https://www.microsoft.com/en-us/research/publication/chartocr-data-extraction-from-charts-images-via-a-deep-hybrid-framework/.

[10] C. Rane, S. M. Subramanya, D. S. Endluri, J. Wu, and C. L. Giles, "Chartreader: Automatic parsing of bar-plots," pp. 318–325, 2021. DOI: 10.1109/IRI51335.2021.00050.

[11] N. Siegel, Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi, "Figureseer: Parsing result-figures in research papers," pp. 664–680, 2016.

[12] M. Levy, R. Ben-Ari, and D. Lischinski, "Classification-regression for chart comprehension," pp. 469–484, 2022.

[13] F. Liu, F. Piccinno, S. Krichene, C. Pang, K. Lee, M. Joshi, Y. Altun, N. Collier, and J. M. Eisenschlos, "Matcha: Enhancing visual language pretraining with math reasoning and chart derendering," 2023. arXiv: 2212.09662 [cs.CL].

[14] F. Liu, J. Eisenschlos, F. Piccinno, S. Krichene, C. Pang, K. Lee, M. Joshi, W. Chen, N. Collier, and Y. Altun, "DePlot: One-shot visual language reasoning by plot-to-table translation," pp. 10 381–10 399, Jul. 2023. DOI: 10.18653/v1/2023.findings-acl.660. [Online]. Available: https://aclanthology.org/2023.findings-acl.660.

[15] A. Rohatgi, *Webplotdigitizer*, 2021. [Online]. Available: https://automeris.io/WebPlotDigitizer.

[16] A. Kirillov, E. Mintun, N. Ravi, *et al.*, "Segment anything," 2023. arXiv: 2304.02643 [cs.CV].

[17] "Computer graphics: The rgb color model," GeeksforGeeks. (Jul. 2022), [Online]. Available: https://www.geeksforgeeks.org/computer-graphics-the-rgb-color-model/.

[18] J. Bennett, *Webcolors documentation*, https://webcolors.readthedocs.io/en/latest/colors.html, 2008.

[19] N. Rahmah and I. S. Sitanggang, "Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra," *IOP Conference Series: Earth and Environmental Science*, vol. 31, p. 012 012, 2016, Published under licence. DOI: 10.1088/1755-1315/31/1/012012.