Tel Aviv University

The Iby and Aladar Fleischman Faculty of Engineering

School of Electrical Engineering

Deep Learning Final Project

Chart To Table: Hybrid NNs For Data Extraction From Scientific Charts

Roi Hizkiyahu

Roihezkiyahu@mail.tau.ac.il

Tzlil Lijishal

Tzlill@mail.tau.ac.il

October 18, 2023

Abstract

This project is part of a Kaggle competition with a crucial initiative to improve accessibility to STEM graphical materials for learners with disabilities. To achieve this, we have developed holistic solution for automated chart-type and data extraction from images. The solution utilizes a hybrid approach that integrates state-of-the-art deep learning models: YOLOv8 for object detection and classification, ResNet34 as an additional classifier, and PaddleOCR for text recognition. This synergistic pipeline is further fortified with diverse augmented datasets and meticulous pre and post-processing techniques. The code is publicly available at `https://github.com/roihezkiyahu/mga`.

Our approach's success is evident in the competition metric results, demonstrating its effectiveness and adaptability to previously unseen data with scores of 0.5 and 0.64 for the private and public test-sets, respectively. This accomplishment is reflected in our leaderboard ranking, securing the 26th and 86th positions among 609 contributors.

Looking ahead, future directions would be held in environments beyond the constraints of the Kaggle platform. These directions include exploring enhancements in object recognition using models like TrOCR by Microsoft, investigating alternative processing strategies, and incorporating large-scale and diverse synthetic datasets. These endeavors aim to address existing blind spots and enhance the pipeline's applicability in real-world, particularly in extreme use-cases.

1. Introduction

The objective of this project is to make the world a little better place for all humankind. While it may sound ambitious and quite dramatic, this aspiration is the driving force behind the Kaggle competition hosted by Benetech [1], in which we participated in as part of our final project. This competition is dedicated to improving accessibility for all learners regardless of their unique learning needs stemming from a range of physical or mental disabilities, including dyslexia, visual impairments, cerebral palsy, and more.

In today's rapidly advancing technological landscape, we have witnessed remarkable strides in enhancing access to resources that many of us often take for granted. As much of the recent advancements and breakthroughs in various fields can be attributed to science and technology, there remain areas where the system still falls short in providing adequate solutions. The disparities in accessibility are particularly pronounced within the STEM fields education. This is of utmost significance as these domains continue to make remarkable progress, and no aspiring innovator should be held back by unnecessary barriers.

One of the obstacles Benetech aims to overcome is the burdensome and costly manual process of making educational materials accessible. This challenge arises from the abundance of vital information in science and math textbooks presented through diverse chart types: bar, points, line and scatter, and hence its name - Making Graphs Accessible (MGA). These charts can take on numerous forms, each with unique style. Additionally, they may represent heterogeneous data on the x-axis, encompassing categorical, discrete and continuous features. Moreover, the elements within these charts are connected, with each point of interest influencing and being influenced by others [2].

The current project is a first step towards a noval approach that utilizes the power of artificial intelligence. It involves the transformation of data extracted from charts, including data series of x-axis and y-axis values, into a tabular format that can be later adapted to individual needs. To achieve this conversion, we have designed a pipeline that harness the capabilities of well-established deep learning concepts. Specifically, we make use of a fundamental property of convolutional neural networks (CNNs), which enables them to capture spatial features from chart images [3]. Further details on this process are provided in the methodology section, and its performances are presented in the results section.

2. Related Work

2.1. ChartOCR

As an extension of the competition framework, we have encountered previous work and literature regarding extracting tabular data from chart images. Some of these studies concentrated on specific types of charts [2,3], while others employed complex rule-based algorithms [4–7]. These approaches did not fully meet our requirements, as we aimed to efficiently and robustly cover all previously mentioned chart types. This is where ChartOCR model designed by Luo et al. (2021) comes into play. The deep hybrid framework underlying this model combines the advantages of deep-learning and rule-based [8]. It utilizes key point detection model to extract common chart elements regardless of the chart type. Then, it trains classifier for chart type, as an addition layers to the direct output of the detection model. Finally, they applied chart-dependent post-processing algorithm, type-specific object detection models and Microsoft OCR API.

Our approaches are similar in that we combine an object detection model, a rule-based post-processing algorithm, and the use of OCR. However, rather than training an additional type-specific object detection models, we employ the same initial object detection model to extract all chart components, both common and distinctive.

2.2. Kaggle Compatitors

Within the framework of the competition, numerous innovative approaches have emerged (e.g., top-3 solutions [9–11]), reflecting the collective dedication to solve the challenge. Similar to ChartOCR, these contemporary solutions vary in the structure of their pipelines and their strategies to handle each chart type, as well as the variety of charts as a whole. Furthermore, since the competition's database

was predefined, each competitor employed distinct methods to augment their training datasets, and identifying and addressing model failures to enhance performance.

3. Data

3.1. Training Datasets

The primary training datasets were provided by the competition host, comprising images and corresponding ground-truth annotations.

3.1.1. Images

In total, the first dataset consists of 60,374 images, each falling into one of five distinct chart types. We previously introduced four main chart types - bar, dots, line, and scatter. Notably, the bar charts encompass two subtypes: horizontal bars and vertical bars. Despite sharing common elements, particularly bar objects, these subtypes are treated differently within the dataset and thus along the pipeline. Moreover, each chart type exhibits a unique set of features and data characteristics. For instance, scatter charts consistently feature continuous data on the x-axis, while vertical-bar and line charts predominantly include categorical data. In contrast, dots charts may incorporate both continuous and categorical data. These distinct properties introduce complexity into the dataset, reflecting the real-world variability of chart designs and content. For a comprehensive understanding of chart conventions and specifications, please refer to the competition's documentation [12].

The initial images set comprises two distinct sources:

1. Extracted: This subset includes images derived from existing documents and materials, mostly scans from textbooks ($n_{extracted} = 1,114$).
2. Generated: These images were specifically created for training purposes, following the same chart properties ($n_{generated} = 59,260$).

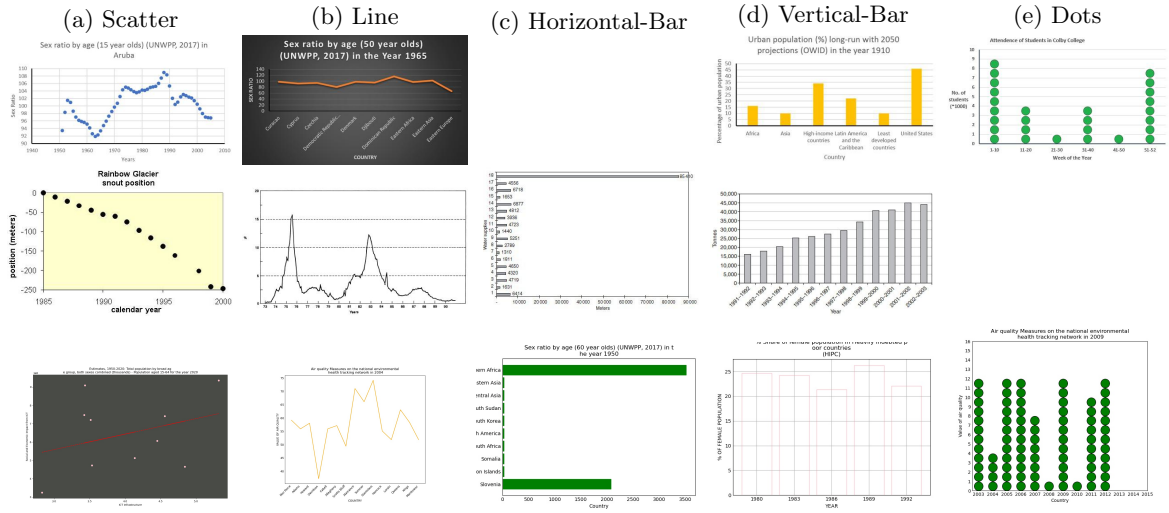


Figure 1: Sample of (upper) generated, (middle) extracted and (bottom) synthetic images by chart-type (a)-(e).

However, upon a thorough review of the provided images, it became evident that they alone did not offer enough variation to ensure robustness and coverage of all potential anomalies. Consequently, we took the initiative to synthesize additional augmented charts ($n_{synthetic} = 62,162$), focusing on specific nuances that we believed the models might encounter during training (for further explanation on use-cases see Appendix A). In Figure 1, you can see a snippet of images from all three sources categorized by chart type.

3.1.2. Annotations

The second initial dataset comprises a collection of JSON files containing exhaustive annotations for all images. These files provide essential information including the image source (e.g., "extracted" or "generated"); chart type; bounding boxes encompassing various components such as text, plots, etc., formatted as $(height, width, x_0, y_0)$; and, most importantly, the data series in the form of (X, Y) pairs (see Figure 2). A detailed description of the annotation structure can be found in [13].

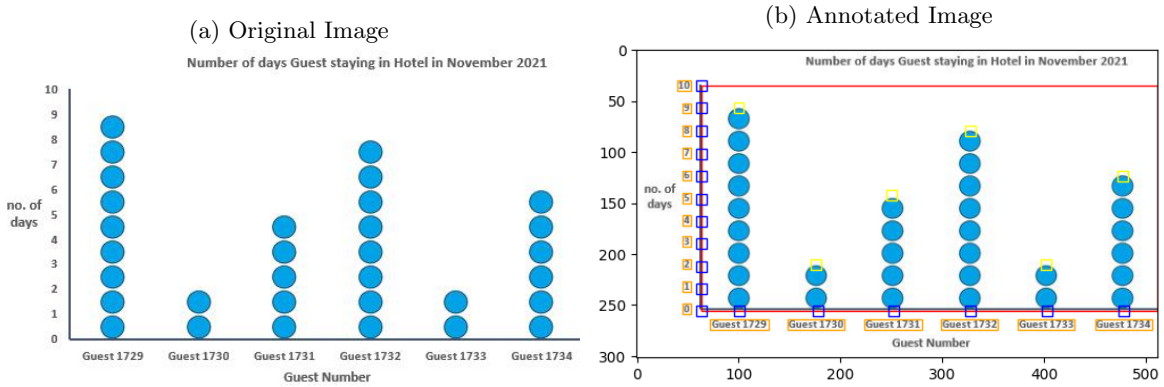


Figure 2: Demonstrating (a) the original image of a dot-chart and (b) its corresponding annotations, which serve as the ground truth in the training process.

During the dataset preparation phase, it was observed that some annotations contained errors or inaccuracies, including obvious mismatches with the chart type. Additionally, certain outliers were identified due to deviation from the standard dataset structure. For example, in Figure 2 the plot bounding boxes extend beyond the image right boundary, making it unsuitable for training. Table 1 provides a summary of the dataset volumes used to train the various models within our pipeline, categorized by source and chart type.

Table 1: Distribution of Chart-Types By Source

Source	Horizontal-Bar	Vertical-Bar	Dot	Line	Scatter
MGA: Extracted	73	457	0	423	165
MGA: Generated	0	18,732	5,131	24,519	11,078
Synthetic	7,835	14,085	13,658	16,053	10,531

3.2. Test-set

Alongside the provided image dataset, there are hidden public and private test sets. The complete test sets comprise approximately 4,000 images collected from professional sources. Notably, these test sets exclusively consists of extracted images and does not include any generated ones. The images extracted for the training and public test sets originate from the same sources. In contrast, the images in the private test set are sourced from a different set of materials. It's essential to acknowledge that there might be variations in the distribution of chart types, including special cases and outliers, between the public and private test sets.

4. Methodology

In this section, we outline our proposed solution for extracting tabular data from chart images. As we first stated in the introduction, our approach involves a multi-step pipeline of DL models designed to correctly classify the chart-type and accurately identify the data series using the chart components. The pipeline is visually depicted in Figure 3.

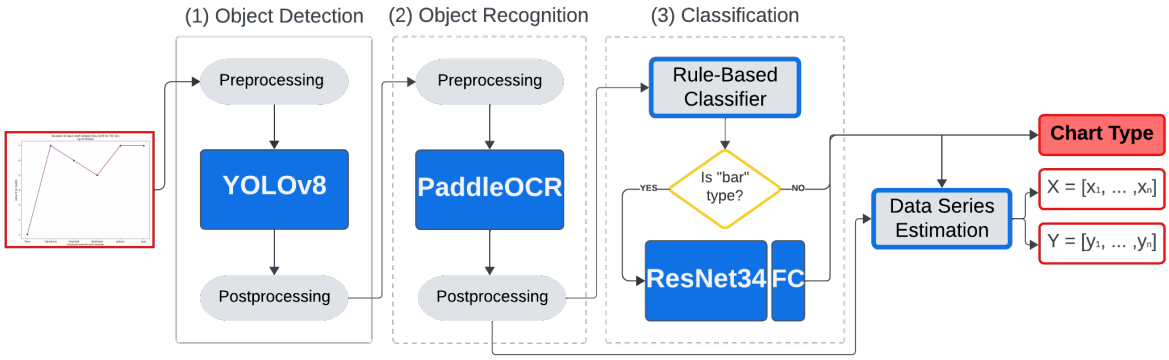


Figure 3: Overview of the multi-step pipeline for chart-type classification and data series extraction.

4.1. Object Detection

In the first step of the pipeline, our goal is to detect relevant visual elements, such as bars, dot points, line and axis ticks, and correctly categorize them. This step lays the foundation for subsequent text extraction and classification process.

The selection of YOLOv8 as the object detection model stems from its outstanding performance in a range of deep learning tasks, including object detection, image classification, and instance segmentation [14]. As of January 2023, YOLOv8 has demonstrated impressive accuracy on the COCO dataset, and has been recognized as the latest SOTA YOLO model suitable for these tasks. Comparatively, in the context of the other top-3 solutions [9–11] discussed in related work, they had employed DePlot [15], YOLOX [16] and MatCha [17].

A notable advantage of YOLOv8 implementation by `ultralytics` is its self-contained preprocessing capability, eliminating the need for extensive image preprocessing. With that being said, in order to enhance robustness against detection errors, we have added several postprocessing implementations:

1. *Line-chart NMS.* In addition to the non-maximum suppression (NMS) conducted by YOLOv8 during training, we have added another NMS step for line charts. In some instances, line charts may exhibit multiple splines, leading to over identification of points of interest within the plot bounding box. This final NMS step mitigates these duplicates with an IOU threshold of 0.1, as depict in Figure 4. The multiple identifications (green squares) in the left image, for example when $x =$ "2014", are reduced to a single point of interest in the right image.

2. *Removal of noisy elements.* Within the plot area, certain descriptive elements like legends can be mistakenly identified as points of interest. This action ensures that redundant visual elements are not erroneously included in the data series, as demonstrated in Figure 5.

3. *Tick labels pairing with corresponding axes.* Precise pairing of tick labels with the respective x or y-axis is performed for all chart types. Take a look at the pairing results in both Figure 4 and Figure 5, illustrated by the absence of the axes ticks in the right images as they are no longer needed for data extraction.

(a) YOLOv8 Bounding Boxes

(b) Postprocessed Bounding boxes

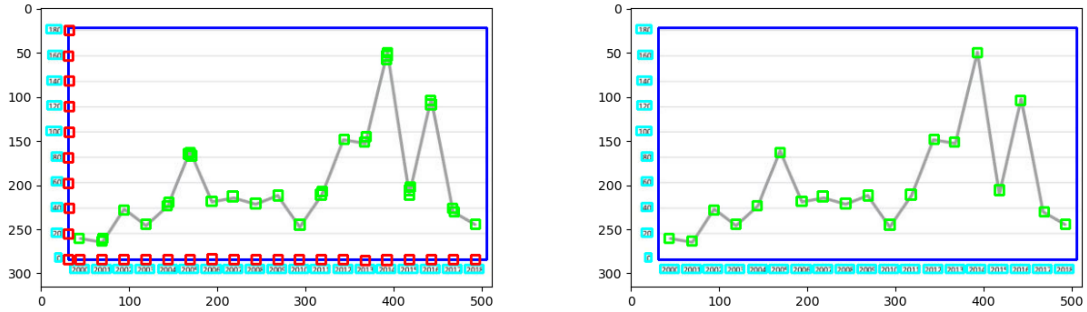


Figure 4: The effect of NMS on line chart annotations. (a) shows the original YOLOv8 output, while (b) displays the results after NMS correction.

(a) YOLOv8 Bounding Boxes

(b) Postprocessed Points-of-Interest

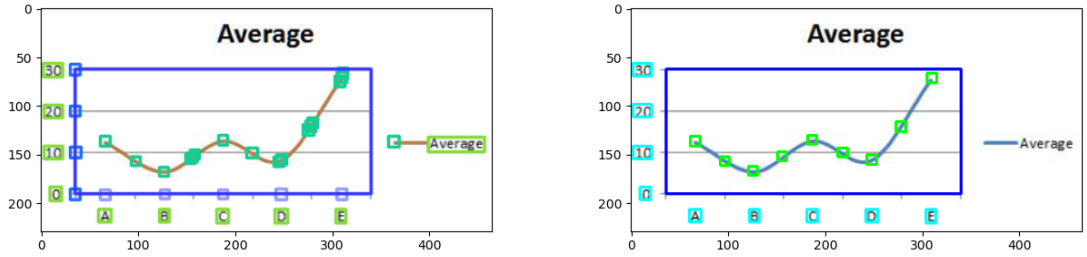


Figure 5: The removal of noisy elements falsly identified by YOLOv8. (a) represents the original YOLOv8 output, while (b) displays the relevant visual elements.

This comprehensive postprocessing ensures the reliable identification of essential chart elements, enhancing the quality of information fed into subsequent pipeline stages and ultimately improving final predictions.

4.2. Object Recognition

In the second step of our pipeline, we focus on precise text recognition within YOLOv8's postprocessed bounding boxes. To accomplish this task, we have exploited PaddleOCR by `PaddlePaddle` [18]. PaddleOCR is a reliable OCR framework known for its accuracy and versatility, making it an ideal choice for our diverse dataset, which includes text in various orientations and fonts. Moreover, compared to alternatives like easyOCR, PaddleOCR delivers superior performance, while still meeting our runtime constraints (unlike TrOCR).

In addition to the postprocessing conducted in the previous step, we implemented supplementary preprocessing techniques. These techniques are mainly designed to prepare and align text images, further improving OCR results. They include identifying the rotation angles of tick labels (which can take three primary orientations: 45 degrees, 135 degrees, or 90 degrees), determining text regions (i.e.

region of interest, ROI) and using a pixel threshold, followed by image cropping based on the adjusted ROIs. This process is illustrated step-by-step in Figure 6. In this example, the tilted label, "China", initially includes the bottom of the y-axis. After applying the preprocessing steps, the image becomes significantly easier to read, resulting in more stable and reliable OCR results.

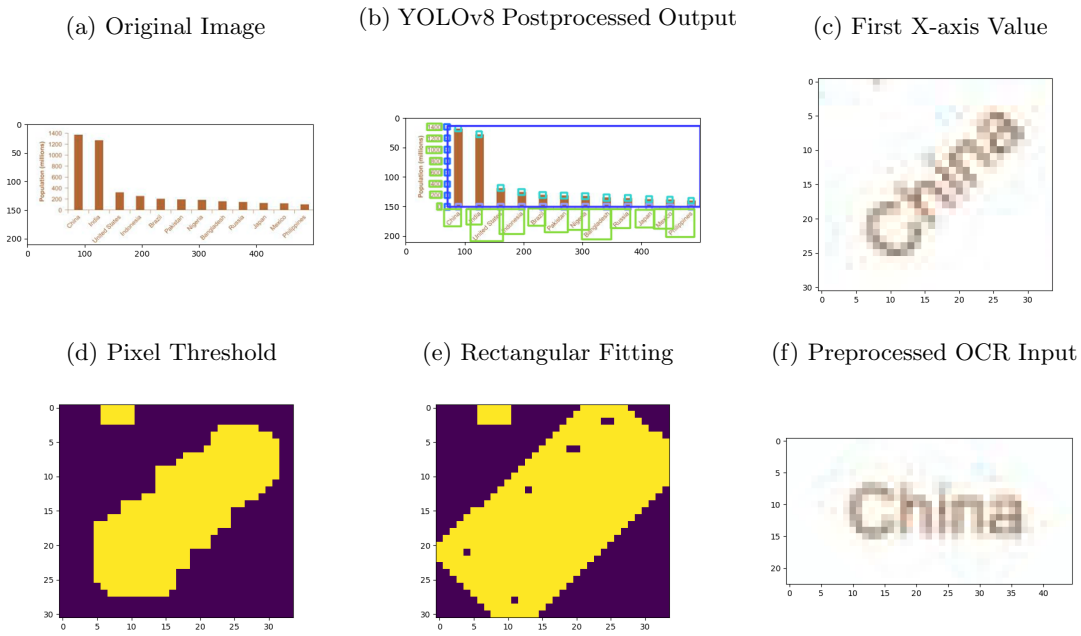


Figure 6: Illustration of label image transformations in the pipeline: starting with (a) the original input, followed by YOLOv8 postprocessing output in (b) and (c). Yellow areas in (d) represent the threshold and dilated image and (e) rectangle fitting. Lastly, (f) shows the preprocessed OCR input containing aligned text characters.

As with YOLOv8, to enhance the robustness of the pipeline and rectify potential fails in the current step, we applied postprocessing steps on the OCR output:

1. *Tick value correction.* To ensure accurate numerical data extraction, we corrected tick values based on the axis range and the number of ticks. This correction prevents unnecessary "jumps" and duplicated values.

2. *Adjustment of tick label coordinates.* Once rotated and validated, the tick label's bounding boxes are recalculated to align with the correct orientation. This adjustment ensures accurate positioning of the tick labels.

At this point, the necessary visual elements and information for extracting tabular data and determining the chart type are prepared. Nevertheless, there are a few more steps remaining to obtain the final predictions.

4.3. Classification

4.3.1 Rule-Based Classifier

This classifier employs a straightforward yet highly effective approach to distinguish between four chart types - line, dot, scatter, and bar charts, regardless of the orientation of the bars. It accomplishes this by analyzing the most common visual element class among the valid detected points of interest (POI). This simple rule-based method ensures rapid and dependable chart classification, thereby improving the efficiency of our pipeline.

4.3.2 ResNet34

To distinguish between horizontal-bar and vertical-bar charts, we utilized a pretrained model, ResNet, with 34 convolutional layers [19]. After conducting multiple trials with varying ResNet architectures (18, 34, and 50 layers), we determined that ResNet34 was the optimal choice. It demonstrated slightly superior accuracy compared to ResNet18 and equivalent performance to ResNet50, all while maintaining higher efficiency.

This step is fed with slightly preprocessed feature maps from YOLOv8 and PaddleOCR corrected output. The preprocessing includes resizing the images to 224×224 pixels and converting them to grayscale. To adapt the direct predictions of ResNet34 to our specific use case, we implemented a simple deep classifier consisting of five fully connected (FC) layers. These FC layers encompass varying dropout probabilities (e.g., $0.1 - 0.3$) for regularization, apply ReLU activation functions, and employ the Adam optimizer with a fixed learning rate of $1e-3$ and cross-entropy loss function.

4.4. Data Series Estimation

The (X, Y) series are derived from the refined chart elements and the chart-type prediction. In the case of categorical data, such as the x-axis in a line chart or the y-axis for a horizontal-bar chart, each POI within the plot bounding box was associated with the closest tick on the corresponding axis, and its value was adjusted accordingly. For continuous data, we identified the two closest ticks - one larger and the other smaller, and computed the value based on their relative distances. Specifically, we calculated the number of pixels between the two ticks and the range between them in the axis unit. Then, we applied linear interpolation to translate pixel increments into the changes in axis values. For instance, if the labels are $y_1 =$ "0" and $y_2 =$ "100", with 10 pixels between them, a 1-pixel increase equates to a 10-unit rise on the y-axis scale. Therefore, the Y value of a POI at pixel 5 would be $5 \times 10 = 50$.

At the end of this step, the pipeline generates the chart-type prediction along with the estimated data-series. This step is crucial for accurately retrieving the tabular data, which, as was mentioned, will serve as the foundation for advanced learnig methods.

4.5. Models Evaluation

As access to separate test sets was unavailable, we opted to create a validation set by partitioning the train-set ($p_{validation} = 0.1$). This validation set serves as a means to assess the performance of individual pipeline phases, as well as the overall system performance.

The metric we have used for evaluating the final pipeline was established by the competition host [20], and its implementation was provided in a code notebook [21] as well. The metric score, denoted as S, is a float value ranging from 0 to 1, where 0 represents a failure and 1 represents the desired level of success. First, results are discarded if either the chart-type or the data-series length is incorrect, resulting in $S_i = 0$ for those cases, where $i = 1, \ldots, N$. Subsequently, depending on the data type (categorical or numeric), the series is assessed using either the Levenshtein distance or RMSE loss, respectively, in comparison to the ground-truth values from the annotations. This evaluation is performed for both the x-axis and the corresponding y-axis series. Since these metrics operate on different scales, additional normalization is applied as follows:

$$z_i = \begin{cases} \frac{RMSE(v_i, \hat{v}_i)}{RMSE(v_i, \bar{v}_i)}, & \text{if } v_i \text{ numeric} \\ \frac{\sum_l Lev(v_i, \hat{v}_i)}{\sum_l \text{length}(v_i)}, & \text{if } v_i \text{ categorical} \end{cases}, \quad v \in \{x_i, y_i\}, i = 1, \ldots, N$$

Finally, the normalized values are rescaled using a sigmoid-type transform defined as follows:

$$\sigma(z_i) = 2 - \frac{2}{1 + e^{-z_i}}$$

The final score is the average of the x-axis and y-axis scores, i.e. $S_i = \frac{\sigma(x_i) + \sigma(y_i)}{2}$.

5. Results

5.1. Final Pipeline

The presentation of pipeline results begins with an overview of each component's training process. For the YOLOv8 model, the training spanned 300 epochs with early stopping triggered after 10 iterations, typically concluding around 180 epochs. After each epoch, the model with the best performing on the validation dataset was saved. Figure 7 showcases the training progress on both the train-set and validation-set. The plot depicts the decay of the training loss, i.e. cross-entropy (denoted as *cls_loss*). Additionally, various metrics, including recall and precision, offer a nuanced understanding of the classification component's performance. Notably, the model achieved remarkable results in both metrics, surpassing the inherent trade-off between them.

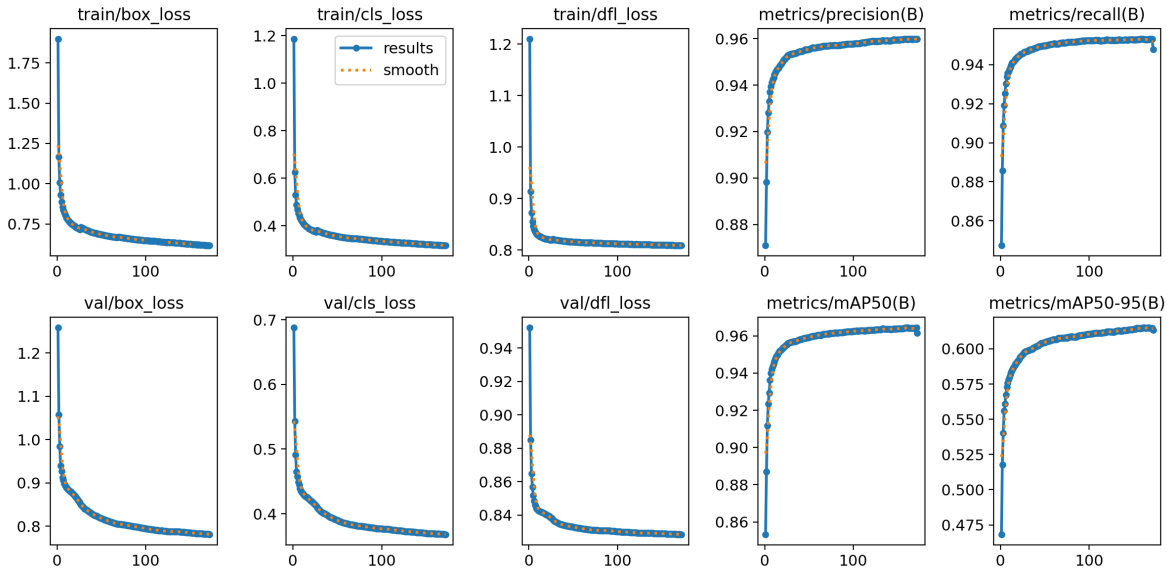


Figure 7: YOLOv8 training process evaluation on the train-set and validation-set.

The ResNet34 model's classifier layer underwent training for 35 epochs, employing early stopping after 5 iterations. The hybrid classification approach, combining YOLOv8 for visual elements and ResNet34 for bar-chart classification, yielded an outstanding accuracy of 99.82% on the validation-set. Lastly, the evaluation of text recognition and subsequent data estimation components utilized the overall competition metric. The obtained scores for the public and private test-sets were 0.64 and 0.5, respectively. Figure 8 serves as a sample of selected examples showcase the effectiveness of our pipeline in addressing unique and complex scenarios, such as instances (a)-(b) with diverse scatter charts, (b) scenarios involving charts with long text labels, and dense visual elements (a) and (d).

Beyond assessing our hybrid pipeline's effectiveness, it is particularly intriguing to see its position on the competition leaderboard. In this contest, there were 610 teams who collectively submitted over $10K$ entries. The top 6 ranked solutions received prizes, making it a highly competitive field. We can observe in Figure 9 (a) the distribution of scores for all competitors based on the private test-set. The dashed green line indicates the score required to win a prize, which was $S_{(6)} = 0.68$, while the gray line represents our solution's score, i.e. $S_{(26)} = 0.5$. Fortunately, our proposed pipeline performed quite well, securing us a flattering 26-th place in this challenging and valuable competition.

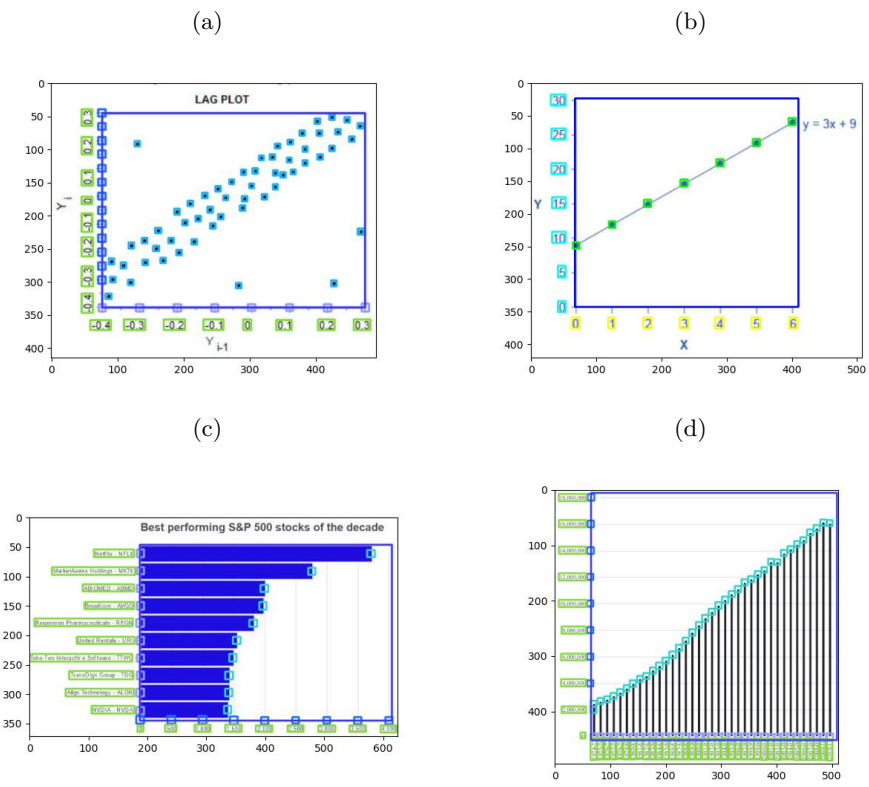


Figure 8: Examples of correctly predicted use-cases by the proposed pipeline

Furthermore, we wanted to assess the generalization capabilities of our pipeline beyond the public test-set. To achieve this, we analyzed the change in the score from the public to the private test-set, considering that the public test-set shared similarities with the train-set, whereas the private test-set did not. As shown in Figure 9 (b), our solution notably generalized well with score difference of only 0.13. This is a good indication that our augmentations and processing manipulations throughout the pipeline effectively contributed to its adaptation to previously unseen data in comparison to other competitors.

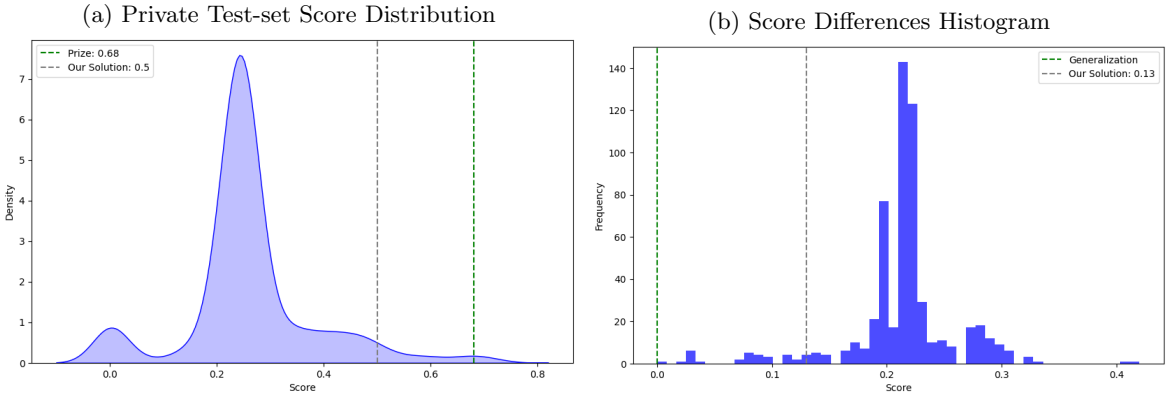


Figure 9: Comparison of our solution's performance: (a) distribution of scores in the private test-set and (b) histogram of score differences in the public and private test-sets.

5.2. Models Comparisons

Prior to finalizing our optimal pipeline, we extensively explored various model configurations, classifier selections, and OCR frameworks to pinpoint the critical factors influencing accuracy and robustness. In this subsection, we offer valuable insights into the effectiveness of the selected components within our data extraction pipeline.

5.2.1. Single YOLOv8 vs Line-chart YOLOv8

The motivation behind this comparison stems from the sub-optimal results we initially obtained when dealing with line charts. Line charts often feature complex elements, including lines with splines and points, making their detection and accurate representation challenging. We aim to assess whether the performance of additional YOLOv8 model, specifically tailored for line charts, can yield improved results. As depicted in Figure 10, the dedicated YOLOv8 model designed exclusively for line charts did not achieve higher accuracy in detecting the "line_point" visual element.

(a) Single YOLOv8 (b) Line Chart YOLOv8

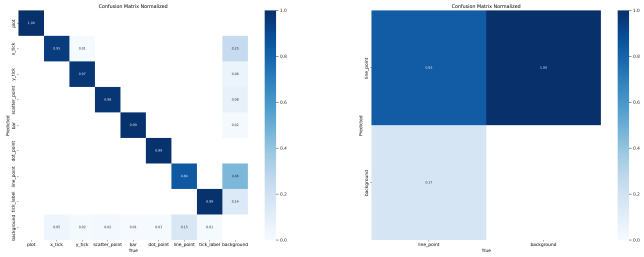


Figure 10: Normalized confusion matrices of (a) single YOLOv8 model for all chart-types compared with (b) additional line chart YOLOv8.

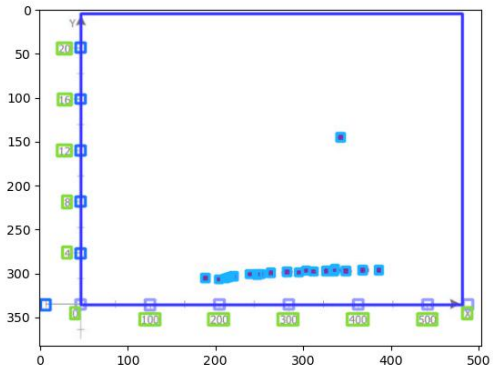
These results indicate that the challenge lies beyond the choice of YOLOv8 architecture. Therefore, resolving the line chart complexity should be addressed in a different approach, such as considering other classification models, enriching the augmented data with a variety of line chart instances, and improving the processing manipulations.

5.2.2. Classification: ResNet34 vs YOLOv8 & ResNet34

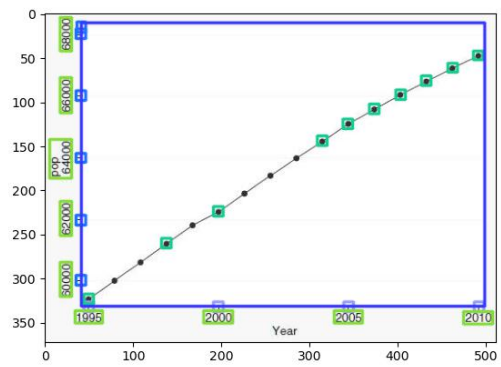
Here, we explored the performance of two classification approaches: (1) ResNet34 classifier for all five chart-types and (2) a hybrid approach that combines YOLOv8 for 4 chart-types along with ResNet34 for identifying the bar chart-types (hereafter referred to as "ResNet34" and "YOLOv8"). Based on the evaluation results, it becomes evident that the ResNet34 classifier underperformed in comparison to the second approach. The ResNet34 classifier achieved a score of 0.47 on the private dataset, 0.61 on the public dataset, and an accuracy of 99.28% on the extracted images.

Figure 11 showcases a selection of images that were misclassified by at least one of the classification approaches. Upon closer examination of the scenarios where ResNet34 failed, it appears that it struggled in correctly identifying line charts, especially when they contained a large number of data points. Conversely, ResNet34 occasionally misclassified scatter charts when the points were densely clustered and resembled a line-like shape. In contrast, YOLOv8 also encountered some difficulties in these scenarios, but the frequency of such failures was notably lower. Furthermore, in one of the most challenging cases represented by scenario (d), where the image contained both bars and lines, both approaches erroneously predicted "line".

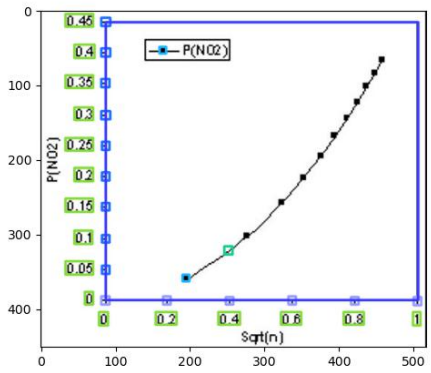
(a) Scatter: YOLOv8 - "scatter", ResNet34 - "line"



(b) Line: YOLOv8 - "line", ResNet34 - "scatter"



(c) Line: YOLOv8 - "scatter", ResNet34 - "line"



(d) Vertical-Bars: YOLOv8 - "line", ResNet34 - "line"

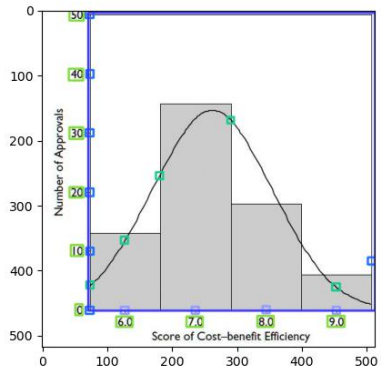


Figure 11: Examples of images misclassified by one or two classification approaches: (a) Scatter chart where YOLOv8 identified them as "scatter," while ResNet34 labeled them as "line". (b) Line chart recognized as "line" by YOLOv8 but classified as "scatter" by ResNet34. (c) Line charts labeled as "scatter" by YOLOv8, while ResNet34 identified them as "line". (d) Vertical-bar chart where both YOLOv8 and ResNet34 categorized them as "line".

The persistent challenge in accurately predicting line charts highlights the need for further refinement and exploration.

5.1.4. PaddleOCR: pre-trained vs fine-tuned

The motivation behind comparing the utilization of the pre-trained PaddleOCR model in our pipeline versus fine-tuning PaddleOCR on our dataset lies in the pursuit of optimizing text recognition performance, with a specific focus on addressing the challenges posed by 76,255 images. Fine-tuning PaddleOCR with optimized hyperparameters on our specific dataset allows us to tailor the model to the unique characteristics of the images, especially on the private test-set images. The fine-tuned model failed to surpass the performance of the pre-trained PaddleOCR model, suggesting that the generic features captured by the pre-trained model are well-suited for our task.

5.1.5. Ablation Study

In this part, we wanted to asses the impact of each component and processing step on the overall performance of our data extraction pipeline. The study involves systematically removing specific elements and analyzing the resulting changes in our solution score on the test-sets. The results are summarized

in Table 2. The first row of the table showcases the performances of our final solution. Each subsequent row represents a trial where a particular step or component was excluded from the pipeline. For example, the second row illustrates the pipeline's performance when we conducted the prediction without using the augmented charts we synthesized, and this process was repeated for each relevant pipeline element.

Table 2: Ablation Study Results (Competition Metric)

	Public Test-set	Private Test-set
Final Pipeline	0.64	0.5
(-) Data Augmentation	0.56	0.39
(-) YOLOv8: Postprocessing	0.64	0.51
(-) OCR: Preprocessing	0.62	0.5
(-) OCR: Postprocessing	0.64	0.5

The results in the table emphasize the importance of image manipulations and adjustments throughout the pipeline to enhance performance, considering the characteristics of the private test-set. With that being said, it is observed that we could omit the YOLOv8 postprocessing step.

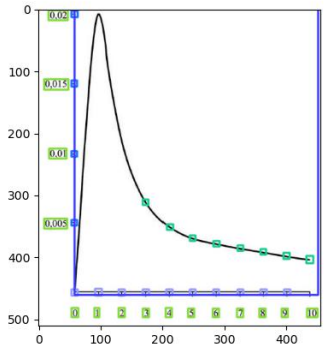
5.3. Failures Diagnosis

During our quest to find the optimal models for the pipeline, we have uncovered certain blind spots our pipeline has yet to eliminate effectively. The following examples highlight some of the key issues.

The primary challenge revolves around line charts, particularly those representing continuous variables (as in Figure 11(c) and Figure 12(a)), as opposed to line charts that connect values of categorical variables (as in Figure 4, Figure 5, and Figure 11 (b)). In the latter case, the model successfully identifies the spikes, making it easier to detect the corresponding ticks. However, in the former case, the line's trend does not correlate with the tick positions. The second challenge, illustrated in Figure 12(b), arises when the text in the tick labels is excessively long. While additional post-processing of YOLOv8 and preprocessing of PaddleOCR improved the results for single-line long labels and multi-line short labels, it had a limited effect on multi-line long text labels. Lastly, we encountered challenges related to text inside plot bounding boxes, such as legends, regression equations in scatter charts, and values near points of interest. Figure 12(b) serves as an example of this issue as well.

To overcome these challenges, we have added more complex examples to the synthetic dataset, as described in Appendix A, but additional improvements are required.

(a) Continuous X-Axis



(b) Long Text Labels and Text Inside Plot

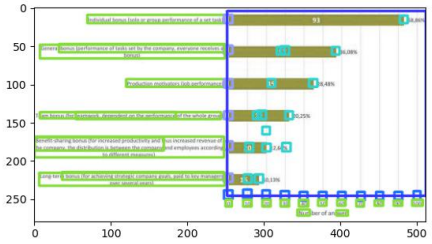


Figure 12: Examples of solution failures in (a) continuous line-chart and, (b) long text labels for ticks and text inside plot bounding-box

6. Conclusion and Future Work

The thorough exploration and navigation of our solution yielded valuable insights into the difficulties and successes encountered during the data extraction process. We have learned that a hybrid approach, combining YOLOv8, ResNet34, and PaddleOCR, showcased promising results for various chart types, excluding extreme instances of line charts. Moreover, its adaptability to previously unseen data is particularly noteworthy. Achieving evaluation metric of 0.5 with a respectful 26th place ranking on the leaderboard highlights its potential applicability in real-world scenarios - the main purpose of this challenge. It is gratifying to know that by providing an open-source solution to this impactful cause, we have contributed, even if in a small way, to bridging gaps in technology infrastructure and thus improving education in STEM materials for learners worldwide.

Although we have conducted several examinations of alternative approaches and implementations throughout this project, all of them were under memory and offline limitations on Kaggle platform. Future applications may involve deploying the pipeline in environments where automated data extraction from images and text would be indispensable. Furthermore, the continuous evolution of the pipeline could benefit from ongoing advancements in deep learning and computer vision. Future extensions could delve deeper into refining some pipeline components, such as the object recognition component by perhaps utilizing TrOCR by Microsoft [22]; exploring alternative processing strategies; and integrating large-scale, diverse synthetic dataset. This could be a promising avenue for future exploration. Hopefully, these enhancements would overcome the blind-spots our current solution did not manage to illuminate, as described in the previous section.

References

[1] Benji Andrews, benjiaa, HCL-Rantig, Hema Natarajan, Maggie, Ron Ellis, and Ryan Holbrook. Benetech - making graphs accessible, 2023. https://kaggle.com/competitions/benetech-making-graphs-accessible.

[2] Xiaoyi Liu, Diego Klabjan, and Patrick NBless. Data extraction from charts via single deep neural network. https://doi.org/10.48550/arxiv.1906.11906, 2019.

[3] Jinho Choi, Sanghun Jung, Deok Gun Park, Jaegul Choo, and Niklas Elmqvist. Visualizing for the non-visual: Enabling the visually impaired to use visualization. In *Computer Graphics Forum*, volume 38, pages 249–260. Wiley Online Library, 2019.

[4] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi. Figureseer: Parsing result-figures in research papers. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*, pages 664–680. Springer, 2016.

[5] Abhijit Balaji, Thuvaarakkesh Ramanathan, and Venkateshwarlu Sonathi. Chart-text: A fully automated chart image descriptor. *arXiv preprint arXiv:1812.10636*, 2018.

[6] Jorge Poco and Jeffrey Heer. Reverse-engineering visualizations: Recovering visual encodings from chart images. In *Computer graphics forum*, volume 36, pages 353–363. Wiley Online Library, 2017.

[7] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. Revision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 393–402, 2011.

[8] Junyu Luo, Zekun Li, Jinpeng Wang, and Chin-Yew Lin. Chartocr: Data extraction from charts images via a deep hybrid framework. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1916–1924. IEEE, 2021.

[9] YumeNeko. Benetech 1st place solution, 2023. https://www.kaggle.com/competitions/benetech-making-graphs-accessible/discussion/418786.

[10] Hallstatt. Benetech 2nd place solution, 2023. https://www.kaggle.com/competitions/benetech-making-graphs-accessible/discussion/418430.

[11] CamemBert. Benetech 3rd place solution, 2023. https://www.kaggle.com/competitions/benetech-making-graphs-accessible/discussion/418420.

[12] Making graphs accessible: Graph conventions, 2023. https://www.kaggle.com/competitions/benetech-making-graphs-accessible/overview/graph-conventions.

[13] Making graphs accessible: Dataset, 2023. https://www.kaggle.com/competitions/benetech-making-graphs-accessible/data.

[14] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Yolo by ultralytics, 2023. https://github.com/ultralytics/ultralytics.

[15] Fangyu Liu, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. Deplot: One-shot visual language reasoning by plot-to-table translation. *arXiv preprint arXiv:2212.10505*, 2022.

[16] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[17] Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Martin Eisenschlos. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. *arXiv preprint arXiv:2212.09662*, 2022.

[18] PaddlePaddle. PaddleOCR. `https://github.com/PaddlePaddle/PaddleOCR.git`, 2023.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Making graphs accessible: Evaluation, 2023. `www.kaggle.com/competitions/benetech-making-graphs-accessible/overview/evaluation`.

[21] Making graphs accessible: Evaluation code, 2023. `https://www.kaggle.com/code/ryanholbrook/competition-metric-benetech-mixed-match?scriptVersionId=123629556&cellId=1`.

[22] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. Trocr: Transformer-based optical character recognition with pre-trained models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13094–13102, 2023.

Appendix A

Synthetic Charts

This appendix offers further details on the supplementary augmented charts meticulously generated to closely emulate the anticipated complexities within the assumed test-sets. Moreover, some use-cases were not part of test-sets, but we thought they would help in model generalization. We have employed the `matplotlib` library, to precisely craft plots to encompass all facets of graph conventions. The following provides a breakdown of the principal non-trivial augmentations.

Axis Ticks

- Long tick labels with multiple rows.
- Numerical tick labels with various formats (e.g., scientific notation, different rounding, European-style formatting, adding % as a suffix, currency symbol as a prefix, etc.).
- Inclusion of tick labels not corresponding to the data series.
- Unequal spacing between x tick labels.

Axes

- Reversed y-axis.
- Shared origin when the line starts at point $(0, 0)$.

Plot

- Horizontal-bar charts due to its absence from the extracted training-set.
- Extreme aspect ratio (very wide to very narrow).
- Continuous with complex trend (e.g. log-log, semi-log plots etc.)
- Blank plots - only "background" POI inside the plot bounding-boxes. Primarily intended to reduce false detection, as can bee seen in Figure A.1.

Figure A.1: Examples of blank plots added to improve preciseness by reducing false detections.