Counting Pixels for an Effective Axis Detection

Lee Keeheon*
Creative Technology Management
Yonsei University
Seoul, Republic of Korea
keeheon@yonsei.ac.kr

Sohn Eury
Creative Technology Management
Yonsei University
Seoul, Republic of Korea
ylee12@yonsei.ac.kr

Ryu Kunhee
Creative Technology Management
Yonsei University
Seoul, Republic of Korea
rgh00826@yonsei.ac.kr

***Abstract*—Scientific documents often use the aid of non-textual components such as charts and tables to illustrate their points. These non-textual components often have integral data needed for deeper understanding of the topic. It is in the scientific community's best interest to ease the viewing of charts for the furtherance of development. This includes automatic transference of charts into a digitized form for better access in the academic community. Our axis detector aims to focus on one specific part of chart digitization; axis detection. The biggest hurdle in axis detection include lack of accuracy when lacking user-sided data to help supervise the process. Focusing on line-charts and bar-charts, our axis detector will create a fully automated system of axis detection in a less-intensive and an accurate manner using OpenCV and pixel detection system.**

***Index Terms*—Computer Vision, Axis Detection, Data Extraction**

I. Introduction

Axes are the most important part in a chart to a reader as well as a writer. A reader interprets a chart and comprehend information based on axes efficiently. A writer transforms data into a chart, a graphical representation of data visualization. The writer can effectively deliver some information to the reader in summary rather than to show the raw data, numerical results, generated from an experiment. The writer chooses an appropriate type of a chart that can deliver the information. Two popular charts are a line chart and a bar chart.

A line chart connects points against horizontal and vertical scales. It is to show changes of y over x. A reader first look at the axes and then at the points in the lines in the line chart[2]. The reader can find patterns including trends, rises and falls, repeats, and crosses. A bar chart sets up bars to represent quantities for a range of different categories. A reader first look at the axis perpendicular to the bars to identify what it is, and then the axis parallel to the bars to identify its quantity. The reader can not only identify the absolute quantity of each bar and the distribution of the bars, but also compare all the bars.

This work was supported by the Technology development Program (S3149215) funded by the Ministry of SMEs and Startups, Sejong City, Korea.

In many cases, to grasp the information delivered by the charts may be sufficient. However, in some other cases, we need to consider the information critically and the synthesis of data for the relevant studies. Then, we need to understand results in detail, e.g., in a raw data form. We used to extract the data from the charts manually. Some may look at the charts and guess the quantities corresponding to the points in a line chart and the bar lengths in a bar chart. The others may ask the writers (authors) of a research paper for the raw data. Now, we have digitized charts in form of images and advanced computer vision technologies so that automatic data extraction is possible.

In automatic data extraction from a chart, accuracy of correct data extraction and time of the data extraction are the major performance measures to consider, not the newness of a machine-learning algorithm that is used. The algorithm can be based on rules or can generate rules[3]. In any cases, the performance measures are dependent on axis detection [1]. If an axis is incorrectly parsed, it will lead to incorrect extraction of values. Therefore, in this paper, we propose an efficient and effective rule-based algorithm to recognize axes.

a) Layout of line and bar charts: Most line-charts share common structures which we are able to use for axis detection[1].

A line chart contains a title, an x-axis, y-axis with value lines. One should note that there are lines within the chart that are not axis lines but placed in ease for human reading. Figure 1. shows these lines as a grey horizontal line across each of the values in the x-axis. These lines are not considered as axis. The value lines represent the data points in the chart, giving information.

A line-chart can be told in 4 regions: axes, graphical information, legend, and text. The axes are x and y

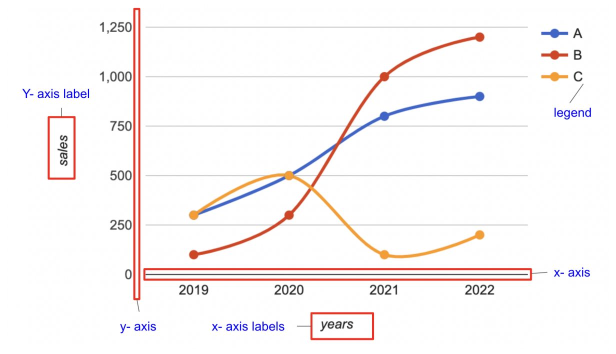

Fig. 1. An example of a typical line graph. Features x and y axis, its values and a legend with values formed as a line

axes which are typically perpendicular to themselves and the edges of the graph. The space between the and x and y axes contain graphical information, which has the values and data points. The legend region show the corresponding colours for the values in the line-formatted chart. The text region contains information the chart title, x-axis and y-axis title and the value points which the information inside the chart follows.

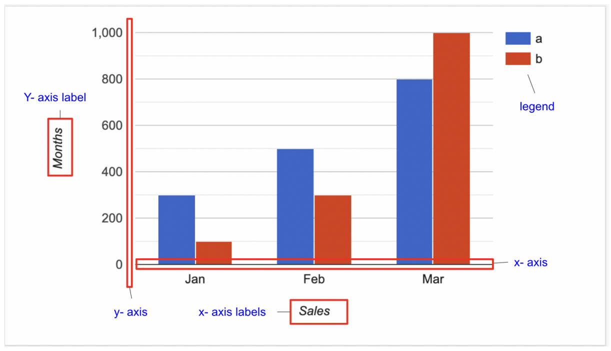

Fig. 2. An example of a typical bar graph. Features x and y axis, its values and a legend with values formed as a bar.

Alike line-charts, most bar-chart share common structures which we are able to use for axis detection. A bar-char can also be categorised into 4 regions: axes, graphical information, legend, and text. The descriptions of the bar-charts are similar to ones in the line-charts excepts the values are presented in a vertical fashion, as bars[11].

As mentioned previously, the current study of chart digitisation are based on manual inputs and works as followed;

1) A user inputs an image file of a plot
2) The program prompts the user to detect x and y positions
3) The program prompts the user to assign a value the axes

The program prompts the user to submit data points for the graphical information Hence, most of the work is done manually in a slow, unscalable manner. It relies on the user's accuracy and consumes their time. Our approach aims to improve step 2 where the program prompts the user to detect x and y positions by manually detecting, assisting the use of digitisation in an autonomous large scale.

Our novel contribution to the field are as follows:

1) A faster, less intensive program for axis detection.
2) Using a coloured detection system based on RGB values
3) Pixel counting method for Axes Detection

II. Related Works

As the challenges of accurately digitizing bar-graphs and line-graphs have been a an interest in the scientific community, there has been various works which focus around building models to extract information, which includes axes detection as well.

a) Deep Neural Networks (DNNs): Some works use DNNs as a tool to parse information. This is either done in whole where they detect the entirely of the chart and the values without separating the different graphical features inside, or is done separately for different regions.[2] DNN is fed an idea of what type of object to detect and they treat the axes and bars as different subjects. Siegal et al (2016) [9] used a convolutional neural network (CNN) to detect the axes in their information extraction process. Their axes detection only related to understand their positions and scales, in order to understand that the picture extracted from the PDF is indeed true.

However, as iterated previously, this method focuses on the extraction of figures from paper, not the detection of axes. Zhang et al (2021)[7] also uses DNN, purposing as object detection by using Cornernet which as a key point detection technology. Using pattern recognition, they predict the chart's structural element and recreate it in a digitised form. The accuracy for axis detection across all types of graphs were high, over 90% averaging a 2.4 second for parsing.

b) Rule-based algorithms: Rane et al (2021)[1] employs the ChartReader algorithm to detect the axes in part of the step to parse figures. Their detection follows the step of concerting the input figure into binary gray-scaled image and implementing a threshold. The threshold is decided as 200 based on observational data. Then this matrix is scanned vertically to identify a continuously black line pixels. The axes are identified by the row which has the longest maximum number of continuous black pixels.

However, one does not need to only use either DNN or rule-based methods. Luo et al (2021)[6] uses a combination of the two, focusing on the latter. After extracting the keywords via range extraction, they use type specific detection based on the rules pertaining to each type to parse through the information. The combination of the two creates the numerical values.

III. Methods

A. Overview

As mentioned previously, a line chart is comprised of four elements: axes, graphical information, legend, and text. These features are shown in different colours to represent the values. In most instance, graphs are represented in a limited about of colour schemes: black, white, green, red, blue, yellow, and grey as shown in Figure 4.

In instances like this, a colour scale based detection method is needed. To conduct such detection, few points of attention arises. First, we must be able to understand the context of the image. As every graphical feature contains different colour schemes which may be similar in human eyes but have widely different RGB figures, the detection threshold must be optimized for each picture instance in a localised manner. Thus, instead of setting a specific colour to detect, the colour to detect is found by finding the maximum number of similar colour (RGB values) pixel in each row and column of the figure. Unlike the previous approaches, the axes detection is primarily done in coloured-scaled instead of grey or binary.

a) Process of our axis detector: As shown in Figure 3, when the user inputs a chart of either line or bar figure, this image is converted into a matrix of pixels values. These values are represented as RGB values as the model detects them. By setting the RGB values into the set threshold, the model select those who are in the colour scheme of humanly discernable black and grey. These pixels are selected as axes and highlighted. Following diagram is a pseudocode of our algorithm, explains our algorithms process from RGB detection to drawing two lines for axes.

Algorithm 1: Axes detection algorithm

```
Result: An image with two red lines represents
        Y and X-axis
while While each images in file path do
    read image;
    if R, G, and B values from image in range of
      threshold then
        assign 1;
    else
        assign 0;
    end
    group sum by row and column
    count largest sums
    if largest sum is too small then
        do not append to a list;
    else
        append to a list;
    end
    find the index of the largest sum in a list
    draw two lines at the two index for each x
      and y axis
end
```

B. Threshold

a) Colour detection: Conventionally, the colour scale for the axis in a computed line chart by any software such as LaTeX follow a similar scale. The darkest colour, black, is shown as (0,0,0) and white is shown as (255, 255, 255) while intermediate colours such as gray dall between the two values. However, although indiscernible to the human eye, the colours of black and white can have different values that differ far from the standard scale.

This discrepancy between RGB colours grow bigger than approaching the colour grey. The colour grey, as a gradient can lie from any scale between the two and they share no pattern. This is not noticeable nor distinguishable until a RGB colour picker has been applied to the image, as shown in Figure 4. The colour on the left represents RGB (192, 192, 192) while on the left, it represents RGB (192, 192, 191). The difference in one tic Even if the values of the Red, Green, and Blue differ from each other, they share the same shade. Hence, if one was to apply a threshold based with no localised context, merely applying it to a number based on universal context, it would lead to a high margin of error. To apply a threshold universally would create errors in an instance especially when the three RGB values to the match each other. As the threshold has to

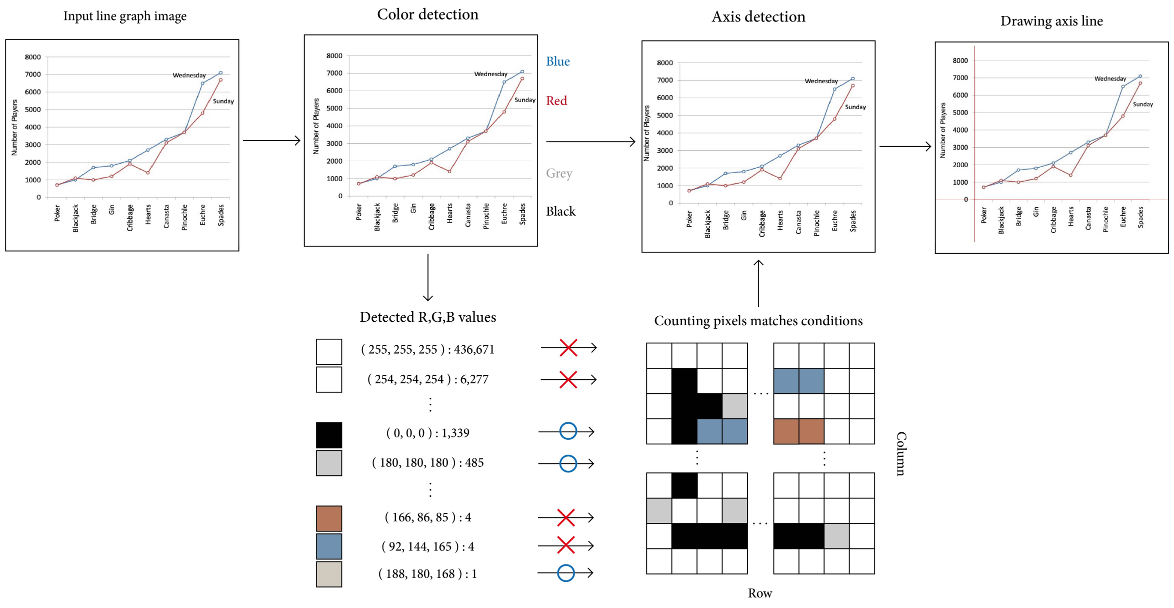

Fig. 3. Flowchart of the axis detector

be specified for each R, G, B value, we set the threshold has a value of each grey pixel to be a median value with each pixel ± 10.

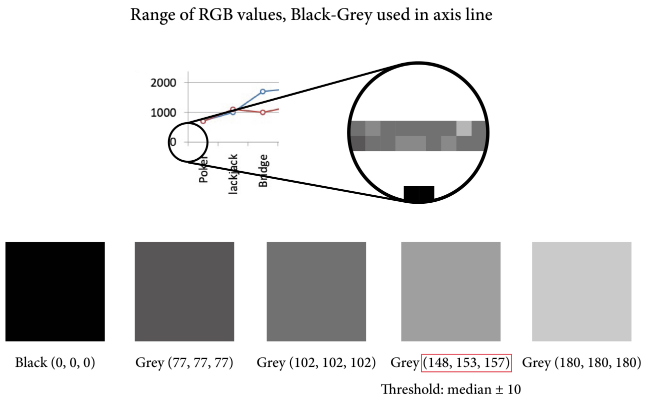

Fig. 4. Human eyes recognize the axis as a single line, but when it comes to the computer, it draws a single line through adding up few pixels together. The problem is that the pixels which have similar R, G and B values appear same to human eyes, but computer differentiates it.

b) Localisation: To increase the accuracy of axes detection, localisation is employed. Localisation in this instance is defined by gathering the context of the image to build the foundation of the model, helping to implement rule-based execution to decrease the margin of error. To conduct localisation, it requires finding a threshold for the line colour detection. This is done most easily by clarifying the area for the initial detection in the graph.

This is demonstrated in Figure 2, where there are several lines which are not the x-axis in the graphical area. The line highlighted red is the only x-axis in the graph, yet there are few other grey lines in the chart which are also continuous and one colour. The grey value lines appear frequently for each value correlated in the y-axis in the whole image including the top and the bottom.

This may be a factor in detection as our axis detector detects the frequency of colour occurring each row and column, so if exist another continuous line which contain the same RGB value, it may detect such line as the axis instead. To prevent this, localisation is used by seeing the intensity of the colours in the high frequency colours. The index of RGB value of the high frequency colours are used in a shortlist to find the one which is located near where the x-axis or the y-axis chart labels are located.

C. Counting Pixels

An image of comprised of various pixels with different RGB values and is perceived as a whole picture. This image is broken into pixels for the purposes of calculation. Our model creates an index of every pixel in the image and shortlist the pixels which match the set rules based

Fig. 5. An example of same colour with different RGB values, (192,192,192) on the left, (192,192,191) on the right.

on localisation and threshold by each row and column. This pixel is than moved to another index, and given a value. This value is set arbitrarily as 1 in order to match the threshold value that would latter be used to follow the the rules and conditions to the next phase of the model. This becomes a mask which the model will once again count of pixel by row and columns.

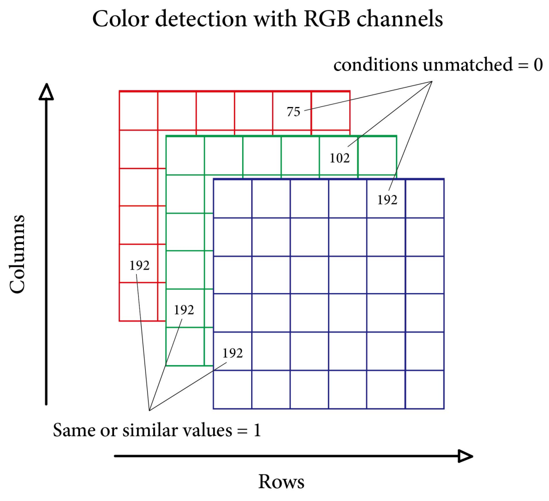

Fig. 6. Explanation of axis color detection method, same or similar values of each R, G and B are detected, if the values are match with the conditions, value 1 will be assigned at the pixel position where the R, G and B detected. If the values unmatched, value 0 will be assigned. The assigned value 0, 1 is an indicator of pixel that has the possibility of being the colour which makes up the axis.

D. Detection

The mask in the previous counting pixel phase leaves a result with pixels which are valued at ”1”. This value indicates the pixels which are the axes in a graph. Based on the rules and conditions set in the previous phases containing the summation of all values on each row and column, we can clarify that into axis by masking over once again a colour to visually signal where the axes are by drawing two vertical and horizontal lines. Figure 5 portrays the vertical and horizontal lines are the two axes which now have been successfully detected.

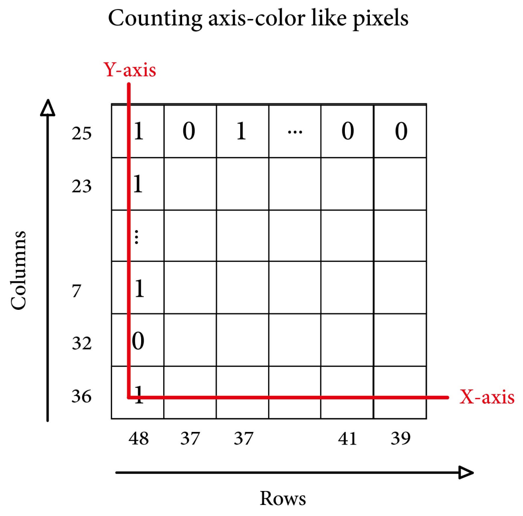

Fig. 7. Model calculates all the 1, 0 values in each row and column. The summation of values represents the number of pixels that has the possibility of being the colour which makes up the axis. After the sorting the sums, finding 10 max values, pick the far left, far down values. Draw lines at the point where the value assigned.

IV. Evaluation and Results

Our data set for evaluation was compiled in the following way. We searched for ”line-graphs” and ”bar-graphs” on Google to obtain 700 figures. The figures were manually inspected to contain the correct data set, having a clear explicit x-axis and y-axis and being the correct chart type. We call this summation of 1400 graphs our full data set. From these 1400 figures, we eliminated ones that does not fit our model, such as the one with implicit axis. This gives the *constrained dataset* of 634 line-graphs and 597 bar-graphs.

A. Evaluating Axes Detection

Table I summarizes the accuracies for *full* dataset. We tested detection of x-axis and y-axis on the two different type of graphs within parameters of accuracy and speed.

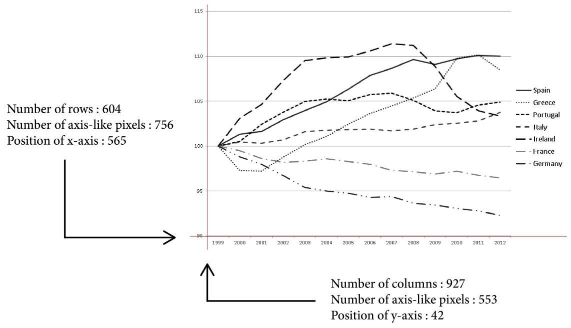

Fig. 8. An example result of axis detection

The accuracy pertains to how the model was able to accuracy detect the axes. The accuracy is measured by the average accuracy of the each figure. This is derived from the time the model took, divided by the number of figures which had their axes successfully extracted. The result are as follows:

TABLE I
ACCURACY AND AVERAGE TIME FOR BAR-CHART AND LINE-CHART AXES DETECTION

Dataset	**Line Chart**		**Bar Chart**	
	Accuracy	*Avg.Time* [a]	*Accuracy*	*Avg.Time* [a]
Full	78.15	0.15	75.29	0.18
Constrained	91.37	0.13	88.43	0.21

B. Comparison with ChartReader

We have compared ChartReader's methods proposed by Rane (2021)[1]with AxesDetector with the dataset provided by Chartreader. There are two dataset given by ChartReader were used to compare the two models. The constrained dataset were chosen as they represent the highest accuracy and the right condition for the model to perform. The constrained dataset from ChartReader are ones excluding bar-plots containing grey-scale, patterned or various other graphs totaling of 516 graphs. The comparsion between the two constrained datasets of bar-graphs are shown in Table II.

TABLE II
ACCURACY AND AVERAGE TIME FOR CHARTREADER AND AXIS DETECTOR WITH BAR CHART DATASET FROM CHARTREADER

Dataset	**AxisDetector**		**Chartreader**	
	Accuracy	*Avg.Time* [a]	*Accuracy*	*Avg.Time* [a]
Constrained	93.59	0.11	71.83	2.3

The results for the axis detection and the data associated with ChartReader only for the axes detection step, not the other steps in the process for the aforementioned paper. The metrics for measuring time and accuracy are the same as previous parameters. AxesDetector scales higher for all metrics for time and average time. This increased accuracy is based on the ability of AxesDetector to ascertain axes based on the pixel colour; not for continuous line such as ChartReader. Our model parses deploys localisation instead of arbitrarily setting a threshold to consider the difference RGB values that may exist. This indicates the robustness of our model used on other datasets that has not been conditioned.

V. LIMITATIONS AND SCOPE FOR IMPROVEMENT

As the nature of our model is to detect axis based on the colour intensity by counting the pixels in the figure, it fails to detect an axis when it is implied; as in not drawn. Figure 6. displays a chart where the y-axis is not drawn explicitly with a solid colour line, merely implied next to the y-axis values. The constrained the full demonstrates how the accuracy is dependent on the factor. Other inaccuracies arise in two occasions:

1) Colour of the axis lines
2) Colour of the background
3) Frequency of same-coloured pixels near the axis line.

This occurs due to the models' rule-based process. Our axis detector detects black-grey range colour in a way which RGB values are similar. Hence, if there exists similar coloured pixels near the axes which needs to be detected, the accuracy of the detection declines. As the axis detection depends on the colour of the line itself, the frequency of high-value columns will affect the axis detection. This includes instances where the chart has a coloured background. The model cannot detect RGB intensity colours when the background also has a high frequency of same pixel by being the same RGB values. Our model was constructed to detect the far-left column and far-down row in which each RGB values are same or similar. So if there are labels of the axis closely written to the axes themselves, or there are legends which are touching the axis, it may interrupt the process of axis detection.

VI. CONCLUSION

We created a framework which implements a set of algorithm from made models to automatically parse and detect axes in bar-graphs and line-graphs using OpenCV and matrices. The overall performance of this model has been improved with greater efficiency. This efficiency

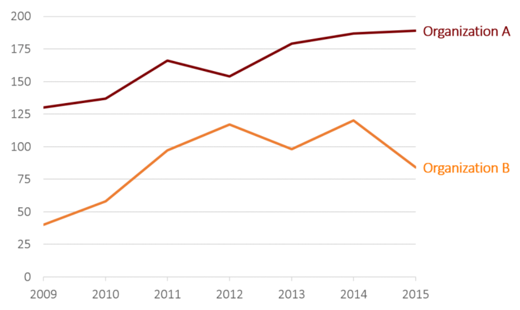

Fig. 9. A line chart where the y-axis is implied and not explicit.

arises from the lack of neural networks nor deep learning as the previous models of such area suggested. This lack of machine learning is crucial to the efficiency of the algorithm. The increase interest in convolutional computation has been one of the driving forces for scientific advances, yet it is not always needed to solve problems. Our axis detector stands as a testament of such fact; able to achieve its purpose in the most optimized manner.

We have addressed the limitations in our model, from the constrained and full dataset. However, even not matching all the conditions needed for the upper boundary execution, our axis detector, in general, performs higher than current models in the community in terms of both speed and accuracy. Hence, our axis detector provides a strong foundation for automatically detecting axes from figures.

Detecting axes in important for plot digitisation as it is the foundation of the process. The final steps of plot digitisation include scalarization. To extract information from charts, one needs to scalar the values given from the chart to present them as value formats in a digital format. This would translate the written values in a chart to a plot-able value.We hope to aid in the parsing of figures in scientific journals or any research articles to help the understanding the the development of models.

Acknowledgment

The authors would like to express our special thanks of gratitude to the MedTech Entertake who gave us an opportunity to perform this research project

References

[1] C. Rane, S. M. Subramanya, D. S. Endluri, J. Wu, and C. L. Giles, “Chartreader: Automatic parsing of bar-plots,” 2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI), 2021.

[2] C. Sohn, H. Choi, K. Kim, J. Park, and J. Noh, “Line chart understanding with Convolutional Neural Network,” Electronics, vol. 10, no. 6, p. 749, 2021.

[3] D. Jung, W. Kim, H. Song, J.-in Hwang, B. Lee, B. Kim, and J. Seo, “Chartsense,” Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, 2017.

[4] H. Li, Y. Wang, A. Wu, H. Wei, and H. Qu, “Structure-aware visualization retrieval,” CHI Conference on Human Factors in Computing Systems, 2022.

[5] J. Luo, Z. Li, J. Wang, and C.-Y. Lin, “CHARTOCR: Data Extraction from charts images via a deep hybrid framework,” 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), 2021.

[6] K. Davila, B. U. Kota, S. Setlur, V. Govindaraju, C. Tensmeyer, S. Shekhar, and R. Chaudhry, “ICDAR 2019 competition on harvesting raw tables from infographics (chart-infographics),” 2019 International Conference on Document Analysis and Recognition (ICDAR), 2019.

[7] L. Zhang, G. Wang, and L. Chen, “Chartmaster: An end-to-end method to recognize, redraw, and redesign chart,” Discrete Dynamics in Nature and Society, vol. 2021, pp. 1–14, 2021.

[8] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, “Revision,” Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST ’11, 2011.

[9] Siegel, N., Lourie, N., Power, R., Ammar, W.: Extracting scientific figures with distantly supervised neural networks. In: Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, JCDL 2018, Fort Worth, TX, USA, June 03-07, 2018. pp. 223–232. ACM (2018).https://doi.org/10.1145/3197026.3197040

[10] R. A. Al-Zaidy and C. L. Giles, “Automatic extraction of data from Bar Charts,” Proceedings of the 8th International Conference on Knowledge Capture, 2015.

[11] Rohatgi, A.: Webplotdigitizer: Version 4.4 (2020)

[12] S. Elzer, S. Carberry, and I. Zukerman, “The automated understanding of Simple Bar Charts,” Artificial Intelligence, vol. 175, no. 2, pp. 526–555, 2011.

[13] W. Huang, C. L. Tan, and W. K. Leow, “Model-based chart image recognition,” Graphics Recognition. Recent Advances and Perspectives, pp. 87–99, 2004.

[14] Z. Ahmed, S. Zeeshan, and T. Dandekar, “Mining biomedical images towards valuable information retrieval in biomedical and Life Sciences,” Database, vol. 2016, 2016.