ChartReader: Automatic Parsing of Bar-Plots

Chinmayee Rane
Computer Science and Engineering
University at Buffalo
Buffalo, New York
cvrane@buffalo.edu

Seshasayee Mahadevan Subramanya
Chemical Engineering Department
Pennsylvania State University
University Park, Pennsylvania
sesha@psu.edu

Devi Sandeep Endluri
Computer Science and Engineering
Texas A&M University
College Station, Texas
dsandeep97@tamu.edu

Jian Wu
Department of Computer Science
Old Dominion University
Norfolk, Virginia
jwu@cs.odu.edu

C. Lee Giles
College of Information Sciences and Technology
Pennsylvania State University
University Park, Pennsylvania
clg20@psu.edu

***Abstract*—Scientific figures such as bar graphs are a critical part of scientific research and a predominant method used to represent trends and relationships in data. However, manually interpreting and extracting information from graphs is often tedious. Since data consumption has exponentially evolved over the past few decades, there is a need for automated data inference from these bar graphs. ChartReader presents a fully automated end-to-end framework that extracts data from bar graphs in scientific research papers focusing on process engineering and environmental science journals. ChartReader uses a deep learning-based classifier to determine the chart type of a given chart image. We then develop novel heuristic methods for analyzing scientific figures (text detection, pixel grouping, object detection) and address prime challenges like axis detection, legend parsing, and label detection. Our framework achieves 98% and 68% accuracy in parsing x-axis and y-axis ticks, respectively. It achieves 83% accuracy in parsing legends and 42% accuracy in parsing data values in the testing corpus. We compare the proposed method with state-of-the-art methods and address its limitations.**

***Index Terms*—Data Extraction, Chart Classification, Text Detection, Computer Vision**

I. Introduction

Bar-plots are prevalent in papers written in almost all scientific domains [6]. Authors use bar-plots for all kinds of purposes, such as illustrating trends, correlations, distributions, contrastive characteristics, and abnormalities. Bar-plots are widely used because they provide a straightforward visualization of data. However, the data used to make the plots are not always available. Such data points can be valuable for many general purposes, such as result verification and baseline comparison. In certain domains such as Chemical Engineering, automatically and precisely reading data from bar-plots and associating them with corresponding semantic information (e.g., x-axis labels and legends) would enable the opportunity to develop kinetic and statistical models to predict product formations. Manually reading exact values from bar-plots is tedious and usually inaccurate. There is thus, an increasing interest in automating image mining and parsing [1], [10] to develop algorithms that can extract numeric and text information from bar-plots with the purpose of parsing values of data points and associating them with corresponding labels and legends. Furthermore, existing academic search engines such as Google Scholar or CiteSeerX focus on text-based mining and retrieval and do not provide interfaces yet to search figures in scientific papers. Such functionality can be developed based on extracting and parsing bar-plots and other types of figures.

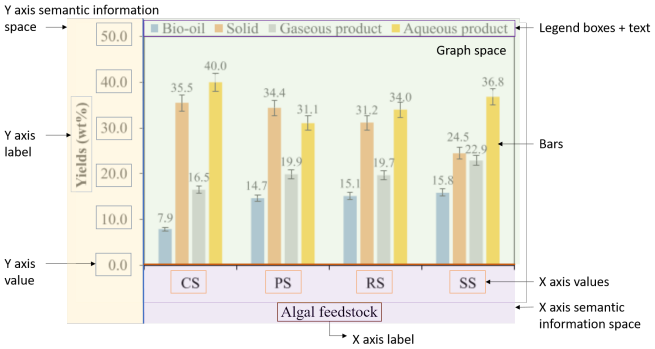


Fig. 1: A typical vertically oriented bar-plot with horizontally oriented legends. Features in this plot are labelled and highlighted (green: graph region, orange: y-axis semantic information region, purple: x-axis semantic information region).

Most bar-plots have common structures, which we can utilize for classification and parsing of numeric and text information. Fig. 1 gives a sample of a typical bar-plot that follows specific rules. For example, legends are colored boxes with text on the right side. These colors are also present in boxes and "bars" that make up the plot. The length of these bars signifies the values while their positions and colors give semantic information.

A typical bar-plot (Fig. 1) can be segmented into 3 regions: a graph region, an x-axis semantic information region, and a y-axis semantic information region. The axes are vertically or horizontally aligned straight lines that bifurcate the graph space from the semantic information regions. Each x- or y-axis semantic information region contains respective labels and tick

values. The graph region contains bars and legends.

As for now, the common practice to parse such plots is manual tools such as webplotdigitizer [11]. When using this tool, a user uploads an image file of the plot, labels the x- and y-axis positions and the bars to obtain value for each bar. This method is slow and thus does not scale to millions of bar-plots extracted from a large volume of journal articles. Another key shortcoming is that it does not parse semantic information and associate it to specific values of bars. Our approach overcomes these shortcomings by automatically extracting bar-plots with captions from the journal article, parsing the values of bars and associated semantic information. Therefore, it can be used for processing bar-plots in scientific articles on a large scale.

The novel contributions of our algorithm are as follows. The software is open source available on GitHub[1].

1) A novel double-pass algorithm integrated with an optical character recognition (OCR) engine improves text detection by noise reduction.
2) An algorithm to parse legends arranged horizontally, vertically, or in a matrix with different colors, pixel values, and positions in the plot.
3) A sweeping-line based algorithm for axis detection.
4) Associating bars with semantic information based on color (legends) and positioning (x-axis value).

II. Related Works

One of the challenges in accurately parsing bar-plots is to handle various types of bar-plots in scientific literature. Previous works focused on building heuristic models that identify key features, such as bars, x-axis semantics, and y-axis semantics, e.g., [17], [18].

For example, Zhou et al. (2001) [16] used ergodic hidden Markov models to identify bar-plots. They used feature extraction to estimate parameters for the hidden Markov model. Davila et al. (2019) [5] used PixelLink text detection and SVM training to match feature labels with their respective feature components. However, both the studies did not extract data values from bar-plots.

Savva et al. (2011) [13] and Al-Zaidy et al. (2015) [2] used a bilateral filter to smoothen small color variations in bar-plots and connected components labelling with lab color space to distinguish components. The bounding-boxes with different RGB values were detected and labelled as unique vectors. The accuracy of detecting bar values was improved from 67% [13] to 80% [2] using the technique above. Arouja et al. (2020) [3] recently developed a model that detects bar-plots from scanned books. However, they focused on improving the classification algorithm but not on parsing numerical information from bar-plots.

Seigel et al (2016) [15] used a convolutional neural network (CNN) to parse legends and detect labels and ticks for all types of graphs (line, bar, scatter etc.). The overall accuracy was low (17.6%), but axes (positions and scales) and legends (labels and symbols) were detected with relatively high accuracies ($> 90\%$ for axis and $> 70\%$ for legends).

Ying et al. (2017) [7] developed a model that parses legends in bar-plots. They related legends and sub-bars (clustered bars with the same x-value) based on positions. The bars and legends were assumed to follow the same order from left to right. The figures were collected from PubMed papers and converted to grayscale. The algorithm detected 32.8% of values from bar-plots. The proposed algorithm failed to work on figures with low resolution or small bars. Legends that were aligned vertically or in a matrix were not detected.

Abhijit et al. (2018) [14] developed a model to improve the accuracy to parse bar values using synthetic data generated from the matplotlib library. The plots were first converted to grayscale, the pixel values of bars were matched with corresponding legends. However, all legends in the synthetic data were below the x-axis and away from the bars, which was incongruous in general because the legend boxes could appear in other places. The model achieved an accuracy ranging from 70% for stacked plots to 76% for simple bar-plots (without legends).

Most of the previous methods do not parse legends. Some made assumptions that legends were always below the plots [14], or ordered horizontally along the same line [7]. This limits on the applicability of these models. Legends contain important semantic information because they enable 2D bar-plots to act as 3D plots with an additional varying parameter. We take this as one of our cornerstone challenges. The previous algorithms were mostly developed for grayscale plots. They did not utilize the color information to parse legends. Also there has been less focus on measuring the accuracy of detecting the axes or label values. Quantifying the accuracy of obtaining these semantic information is important for understanding where the capping limits are in this evaluation process.

III. Methods

A. Overview

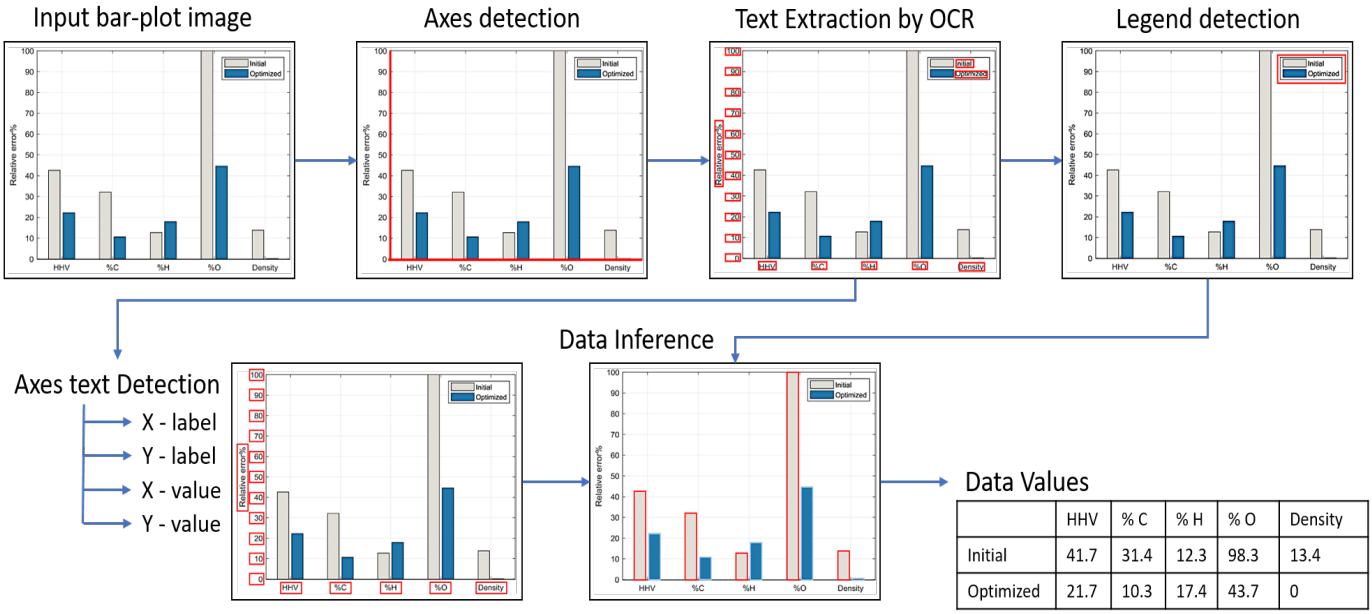
Our parsing approach (see Figure 2) starts with axis and label detection. We use the maximum threshold method to distinguish the x- and y-axis from the horizontal and vertical lines within the images. The OCR Engine AWS Rekognition[2] is applied on the images to detect the text and its locations. The location information is used to compute bounding-boxes of the detected text. A bounding-box (or a contour) is a simple rectangle surrounding the text detected by the OCR. The axes detected are used to filter x-labels, y-labels, and legends from all of the text detected. Finally, we obtain the y-values by converting pixel values of y-labels and their corresponding ticks.

B. Figure Extraction

We used PDFFigures2 [4] to extract figures from research papers. PDFFigures2 resolves peculiar challenges like handling documents with widely differing spacing conventions,

[1]https://github.com/Cvrane/ChartReader.git

[2]https://aws.amazon.com/rekognition/



	HHV	% C	% H	% O	Density
Initial	41.7	31.4	12.3	98.3	13.4
Optimized	21.7	10.3	17.4	43.7	0

Fig. 2: An overview of ChartReader model.

avoiding false positives while maintaining the ability to extract a broad range of possible captions, and extracting a highly varied selection of figures and tables. We also extracted captions that provided brief descriptions of figures. All metadata and captions were saved into a database for convenient access.

C. Figure Classification

1) Chart Image Dataset Construction: To identify bar-plots from a large collection of figures, we trained a model to classify figures extracted above. We followed a similar approach as ChartSense [8] to prepare the training dataset. We developed a script using the python module google_images_download[3] to download chart images (in a similar way as ReVision [13]). Then, we manually identified and removed all the incorrect samples from the downloaded figures. We obtained a total of 4320 figures in 13 categories, including 528 bar-plots

2) Training: We trained a multiclass classification model using the chart image dataset and evaluated it with a five-fold cross-validation.

We used Keras[4] to implement the transfer learning model. These models were pre-trained on the ImageNet dataset [12]. We loaded these pre-trained models and fine-tuned them on the chart classification dataset. We froze all the layers except the last convolutional layer. The fully-connected layer used a softmax function to classify figures into 13 chart categories. The convolutional layer and the added fully-connected layer were retrained for 30 epochs using Adadelta as an optimizer.

A dropout layer with a rate 0.3 is included before the final fully-connected layer to prevent overfitting. All figures were rescaled to a fixed size of $224 \times 224 \times 3$ to meet the input image dimension requirement of VGG-19. Figure 3 shows the model architecture. In this model, only the fully connected layers and the last convolutional layer block (conv5) are trained.

TABLE I: Validation accuracy (averaged over 5 folds) of CNN models for classifying bar-plots from the Chart Image Dataset.

Model	Training parameters	Accuracy (%)
VGG-19	47M	82.11 ($\pm$ 1.37)
EfficientNetB3	107M	84.53 ($\pm$ 0.92)
Inception-V3	91M	84.53 ($\pm$ 1.61)
ResNet-152V2	143M	83.54 ($\pm$ 1.19)

3) Results: We compared four CNN-based neural networks (CNNs), namely VGG-19, ResNet-152V2, Inception-V3 and EfficientNetB3. The results are shown in Table I. VGG-19 has an average validation accuracy of 82.11%. The other three baselines achieved a similar accuracy, but the number of parameters of VGG-19 model was significantly smaller than that of the other three models we have experimented with. One phenomenon we observed in error analysis was that these CNN based models tend to produce the same false positives, indicating that alternative types of neural networks may be needed to achieve a better performance.

D. Axes Detection

The first step of the axis detection algorithm is to convert input figure into binary figure, by converting into gray-scale image, then replacing all pixels in the resulting figure with luminance greater than 200 with the value 1 (white) and replacing all other pixels with the value 0 (black). We experimented with different values for binary thresholding and observed that a threshold of 200 gave the best results. After this, we scanned the matrix vertically and traced the continuity of black pixels within adjacent columns. To identify the true y-axis from these candidates, we used a threshold value of 10

[3]https://pypi.org/project/google_images_download/
[4]https://github.com/fchollet/keras

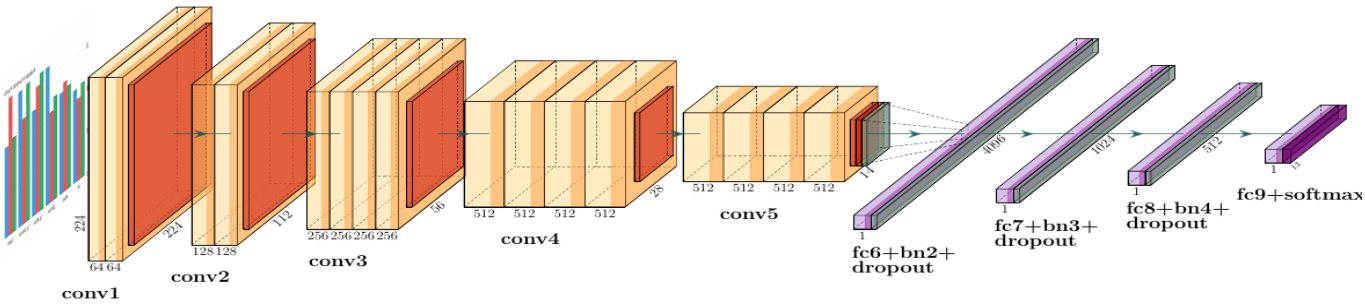


Fig. 3: Model architecture used for figure classification. We use conv to denote convolutional layer, fc to denote fully connected layer, bn to denote batch normalization Layer.

for continuity of black pixels. Finally, the first column that has the highest number of continuous black pixels was identified as the y-axis.

Similarly, we detected the x-axis by horizontally tracing the number of continuous black pixels in each row of the matrix converted from the original figure. The x-axis was identified as the last row containing the maximum number of continuous black pixels above a threshold of 10.

E. Text detection

The AWS-Rekognition DetectText API was used to detect text in a figure. DetectText also provided rectangular bounding boxes of the detected text. We employed a double-pass algorithm, illustrated in Fig. 4, to improve the text detection results. The pseudo-code is shown in Algorithm 1. AWS Rekognition outputs a confidence score for the detected text indicating the probability that the prediction is correct. Our algorithm was motivated by the intuition that whitening out detected text with high confidence scores may reduce their interference and thus improve the quality and confidence scores of the other text that had relatively lower confidence scores in the first pass. We examined the text detection results on a set of randomly selected figures and found that most of the correctly recognized text yielded confidence scores ranging from 85% to 90%. Therefore, we empirically set the threshold of the confidence score to be 80%.

Algorithm 1 Double-pass Algorithm

1: Run *AWS Rekognition* on the figure to detect text and obtain corresponding locations of the text
2: Select the detected text with *Confidence* $>= 80$
3: Compute the bounding-boxes using location output for the text obtained after Step 02
4: Fill the bounding-boxes corresponding to the selected text with white color
5: Repeat Steps 01, 02 and 03 (2^{nd} pass)

F. Label Identification

A figure may contain different types of labels, so we develop an algorithm to first classify all detected labels into the following categories (see Fig. 1 for an illustration):

- *x-labels:* the text enclosed by the bounding-boxes below the x-axis.
- *x-text:* the text enclosed by the bounding-boxes below the x-labels.
- *y-labels:* the numerical values enclosed by the bounding-boxes detected to the left of the y-axis.
- *y-text:* the text enclosed by bounding-boxes detected to the left of the y-labels.
- *legend labels:* the text enclosed by the bounding-boxes detected to the right of the colored boxes in the legends.

We apply a sweeping-line based method to classify labels obtained from text detection approach described in section

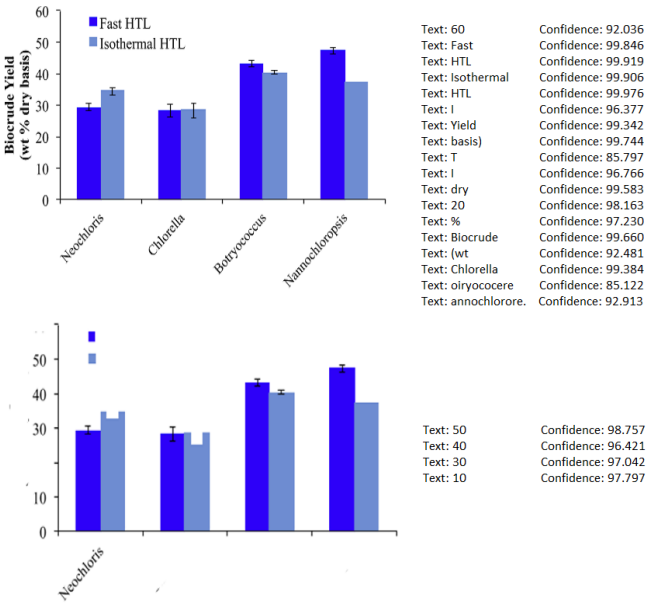


Fig. 4: Double Pass Algorithm output on a sample figure: The first iteration detects the text. Bounding-boxes corresponding to the text with confidence $\geq 80\%$ are selected. The second iteration whitens the selected bounding-boxes and re-runs the text detection algorithm.

III-E. In this method, a horizontal line sweeps from the top to the bottom. This line will intersect bounding boxes of labels. The x-labels can be determined using the number of intersections with the bounding boxes along the line. A similar algorithm can be used for detecting y-labels and y-text using a vertical sweeping line.

G. Legend detection

The legend detection is based on results of label detection. A legend typically includes a set of label-color pairs. In most bar-plots, the legends appear above the x-axis and to the right of the y-axis. We first remove two types of text boxes from this region. The first type of text boxes are results of AWS Rekognition, which usually interprets error bars as the letters "I"(s), so we remove bounding boxes containing only a single "I" character. The second type is text boxes containing numerical values above bars. The remaining text boxes contain legend color boxes and legend names. Because the legend names may contain multiple words, we merge the bounding boxes with distances less than 10 into a single legend name. The distance between the bounding boxes is measured as the difference between the leftmost x-value of the first bounding box and the rightmost x-value of the second bounding box. These bounding boxes were grouped in such a way that each member of the group is either horizontally or vertically aligned to at least one other member in the group. The largest group gave us the legends. The method is described in Algorithm 2.

Algorithm 2 Legend Detection Algorithm

```
legend_groups ← empty list
for each bounding-box in the figure do
    for each group in legend_groups do
        if bounding-box is either horizontally or vertically aligned to any member in the group then
            Add bounding-box to the group
        end if
    end for
    if bounding-box couldn't become member of any of the group then
        Create a new group with this bounding-box
        Add this new group to legend_groups
    end if
    Calculate the maximum length group of bounding-boxes to determine the legends
end for
```

H. Color Estimation

The color estimation module combines the double-pass and label detection methods. Before we match colors in the legend to the colors in the bars, we associate the text description of each bounding box to its corresponding legend. The text bounding-box of a particular legend is determined to be the nearest bounding-box with a similar height on its left.

We use the original figures to identify the color corresponding to a legend text label. Here, we assume the color boxes are located on the left side of legend labels. Ideally, pixels within a box should have exactly the same pixel values. However, due to many reasons (e.g., image compression, scanning, etc.) the pixel values may vary. Similar to Algorithm 2, we start with a new group with a random pixel and incrementally add pixels whose R, G, and B values are no more than 5 compared with the average of all pixels in the group. The average of all pixels in R, G, and B channels in the largest group is used as the color of a legend label. These colors are later used for identifying bars matching a particular legend.

I. Data extraction

Using the color estimation method, we obtain a simplified bar-plot for each legend that can be used to detect the bars corresponding to a particular legend. To estimate each bar's height, we need a mapping function that converts pixel values to actual values using a factor called the *value-tick ratio* (α).

As the method to detect numeric values is not 100% accurate, the pixel values that deviate significantly from a normal distribution of the ticks or the labels are removed assuming that data points with random errors should follow a normal distribution. After removing the outliers, we calculate the average distance between ticks (Δd) in pixels and the average of the actual y-label ticks ($\overline{N}_{\text{tick}}$). The value-tick ratio is calculated by:

$$\alpha = \frac{\overline{N}_{\text{tick}}}{\Delta d}$$

Y-value estimation: Using color groups, and the value-tick ratio, the y-values in a bar-plot can be extracted. The y-values for each bounding-box of the bar-plot using Equation III-I as $\text{y value} = \alpha \times H$, where H is the height of a bar.

IV. Evaluation and Results

We compiled our dataset in the following way. We searched the keyword "hydrothermal liquefaction" on the Web of Science search interface to obtain a list of DOIs. Using the DOIs, we downloaded 1857 PDF papers. The papers downloaded were published from 1990 to 2020 to cover hand-drawn and machine-drawn figures. We used PDFFigures2 [4] to extract figures and corresponding captions and classified them to obtain bar-plots. We further selected 1240 bar-plots from 654 PDFs that contain keywords "oil yield" and "GC-MS" in figure captions. These bar-plots contain data of interest for the users. The ground truth was obtained in a semi-automatic manner by visually inspecting figures with the aid of *webplotdigitizer* [11]. We call the 1240 bar-plots the *full dataset*. Out of the full dataset, we excluded bar-plots containing grayscale or patterned bars and horizontal bar-plots and constructed another dataset called the *constrained dataset* containing 516 bar-plots.

A. Evaluating Bar Value Extraction

Table II summarizes the accuracies for entire dataset and constrained dataset for each of the parameters evaluated. We define accuracy as the percentage of data points from the dataset of interest that are correctly detected. In the next

TABLE II: Metrics evaluating successful determination of data values and semantic features for the entire and constrained in-house dataset.

Parameter	Accuracy(%)	
	Full dataset	Constrained dataset
Legends	83.28	-
X-axis ticks	97.97	-
Y-axis ticks	68.78	-
Height/value ratio	88.63	-
Y-axis label	77.58	-
X-axis label	71.29	-
Data association	65.42	75.37
Data values	22.98	42.10

section, we discuss the reason behind these limitations and the algorithms needed to overcome them.

There are no constraints for evaluating the legend, label, y-values and tick determinations. Therefore, we do not calculate the accuracies for the constrained dataset for these parameters. The limitations mentioned above are typical to the bar detection and do not impact parsing semantic information. Also, note that all the values are determined on an absolute basis. For example, for legends, it would be the number of correct legends detected divided by the total number of legends in all plots. This allows us to estimate the number of data points that can be parsed from a given dataset.

TABLE III: The criteria to determine if various parameters are correctly extracted.

Parameter	Detection protocol
Legends	Colour of legend and corresponding text are correctly matched
Text in axis	Axis values for X (all x-axis text) and Y (at least 5 numeric values) axis are correctly evaluated
Y-axis value	Height to value ratio is accurately evaluated
Labels in axis	x- and y-axis labels are detected
Data association	x-axis and legend values are matched with corresponding bars
Data values	Value of the bar point is accurately determined within ± 5 % deviation

B. Legend detection

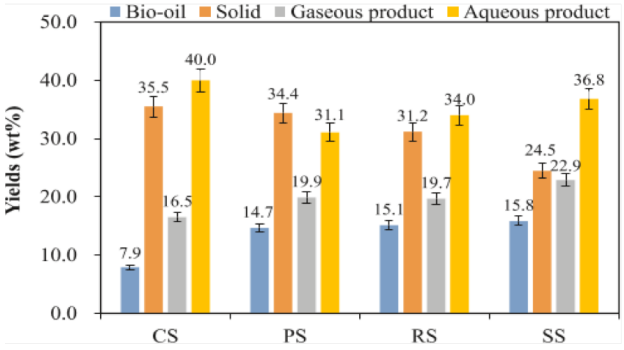
Our algorithm achieves an accuracy of 83.28% with 3615 out of 4341 detected correctly. This caps the maximum accuracy that the whole pipeline can achieve. Because the legends are parsed by scanning the graph space (see Fig. 1), the algorithm does not work well in cases where legends appear below the x-axis. There were 39 out of 1240 such plots. In other failed cases, the legend color boxes were not recognized because their sizes were too small and were treated as the background noise.

C. Axis tick and label detection

The x-axis tick detection achieves an accuracy of 97.97% (7879 out of 8042). This is attributed to the robust axis determination algorithm. However, the y-axis tick detection has a lower accuracy (68.78%). Y-axis ticks are numerical values in most cases. However, a fraction of plots have a "%" symbol accompanying an alphanumeric value (see Fig. 6(b)). The OCR engine used does not distinguish the numeric value and symbols well, which results in a lower accuracy. However, the accuracy for height/value ratio determination is relatively high (88.63%). This is because as long as two y-axis tick values are correctly detected in a given plot, the above ratio can be correctly determined. To improve the accuracy of the ratio, we calculate multiple ratio values and take the average when more than two data points are available. The x-axis and y-axis labels are determined with accuracies of 71.29% and 77.58%, respectively. Note that a small fraction of plots did not have an x- or y-axis, and are taken as false positives.

D. Data evaluation and association

To build the ground truth, we manually extracted 18,830 data points from 1,240 bar-plot figures in the full dataset. Here a data point is defined as an individual bar in a bar-plot. Each data point has a legend and x-axis tick associated and a y-value that is to be determined. As seen in Table III, there are two parameters we need to evaluate. The "data association" parameter measures whether a data point correctly matches its corresponding legend and its x-axis value. The "data value" parameter measures whether the bar value is calculated within



doi	doi:10.1016/j.fuel.2019.116946					
file name	1-s2.0-S0016236119323397-main-Figure1-1.png					
x-text						
x-labels	CS	PS	RS	SS		
y-text	Yields	(wt%o)				
y-labels	50	40	30	20	10	0
legends	Bio-oil	Solid	Gaseous pr	Aqueous product		
		CS	PS	RS	SS	
Bio-oil		7.8	14.7	15.1	15.9	
Solid		35.3	34.3	31	24.3	
Gaseous product		16.5	19.8	19.6	22.7	
Aqueous product		39.8	31	33.9	36.7	

Fig. 5: Data values extracted from a figure. The x-text, x-labels, y-text and y-labels inferred are also shown in the right table. The example here does not have an x-text.

an offset of $\pm 5\%$ with respect to the true value. Data parsed from a bar-plot are stored in a spreadsheet. An example is shown in Fig. 5.

Fig. 2 shows that successfully detecting bar values and finding their corresponding semantic information relies on the detection of legends, x-axis ticks, and height/value ratio. Therefore, the best accuracy we can achieve for the bar value detection is the multiplication of the accuracies of these three tasks, which is about 72.3%. Improving legend and axis tick detection will push this limit higher.

The data association accuracy reported in Table II is 65.4%, which is close to the upper bound accuracy estimated above (72.3%), indicating that the proposed algorithm for associating data worked well. On the other hand, the accuracy of reading data values is relatively low (22.98%).

We performed an error analysis to find that certain types of bar-plots are not correctly parsed. These include gray-scale figures (see Fig. 6(a)), figures with line grids, figures with horizontal bars, figures with legends below x-axis, figures with legends outside the plot region, or figures with colored patterns in the bars. Excluding these cases increases the accuracy to 42.10%.

E. Testing with open-source dataset

We test the accuracy of this framework to open-source and generic datasets. This helps in reducing the biases that may have incurred during the model evaluation with the in-house dataset. It also explores the validity of this framework to charts from generic fields. We chose 2 datasets - IEEE Luo 2021 [9] and ICDAR 2019 [5]. We processed all 386,966, 198,010 charts in each of these datasets and randomly chose 100 for evaluation (same methodology as Section IV). The results are summarized in table IV.

The accuracies for ICDAR and in-house datasets are similiar. The framework has lower accuracies for the IEEE dataset. Most of the charts in the IEEE dataset were of the same template with changes in bar values. Legend parsing failed on 62 charts because legends are below the axis. Despite 88 of the 100 plots in the ICDAR dataset containing patterned legends, the accuracies are still relatively high. This indicates the robustness of the model to work for colored patterned legends. Note that none of these datasets contain grayscale charts. Hence for chart-parsing frameworks, extracting information from varied sources is required to minimize evaluation biases.

F. Comparison with Al-Zaidy et al. [2]

We compared ChartReader with the method proposed by Al-Zaidy et al. (2015) on the same dataset. The figure dataset used by [2] contains 2257 bar-plots. We ran the ChartReader on all bar-plots and randomly selected 100 for evaluation (same methodology as Section IV). The comparison results are shown in Table IV.

The results for parameters such as legends, axis ticks, height/value ratio, and data association are similar to ChartReader (Table II). However, the metrics for successful determination of data values and axis labels are higher than ChartReader. The increase of accuracy for axis label detection is due to the presence of a larger number of special characters such as "%" and "°" in our dataset. Furthermore, the figure dataset used for Table IV includes a larger fraction of ATLAS charts that resembles bar-plots shown in Fig. 1. This increases the accuracies for data value detection. The dataset used by [2] also contains gray-scale figures, figures with patterns, and horizontal bars and they were extracted from multiple domains, so the consistency of results in Table IV to Table II indicates the robustness of our model on processing bar-plots extracted from journals beyond chemical engineering.

The model proposed by [2] did not parse legends and obtained bar values for approximately 67% of all bar-plots (Column 3 in Fig. IV). Our bar value detection accuracy is lower (49%) due to addition of legend parsing. Overall, compared with [2], ChartReader achieved a comparable performance in most parameters measured. It added the functionality to parse legends and significantly improved y-axis label detection.

V. Limitations and scope for improvement

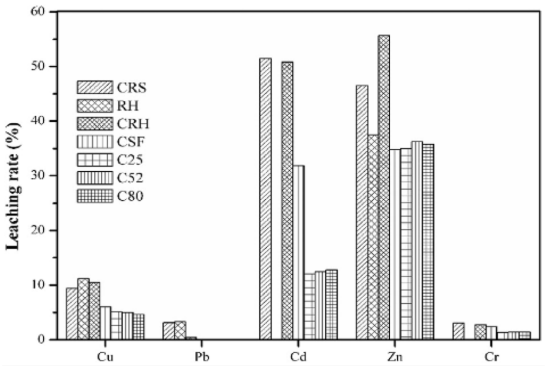
As shown in Section IV, although ChartReader achieved a relatively high performance in the *constrained dataset*, it still does not work well on certain types of bar-plots. The limitation of the framework and the future works for improvement are proposed here.

Axes Detection. Our algorithm failed to detect axes when there were no solid lines representing the y-axis. In this case, the y-axis can be detected by identify bounding boxes along a vertical line in the bar-plot. However, if there are symbols accompanying the numeric values, a module is needed to correctly interpret these symbols (Fig. 6(b)). Another case when axis detection may fail is when the x-axis is at the top of the plot. This case can be handled by using a bidirectional sweeping line with heuristic rules.

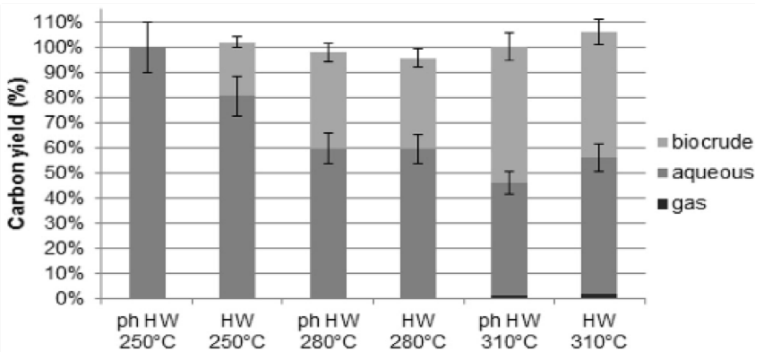
Detecting legends and parsing data values. We showed that ChartReader achieved reasonable performance for colored bar-plots. ChartReader failed to parse most gray-scale bar-plots because gray-scale plots predominantly have patterns based legends. Integration of pattern recognition techniques with our innate color segmentation methods is a promising approach to detect patterned legends. Grouping pixels in squares using a neural network model and analyzing patterns is a promising first step. Fig. 6(a) show an example of a gray-scale pattern bar-plot which failed to be parsed by ChartReader.

VI. Conclusion

We developed a framework implementing a set of algorithms that automatically parse numeric values and associated semantic information from bar-plots in journal papers and generate reconstructive datasheets using neural network (classification) and heuristic (bar-plot parsing) methods. The overall performance for parsing data from a general bar-plot has been improved compared with the state-of-the-art with real-world datasets. We addressed the limitations of the proposed algorithms and proposed possible solutions to overcome these challenges.



(a) The input image is a gray-scale bar-plot.



(b) The image where the "%" symbol accompanies the y-labels.

Fig. 6: Data extraction results that failed the ChartReader model.

TABLE IV: Accuracies (%) for successful determination of data values and semantic features for entire and constrained datasets from IEEE Luo 2021, ICDAR 2019 [5], Al-Zaidy et al. [2].

Parameter	Accuracy(%)						
	Entire ICDAR dataset	Constrained ICDAR dataset	Entire IEEE dataset	Constrained IEEE dataset	Entire Al-Zaidy dataset	Constrained Al-Zaidy dataset	Al-Zaidy evaluation on Al-Zaidy dataset
Legends	52.07	52.07	70.00	70.00	82.91	82.91	-
x-axis ticks	97.92	97.92	72.59	72.59	92.53	92.53	-
y-axis ticks	99.06	99.06	89.87	89.87	89.77	89.77	-
height/value ratio	99	99	98	98	89	89	-
y-axis label	98	98	97	97	98	98	20
x-axis label	96	96	94	94	98	98	-
Data association	58.27	45.45	42.68	44.87	61.95	77.93	-
Data values	27.19	45.45	16.67	18.44	49.31	76.55	67.00

ChartReader provides a stronger baseline for automatically reading data from scientific bar-plots at scale. Bar-plots are extracted and parsed on average in 8.3 secs on a personal computer. This aids in expediting mining of information from a large corpus of research articles to develop data-driven models.

REFERENCES

[1] Ahmed, Z., Saman, Z. and Thomas, D, "Mining biomedical images towards valuable information retrieval in biomedical and life sciences." Database (2016)

[2] Al-Zaidy, R.A., Giles, C.L.: Automatic extraction of data from bar charts. In: Barker, K., G oomez-Perez, J.M. (eds.) Proceedings of the 8th International Conference on Knowledge Capture, K-CAP 2015, Palisades, NY, USA, October 7-10, 2015. pp. 30:1–30:4. ACM (2015).

[3] Araujo, T., Chagas, P, A.J., Santos, C., Sousa Santos, B., Serique Meiguins, B.: A real-world approach on the problem of chart recognition using classification, detection and perspective correction 20, 4370.

[4] Clark, C., Divvala, S.: Pdffigures 2.0: Mining figures from research papers. In: 2016 IEEE/ACM JointConference on Digital Libraries (JCDL). pp. 143–152 (June 2016)

[5] Davila, K., Kota, B.U., Setlur, S., Govindaraju, V., Tensmeyer, C., Shekhar, S., Chaudhry, R.: Icdar 2019 competition on harvesting raw tables from infographics (chart-infographics). International Conference on Document Analysis and Recognition. pp. 1594–1599 (2019).

[6] Davila, K., Setlur, S., Doermann, D., Bhargava, U.K., Govindaraju, V.: Chart mining: A survey of methods for automated chart analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)

[7] He, Y., Yu, X., Gan, Y., Zhu, T., Xiong, S., Peng, J., Hu, L., Xu, G., Yuan, X.: Bar charts detection and analysis in biomedical literature of PubMed Central. AMIA ... Annual Symposium proceedings. AMIA Symposium2017, 859–865 (2017)

[8] Jung, D., Kim, W., Song, H., Hwang, J.i., Lee, B., Kim, B., Seo, J.: Chartsense: Interactive data extraction from chart images. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. p. 6706–6717. CHI '17, Association for Computing Machinery, New York, NY, USA (2017).

[9] Luo, Junyu, Zekun Li, Jinpeng Wang, and Chin-Yew Lin. "ChartOCR: Data Extraction From Charts Images via a Deep Hybrid Framework." In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 1917-1925. 2021.

[10] Nagthane, Deepika, K.: Image mining techniques and applications. International Journal Of Engineering Sciences Research Technology (2013)

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

[12] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A., Fei-Fei, L.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision115, 211–252 (2015)

[13] Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M., Heer, J.: ReVision: automated classification, analysis and redesign of chart images. In: Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11. p. 393. ACM Press, Santa Barbara, California, USA (2011).

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

[15] 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

[16] Yanping Zhou, Chew Lim Tan: Chart analysis and recognition in document images. In: Proceedings of Sixth International Conference on Document Analysis and Recognition. pp. 1055–1058 (Sep 2001)

[17] Yokokura, N., Watanabe, T.: Layout-based approach for extracting constructive elements of bar-charts.International Workshop on Graphics Recognition pp. 163–174 (1997)

[18] Zhou, Yan, P., Chew, L.T.: Hough technique for bar charts detection and recognition in document images. International Conference on Image Processing (Cat. No. 00CH37101)2, 605–608 (2000)