ChartOCR: Data Extraction from Charts Images via a Deep Hybrid Framework

Junyu Luo*[1], Zekun Li*[2], Jinpeng Wang[3], and Chin-Yew Lin[3]

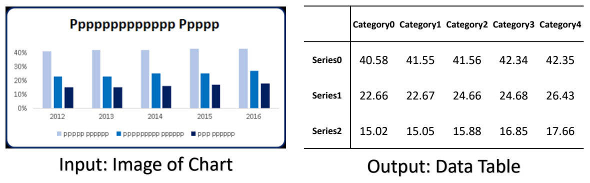
[1]Pennstate University, Pennsylvania, USA
[2]University of Southern California, Los Angeles, California, USA
[3]Microsoft Research, Beijing, China

Abstract

Chart images are commonly used for data visualization. Automatically reading the chart values is a key step for chart content understanding. Charts have a lot of variations in style (e.g. bar chart, line chart, pie chart and etc.), which makes pure rule-based data extraction methods difficult to handle. However, it is also improper to directly apply end-to-end deep learning solutions since these methods usually deal with specific types of charts. In this paper, we propose an unified method ChartOCR to extract data from various types of charts. We show that by combing deep framework and rule-based methods, we can achieve a satisfying generalization ability and obtain accurate and semantic-rich intermediate results. Our method extracts the key points that define the chart components. By adjusting the prior rules, the framework can be applied to different chart types. Experiments show that our method achieves state-of-the-art performance with fast processing speed on two public datasets. Besides, we also introduce and evaluate on a large dataset ExcelChart400K for training deep models on chart images. The code and the dataset are publicly available at `https://github.com/soap117/DeepRule`.

1. Introduction

Chart images can be easily found in news, web pages, company reports and scientific papers[24, 18, 10]. Automatic analysis of these data can bring us huge benefits, including scientific document processing, automatic risk assessment based on financial reports, and reading experience enhancement for visually impaired people. However, raw numerical tables are lost when charts are published as images. These underlying data of charts can be easily decoded



	Category0	Category1	Category2	Category3	Category4
Series0	40.58	41.55	41.56	42.34	42.35
Series1	22.66	22.67	24.66	24.68	26.43
Series2	15.02	15.05	15.88	16.85	17.66

Figure 1: Example of data extraction from a chart image

by human, but not by machines [20]. Extracting the raw data table from chart images (see Figure 1 for an example) is the key step for understanding the chart content, which would lead to better analysis of related documents. Recent studies about question answering [12, 5, 13, 14] focusing on querying chart images would also benefit from it.

Some methods [1, 2, 20, 21] have been proposed for chart data extraction. These previous work heavily rely on manually crafted features. The diversity of chart designs and styles makes rule-based chart component extraction approaches difficult to scale. End-to-end solutions based on deep neural networks are also employed to tackle this problem because of their better accuracy [17, 6, 3], but these methods can not generalize well on all the chart types. For example, a framework designed for the pie chart cannot be applied to the line chart. Moreover, comparing to the heuristic rule-based methods, deep end-to-end approaches usually have no control of the intermediate results. Hence, a more general and flexible approach is desired to comprehend various chart images to further enhance the document analysis.

In this paper, we propose an approach that tackles the chart components detection problem with key point detection methods [15, 7, 16]. In this way, the chart data extraction can be simplified as a uniform task regardless of the styles of the chart images. Afterwards, an unified network is used for underlying data extraction. We design a deep hybrid framework that combines the advantages of

* Contribution during internship at Microsoft.

both deep and rule-based methods. As shown in Figure 2, our method first run common information extraction to obtain key points and chart type. Then, we apply type-specific rules to construct the data components (e.g. bar components, sector components) and data range. Finally, we transform these components into structured data format (e.g. tables). It not only exploits the generalization ability of deep methods, but also generates semantic rich intermediate results as in rule-based methods. When dealing with chart images with new styles, we only need to enrich the training data for key point detection network without changing other parts of the framework. The experiments on three datasets FQA, WebData and ExcelChart400K show that our method has good performance on three major chart types, including bar, pie and line charts.

The contribution of this work can be summarized as follows: (1) We propose CharOCR, a deep hybrid framework that combines the advantages of deep-learning and rule-based methods. ChartOCR achieves state-of-the-art performance on chart data extraction task for all three major chart types. (2) We also design new evaluation metrics for these chart types. (3) We collect a fully annotated chart data set with 400K Excel chart images to enable the training of deep learning models.

2. Related Work

2.1. Rule-based Methods

The feature-based methods [4, 8, 20, 23] have been the mainstream for solving chart element extraction problem. They use color continuous searching and edge extraction to find the raw components. Afterwards, predefined rules are applied to eliminate the wrong candidates. However, those methods highly rely on hand-crafted rules and pre-defined features. They are efficient for the data with certain styles, but they can not generalize well on various types. For example, a rule designed for extracting vertically aligned bars can not be used to find horizontally aligned bars. Methods like ChartSense[11] try to solve this problem by adding user interaction and ask users to correct the mistakes during the process. Although adding user input can achieve better performance, it also increases the time cost significantly.

2.2. Deep Neural Networks

Some works try to solve the chart data extraction problem with deep neural networks. For Bar Chart, [6, 17] adopt the idea of general object detection to detect the bar components by treating each bar as an object. For pie Chart, Liu[17] proposes to use the recurrent network and feature rotation mechanism to extract the data. Despite of the great improvement of time efficiency, the deep neural networks are highly restrained to a certain chart type as well. The bounding box detection on bar charts can not be adapted to other chart types like line charts. In the meanwhile, intermediate result like the data range and plot area information cannot be learned with deep methods.

2.3. Keypoint-based Object Detection

Keypoint-based idea has been adopted in many complicated object detection tasks like pose detection[19], face detection[25] and general object detection[15, 7]. Instead of generating the bounding boxes directly, key point methods output the semantically important key points of the target components. For example, in pose detection, the key points are the critical joints of the human body. In face detection, they are the landmark points of the ear, eyes, lip, nose, and mouth. In the chart understanding task, comparing to directly detecting the object bounding boxes, the key points based methods are more flexible for the detection of various chart objects. There is no need to design specific networks for each chart object (e.g. bars, lines and sectors) anymore. Instead, we only extract the keypoints that defines the object. Although the chart objects vary in shape, they are highly structured. This enables us to reconstruct the chart components based on only key points. After the extraction of key points, some task specific rules are applied to group the key points to form complete objects.

3. Our Method

As shown in Figure 2, the framework can be divided into three major parts: common information extraction, data range extraction and type specific detection. The common information extraction includes the key point detection and the chart type classification. Data range extraction determines the range of the numerical values in the plot area. Type specific detection uses the type-dependent rules to obtain the data components (e.g. sectors for pie chart) of each type of chart. Finally by combing the data components and the data range, we can obtain the numerical chart values.

3.1. Common Information Extraction

Key Point Detection In this step, we extract key points of chart components independent of the chart style. With the universal key point detection model, we no longer need to train separate object detection modules for different charts. For chart images with unseen style, we only need to finetune the existing key point detection model by adding more samples that reflect the new chart style.

The key points are defined slightly differently depending on the chart type. For the **bar** chart, the key points are the top-left and bottom-right corner of each separate bar. For the **line** chart, the key points are the pivot points on the line. For the **pie** chart, the key points are the center points plus the intersection points on the arc that segment the chart into multiple sectors. As shown in Figure 3, we adopt a modified version of CornerNet[15] with Hourglass Net[19] backbone

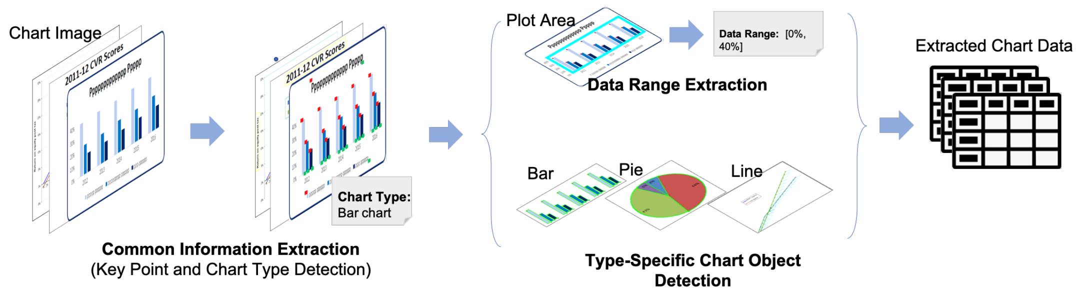


Figure 2: ChartOCR framework outline. There are three major steps: common information extraction, data range extraction and type specific chart object detection. **Common information extraction** aims to detect the keypoints and the chart type. **Data range extraction** infers the range of the data that the chart represents. **Type specific detection** focuses on extracting the chart objects (eg. bars, lines and etc).

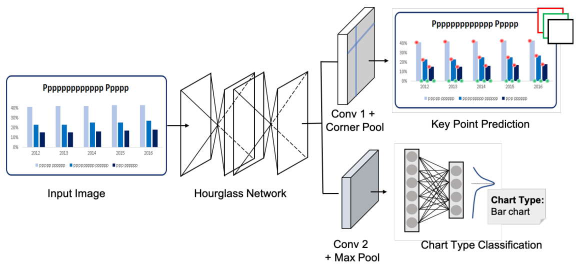


Figure 3: Common information extraction network. We use a hourglass network to provide pixel-level probability map of the keypoint locations. Corner pooling layer is applied on the penultimate layer of keypoint detection branch to increase the receptive field on along the horizontal and vertical direction. (Best viewed in color)

for key point proposal. The output for key point detection network is a probability map that highlights the pixels in key point locations. The probability map has 3 channels to predict the locations of *top-left*, *bottom-right* and *background*. The size of the output probability map is the same as input image. The penultimate layer of key point detection network is a *corner pooling* layer adopted from the CornerNet[15]. Corner pooling layer performs max-pool on the horizontal and vertical direction respectively, which helps the convolutional layers to better localize key point locations. We follow the same setting of CornerNet[15] and define the loss functions as the summation of probability map loss and the smooth L1 loss for keypoint coordinates.

Chart Type Classification We add an additional convolutional layer to the direct output of Hourglass Net and convolve the key point feature map into a smaller size e.g. (32×32). Then we apply max-pooling on it to obtain a one-dim vector. We then feed the intermediate feature vector to fully connected (FC) layers to predict the chart type of the input image. The last FC layer of this branch has `softmax` activation and this branch is trained with cross-entropy loss. Let N be the number of samples in the batch, C be the number of classes. y_{ic} is 1 if and only if ith sample belong to class c, and $\mathbf{p}$ is the prediction of probability distribution. Then the loss can be written as

$$L_{CE}(\mathbf{y}, \mathbf{p}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic} log(p_{ic}) \tag{1}$$

3.2. Data Range Extraction

Data range extraction helps us to convert the detected key points from image pixel space to the numerical readings. The data range extraction applies to line and bar charts. For pie chart, the summation of all the sectors should be 100% by default, thus the data range extraction can be skipped.

We use Microsoft OCR API [1] to extract the text from the image. The extracted text comes from legend, title and axis-labels. For data range extraction, we need to identify the numbers that are associated with y-axis only. To separate out those y-axis labels, we assume that those numbers are always on the left-hand side of the plot area. Thus we only need to locate the plot area, then based on its position, the y-axis labels can be filtered out easily. The plot area is also defined by the top-left and bottom right corners, so we could follow the similar routine as keypoint detection described in 3.1 to locate the plot area. Once we have the plot area location and the OCR result, we design the Data Range Estimation algorithm 1 to get the data range of the chart. In this algorithm, we first use the detected corner points to identify the plot area, then find the recognized numbers that are on the left-hand side of plot area, and finally use the top and the bottom numbers to calculate the data range and pixel range to map the points to the actual data value.

[1]Microsoft OCR API: https://dev.cognitive.azure.cn/docs/services/5adf991815e1060e6355ad44/operations/587f2c6a154055056008f200

Algorithm 1 Data Range Estimation

```
Require: Plot Area Info: $Top, Left, Bottom$. OCR results $R$
Ensure: $Y_{scale}, Y_{max}, Y_{min}$
1: Find the nearest candidate $r \in R$ as $r_{max}$ to point $(Left, Bottom)$
   where $r.r < Left - 4$ and $r.text$ is number
2: Find the nearest candidate $r \in R$ as $r_{min}$ to point $(Left, Top)$
   where $r.r < Left - 4$ and $r.text$ is number
3: $r_{min}.num$ = number($r_{min}.text$)
4: $r_{max}.num$ = number($r_{max}.text$)
5: $Y_{scale} = \frac{r_{max}.num - r_{min}.num}{r_{min}.t - r_{max}.t}$
6: $Y_{min} = r_{min}.num - Y_{scale}(Bottom - \frac{r_{min}.t + r_{min}.b}{2})$
7: $Y_{max} = r_{max}.num + Y_{scale}(\frac{r_{max}.t + r_{max}.b}{2} - Top)$
```

3.3. Type-specific Chart Object Detection

3.3.1 Bar Chart

In Section 3.1, we have extracted the top-left and bottom-right key points from bar images using the key point detection network. In this step, we need to match all the top-left key points to the corresponding bottom-right key points to construct the bar objects. We binarize the key point probability map by threshold value $s = 0.4$. For each top left point p_{tl}, we find the closet bottom right point p_{br} and group them together to obtain the bounding box. The distance measure is defined as a weighted distance on x-axis $\text{dist}_x(.,.)$ and y-axis $\text{dist}_y(.,.)$:

$$\text{dist}(p_{tl}, p_{br}) = \gamma \text{dist}_x(p_{tl}, p_{br}) + \nu \text{dist}_y(p_{tl}, p_{br}) \quad (2)$$

For vertical bar charts, we use $\gamma > \nu$. For horizontal bars $\nu > \gamma$. In the case of vertical bar charts, to find the corresponding bottom-right key point, we only search for the right side of the plot area for each top-left key point.

3.3.2 Pie Chart

To get the location of the pie center and arc points, we use the same key point detection network as decribed in Section 3.1. We replace the corner pooling layer by center pooling layer from [7] to capture the 360-degree neighborhood information. We filter the key points prediction probability map by threshold $s = 0.3$ to get binarized heat map.

For each sector element, the key point detection network extracts the center point p_v and arc point p_{arc}. When grouping the key points to form the sectors, we consider two cases: (1) **tight pie chart** where all the sectors are laying together to form one circle (oval) (2) **exploded pie chart** where one or more sectors are separated from each other. Previous works [6][23] can process the pie charts in the first case but fail to deal with the charts in the second case. In this work, we design a algorithm SECTOR COMBINING 2 to find the key points in each sectors for both cases. For the first case, we only need to sort the arc points in clock-wise order and calculate the portion of each sector. For the second case, we include the pie radius estimation step where we find the optimal radius that can link all center and arc points. The center and arc points has 1:N mapping, meaningly, one or more sectors can be attached with one center point. We check if the distance between a center point and the candidate arc points is within some threshold. If yes, then this pair belongs to the same sector, otherwise not. (Details of the pie radius estimation can be found in the supplemental material.)

Algorithm 2 Sector Combining

```
Require: center points $\{p_v\}$, arc points $\{p_{arc}\}$
Ensure: sectors defined by edge points $[p_v, p^b_{arc}, p^e_{arc}]$
1: if length($\{p_v\}$) == 1 then
2:     for $p_{arc} \in \{p_{arc}\}$ do
3:         find the nearest $p^*_{arc} \in \{p_{arc}\}$ in clockwise order
4:         new sector = $[p_v, p_{arc}, p^*_{arc}]$
5:     end for
6: else
7:     $r^*, t$ = Pie Radius Estimation($\{p_v\}, \{p_{arc}\}$)
8:     for $p_v \in \{p_v\}$ do
9:         for $p_{arc} \in \{p_{arc}\}$ do
10:            find the nearest $p^*_{arc} \in \{p_{arc}\}$ in clock wise order
11:            if $\frac{dis(p_v, p_{arc}) - r^*}{r^*} < t$ and $\frac{dis(p_v, p^*_{arc}) - r^*}{r^*} < t$ then
12:                new sector = $[p_v, p_{arc}, p^*_{arc}]$
13:            end if
14:        end for
15:    end for
16: end if
```

3.3.3 Line Chart

The key point detection network predicts the locations of pivot points on the line. In order to group the key points according to the lines that they belong to, we attach a convolutional layer in the key point extraction branch (after conv1 in Figure 3) as the embedding layer. We enforce the feature embeddings of points in the same line to be as close as possible, and the embeddings from different lines to be as far as possible. We define the embedding loss function following the practice of [15]:

$$e^k_m = \frac{1}{N} \sum_i e^k_i, \text{where } \{e^k_i\} \text{ belong to a same line } k \quad (3)$$

$$loss_{pull} = \frac{1}{K} \sum_k \frac{1}{N} \sum_i (e^k_i - e^k_m)^2 \quad (4)$$

$$loss_{push} = \frac{1}{\mathbf{C}_K^2} \sum_i \sum_{j>i} \max((1 - |e_m^i - e_m^j|), 0) \quad (5)$$

$$loss_{embedding} = loss_{pull} + loss_{push} \quad (6)$$

The total loss of key point detection network for line chart is defined as $loss_{point'} = loss_{point} + \lambda \cdot loss_{embedding}$ where $loss_{point}$ is the summation of the probability map loss and smooth L1 loss described in Section 3.1. We use λ =0.1 in experiments.

To form lines given key points, we adopt the hierarchical clustering strategy to group the embedding of the key points with the classical union-find algorithm. (The details of this algorithm can be found in the supplemental material.) In this way, each cluster contains points that belong to the same line. However, some points may belong to two (or more) lines and they are usually treated as outliers in the clustering algorithm. We call these points *intersection points* and propose the QUERY network to predict which lines they should be assigned to. For each pair, let (x_s, y_s) denote the location of intersection point s, and $e = (x_e, y_e)$ denote the closest point from s that has been assigned to a cluster. We sample K points equidistantly on the line $s - e$. The location of sample points are calculated using following equations:

$$p_k = (x_s + (k-1)d_x,\ y_s + (k-1)d_y) \quad (7)$$

$$d_x = \frac{x_e - x_s}{k},\ d_y = \frac{y_e - y_s}{k} \quad (8)$$

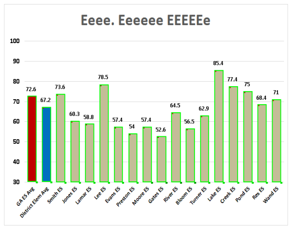
where k means the kth sample point. Since the sample point locations are float numbers instead of integers, we use linear interpolation to obtain the feature of the sample point. Then we can use the QUERY network to take the K sample points as input and classify if point s and e should belong to the same line.

4. Data Set

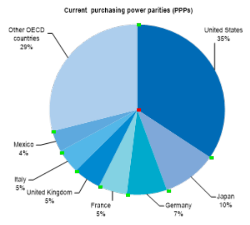
FQA [6] This dataset contains 100 synthetic images for bar chart, pie chart and line chart. However the variation on chart style not large.

WebData [6] This dataset has the same size as FQA. The images are crawled from the web, and the variation in chart style is much larger than FQA.

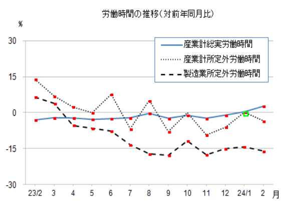
ExcelChart400K Deep neural networks easily overfit on small datasets like FQA and WebData, thus we collect a large-scale dataset that contains 386,966 chart images by crawling public Excel sheets from the web. We first capture the chart image with Excel APIs, then extract the underlying data values of the chart. (To protect privacy, we have conducted data anonymization by overwriting texts in the charts with random characters.) The collected dataset not only provides the bounding boxes locations for the chart components but also the numerical readings of the charts. Figure 4 shows some samples and the annotations from this dataset. Table 1 summaries the statistics of our dataset. Compared with previous chart data sets used in[6, 23, 12, 17], this dataset has a wider range of variations in type and style. Moreover, they are authentic images used in real-world scenarios instead of synthesized from data-generation.



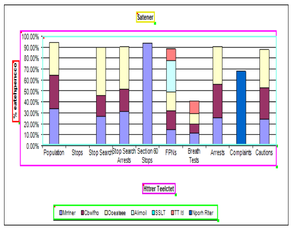
(a) Bar chart



(b) Pie chart



(c) Line chart



(d) Common chart components

Figure 4: Example chart images from ExcelChart400K with annotated ground-truth positions of chart components. a) the bounding box of each bar for bar chart; b) the key point positions of each sector for pie chart; c) the data points of each line for line chart; d) the bounding boxes of chart components (only the position of plot area is used in this paper).

Table 1: ExcelChart400K dataset statistics

Type	train	val	test
Bar	173,249	6,935	6,970
Line	116,745	3,073	3,072
Pie	73,075	1,924	1,923

5. Training Details

In the design of the keypoint detection network, for all three types of chart images, we use the same backbone network- HourGlass Net with 104 layers. During training, we use Adam optimizer with learning rate 2.5e-4 and decrease the learning rate to 2.5e-5 for the last 5,000 batches. Batch size is set to be 27. $\alpha = 2, \beta = 4$. Soft-NMS is applied to merge key points from the heat map. All experiments are conducted in the same environment with 4 Tesla P100 GPUs. For training details of different type of charts please refer to the supplemental material. A validation set contained in ExcelChart400K is used to set the hyper-parameters. We use the early-stopping strategy for the model training.

6. Evaluation Metric

In previous works, researchers usually borrow evaluation metrics from other domains, e.g., object detection or information retrieval. Those methods do not take into account the specialty of chart data. In this paper, we propose three evaluation metrics for three chart types.

6.1. Bar Chart

For bar chart inputs, our goal is to match the bounding box $p = [x_p, y_p, w_p, h_p]$ to the ground truth bounding box $g = [x_g, y_g, w_g, h_g]$. First, we define a custom distance function for pairwise-wise point comparisons:

$$D(p, g) = \min(1, \| \frac{x_p - x_g}{w_g} \| + \| \frac{y_p - y_g}{h_g} \| + \| \frac{h_p - h_g}{h_g} \|) \tag{9}$$

Here we only consider the differences between x, y, h because w is not related to chart reading. Then we compute the pairwise cost matrix $\mathbf{C}$, where $\mathbf{C}_{n,m} = D(p_n, g_m)$. Then we can find the minimum total cost by taking it as the job-assignment problem:

$$cost = \min_{\mathbf{X}} \sum_{i}^{K} \sum_{j}^{K} \mathbf{C}_{i,j} \mathbf{X}_{i,j} \tag{10}$$

Then the score can be defined as $score = 1 - cost/K$, where $K = \max(N, M)$. $\mathbf{X} \in \{0, 1\}$ is a binary assignment matrix since each point will only be assigned once.

6.2. Line Chart

Since a line defines a sequence of continuous data, we treat it as a continuous similarity problem. Let $P = [(x_1, y_1), ..., (x_N, y_N)]$ be the predicted point set and $G = [(u_1, v_1), ..., (u_M, v_M)]$ be the ground-truth set. We define the average error rate between the ground-truth point set G to the predicted point set P using precision and recall

$$Prec(P, G) = Rec(G, P) \tag{11}$$

$$Rec(P, G) = \frac{\sum_{i=1}^{M} (1 - Err(v_i, u_i, P)) * Intv(i, G)}{u_M - u_1} \tag{12}$$

$$F1 = 2 \cdot Prec \cdot Rec(Prec + Rec) \tag{13}$$

where $Err(v_i, u_i, P)$ defines error rate of matching point (u_i, v_i) for point set P. $Intv(i, G)$ defines the ratio of the ith point in final score. More specifically,

$$Err(v_i, u_i, P) = \min(1, \| \frac{v_i - I(P, u_i)}{v_i} \|) \tag{14}$$

$$Intv(i, G) = \begin{cases} \frac{u_{i+1} - u_i}{2} & \text{for } i = 1 \\ \frac{u_i - u_{i-1}}{2} & \text{for } i = M \\ \frac{u_{i+1} - u_{i-1}}{2} & \text{for } 1 < i < M \end{cases} \tag{15}$$

Here $I(P, u_i)$ is a linear interpolation function that computes the value of line P at the point u_i. The $Err(v_i, u_i, P)$ ranges from 0 to 1. The $Intv(i, G)$ ratio will arise if the gap gets bigger between two points as described in Eq. (15). If there are multiple lines in one chart, we will enumerate the combinations to find the best match score.

6.3. Pie Chart

Both the data values and the ordering are important for pie chart reading. For Pie Chart images we consider the data extraction as a sequence matching problem. Let $P = [x_1, ..., x_N]$ be the predicted data sequence in clockwise order and $G = [y_1, ..., y_M]$ be the ground-truth data sequence. Then the matching $score(N, M)$ can be defined as:

$$\begin{aligned} score(i, j) = max(&score(i-1, j), score(i-1, j), \\ &score(i-1, j-1) + 1 - \| \frac{x_i - y_j}{y_j} \|) \end{aligned} \tag{16}$$

$$score = \frac{score(N, M)}{M} \tag{17}$$

Where $\forall i \; score(i, 0) = 0$, $\forall j \; score(0, j) = 0$. The score is obtained through dynamic programming.

7. Experiment

7.1. Baseline Methods

We compare our method with three types of methods: rule-based methods, deep-learning-based methods and off-the-shelf commercial product. For rule-based methods, **Revision**[23] is a model capable for data extraction of bar chart and pie chart. For deep-learning-based methods, we report the performance of **Vis** [6] on the public datasets and ExcelChart400K dataset. We also implement **ResNet+Faster-RCNN** which is an enhanced version of [6, 17] with Faster-RCNN [22] using ResNet [9] backbone for bar chart extraction, and **Rotation RNN** [17] for pie chart extraction. In addition, we report performance on **ResNet+RNN**, a fully end-to-end deep RNN approach as a strong baseline. In this model, after the deep feature extraction network, RNN is directly applied to output the desired data in the sorted order. For commercial product, we use **Think Cell**[2] which is capable for bar chart data extraction. We can only access it

[2]https://www.think-cell.com/

Table 2: Quantitative results on ExcelChart400K with proposed evaluation metrics in Section 6 (the higher the better)

Methods	Bar	Pie	Line
ChartOCR (Ours)	**0.919**	**0.918**	**0.962**
ChartOCR (Ours) + GT Point	0.989	0.996	0.991
ResNet+Faster-RCNN [17, 6]	0.802	-	-
Revision [23]	0.582	0.838	-
ResNet+RotationRNN [17]	-	0.797	-
ResNet+RNN	0.000	0.411	0.644

through the graphical interface for each example, hence we only show qualitative comparison for it by visualizing input images and predicted results.

7.2. Quantitative Analysis

We report the quantitative results on ExcelChart400K with the proposed evaluation metrics in Section 6.

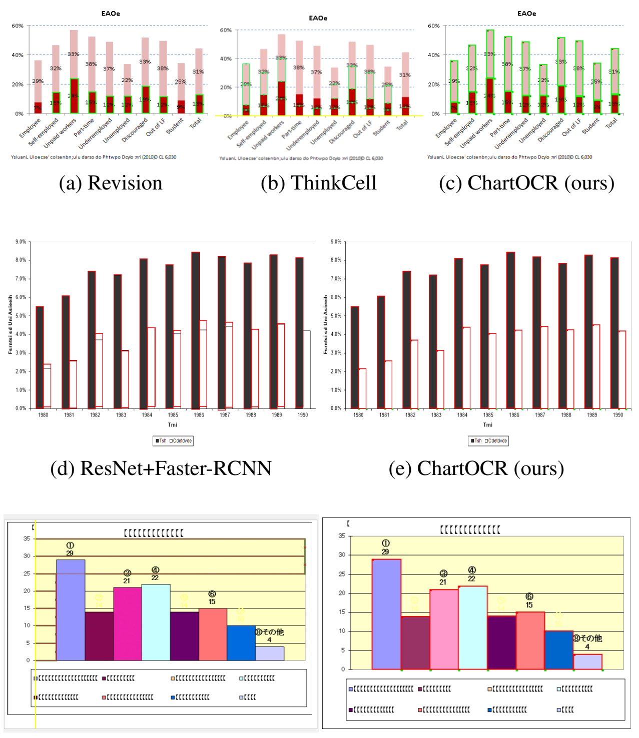
In Table 2, our method has the highest scores in all three types of charts. For **bar chart**, compared with ResNet+Faster-RCNN, our method achieves nearly 14.5% improvement. The improvement is majorly due to the accurate detection of key point positions. As shown in Figure 5, the ChartOCR is more accurate in defining the bars via key points compared with traditional object detection approaches. For **pie chart**, our model outperforms Revision by over 9.5% which is due to the generalization ability of deep key point method. As shown in Figure 7, in the case of detached sectors, key point method can still work well while rule-based method cannot. For ablation study, we also perform an experiment where key point detection results are replaced by the ground-truth key point locations. (See ChartOCR (Ours) + GT Point entry in Table 2). This result can be seen as the upper-bound of our ChartOCR model with the key point based approach. We can see that the performance of bar and pie charts reading has been improved to a nearly perfect score, which means that our rule-based module is exceptionally reliable. For **line chart**, our method improves around 50% compared with the ResNet+RNN baseline.

We also follow the experiment setting[3] of Vis[6] and report the Mean Error Rate in Table 3. For ablation study, we use the ground-truth label information to replace the OCR result, reported as **GT OCR**. Since WebData does not have ground-truth OCR results available, we only compared the methods with and without GT for the FQA dataset. Compared with Vis, ChartOCR shows significant improvement on line and pie type charts. It is because that in Vis, the detection of pie and line components is still based on pure rule-based approach. Ground-truth OCR also greatly reduces the error for FQA data set for bar charts, which indicates that the major error for bar data extraction is caused by poor OCR performance on this dataset.

[3]The comparison is based on reported results in the paper

Table 3: Comparison on the public datasets: FQA and WebData. (* numbers are taken from the original paper)

FQA Mean Error ↓	Bar	Pie	Line
ChartOCR(Ours) + GT OCR	0.093	0.038	0.496
ChartOCR(Ours)	**0.185**	**0.038**	**0.484**
Vis [6]	0.330*	1.010*	2.580*
Revision[23]	0.500	0.120	-
WebData Mean Error ↓	Bar	Pie	Line
ChartOCR(Ours)	**0.285**	**0.439**	**0.740**
Vis [6]	0.450*	0.810*	2.070*
Revision [23]	2.230	0.570	-


(a) Revision (b) ThinkCell (c) ChartOCR (ours)

(d) ResNet+Faster-RCNN (e) ChartOCR (ours)

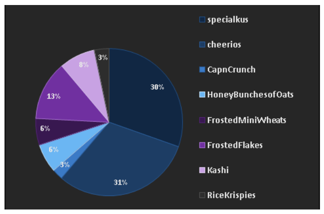
(f) ThinkCell (g) ChartOCR (ours)

Figure 5: Comparison of the methods on bar charts. First row: stacked bar charts; Second row: clustered bar charts; Third row: tight clustered bar charts.

7.3. Qualitative Analysis

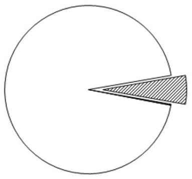
In this section, we compare our method with state-of-the-art methods and show the performance for different type of chart images by visualizing the predicted results.

Bar Chart Cases: In Figure 5, compared with rule-based methods, ChartOCR is more stable on different types of bar



Ground Truth:	0.30	0.31	0.03	0.06	0.06	0.13	0.08	0.03
ChartOCR:	0.30	0.31	0.02	0.06	0.06	0.13	0.08	0.03
Revision:	0.02	0.03	0.38	0.03	0.05	0.41	0.06	0.02
RotationRNN:	0.30	0.31	0.02	0.06	0.06	0.13	0.08	0.03

Figure 6: Both ChartOCR and RotationRNN perform reasonably well for this image, but the Revision method does not work well due to the black background. (Underlines indicate un-matched results from ground-truth)



Ground Truth:	0.06	0.94		
ChartOCR:	0.06	0.94		
Revision:	0.07	0.00	0.90	0.89
RotationRNN:	0.03	0.94	0.02	0.01

Figure 7: For pie charts that contain detached sectors, only ChartOCR has satisfying performance. (Underlines indicate un-matched results from ground-truth)

charts. Rule-based method is not good at extracting stacked elements as shown in Figure 5 (a)(b) and it can be disturbed by the text content. Compared with ResNet+Faster-RCNN in (d), ChartOCR is better at detecting the borders. Figure 5 (b) and (f) show that the commercial product ThinkCell also suffer from the poor generalization problem. In (f) it fails to detect the actual bar components and mistakenly treat the background ruler line as targets.

Pie Chart Cases: Since not every method provides the result for sector extraction[4], here we compare the final numerical output in Figure 6 and 7. In the case of rare background color or detached sector, ChartOCR can still make precise prediction while other methods are severely disrupted.

Line Chart Cases: In this section we only show our results due to the lack of comparable automatic extraction methods. For simple cases such as the first row of Figure 8, ChartOCR gives pretty good result. For hard examples in the second row, the performance is not very satisfying. The reason is that the QUERY network can not deal with complicated situations where multiple line segments are entangled.

7.3.1 Efficiency Analysis

In terms of the time efficiency of each methods, we show the running time in Table 4 which includes the comparison

[4]RotationRNN directly output the percentage of each sector.

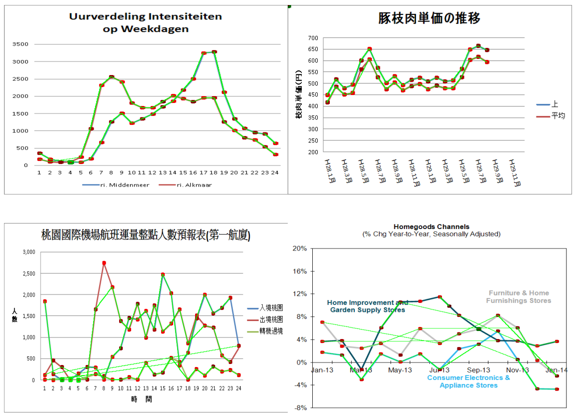


Figure 8: Detection result for line charts. First row: easy samples. Second row: hard samples. Green lines are the predicted lines and red dots are the extracted key points.

Table 4: Average Running Time

Methods	Bar	Pie	Line
ChartOCR	0.206s	0.193s	0.507s
ResNet+Faster-RCNN	0.120s	-	-
Revision	20.032s	5.423s	-
ResNet+RotationRNN	-	0.421s	-

with both the deep learning methods and rule-based methods. As we can see deep methods show a great advantage in time efficiency. For line type ChartOCR takes twice the time of other types, because we added an additional QUERY network as mentioned in Section 3.3.3. The QUERY network does not share parameters with the keypoint detection network. In situations where time efficiency is highly demanded, we can merge these two networks into one single common backbone to reduce the processing time.

8. Conclusion

In this paper we proposed ChartOCR network for precise data value extraction on chart images by combining the rule-based methods and deep neural network based methods. We also introduced a novel benchmark dataset ExcelChart400K that comes with detailed annotations for the chart components. This dataset lays a stepping stone for further research on chart understanding. Our experiments on multiple datasets show that ChartOCR has better performance than both pure rule-based and traditional deep end-to-end methods. Compared with deep models, our chart extractor can be easily generalized for different type of charts such as bar, line and pie charts. Compared with rule-based methods, our approach has much higher precision. For future work, we will expand this work for more chart types.

References

[1] Rabah A Al-Zaidy, Sagnik Ray Choudhury, and C Lee Giles. Automatic summary generation for scientific data charts. In *Workshops at the 30th AAAI Conference*, 2016.

[2] Rabah A Al-Zaidy and C Lee Giles. A machine learning approach for semantic structuring of scientific charts in scholarly documents. In *29th IAAI Conference*, 2017.

[3] Jihen Amara, Pawandeep Kaur, Michael Owonibi, and Bassem Bouaziz. Convolutional neural network based chart image classification. 2017.

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

[5] Ritwick Chaudhry, Sumit Shekhar, Utkarsh Gupta, Pranav Maneriker, Prann Bansal, and Ajay Joshi. Leaf-qa: Locate, encode & attend for figure question answering. *arXiv preprint arXiv:1907.12861*, 2019.

[6] 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.

[7] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. *arXiv preprint arXiv:1904.08189*, 2019.

[8] Jinglun Gao, Zhou Yin, and K. E. Barner. View: Visual information extraction widget for improving chart images accessibility. In *IEEE International Conference on Image Processing*, 2013.

[9] 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.

[10] Weihua Huang and Chew Lim Tan. A system for understanding imaged infographics and its applications. In *Proceedings of the 2007 ACM symposium on Document Engineering*, pages 9–18, 2007.

[11] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. Chartsense: Interactive data extraction from chart images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 6706–6717. ACM, 2017.

[12] Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. Dvqa: Understanding data visualizations via question answering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[13] Kushal Kafle, Robik Shrestha, Brian Price, Scott Cohen, and Christopher Kanan. Answering questions about data visualizations using efficient bimodal fusion. *arXiv preprint arXiv:1908.01801*, 2019.

[14] Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*, 2017.

[15] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018.

[16] Wenbo Li, Zhicheng Wang, Binyi Yin, Qixiang Peng, Yuming Du, Tianzi Xiao, Gang Yu, Hongtao Lu, Yichen Wei, and Jian Sun. Rethinking on multi-stage networks for human pose estimation. *arXiv preprint arXiv:1901.00148*, 2019.

[17] Xiaoyi Liu, Diego Klabjan, and Patrick NBless. Data extraction from charts via single deep neural network. *arXiv preprint arXiv:1906.11906*, 2019.

[18] Yan Liu, Xiaoqing Lu, Yeyang Qin, Zhi Tang, and Jianbo Xu. Review of chart recognition in document images. In *Visualization and Data Analysis 2013*, volume 8654, page 865410. International Society for Optics and Photonics, 2013.

[19] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[20] Jorge Poco and Jeffrey Heer. Reverse-engineering visualizations: Recovering visual encodings from chart images. In *Computer Graphics Forum*, pages 353–363, 2017.

[21] Jorge Poco, Angela Mayhua, and Jeffrey Heer. Extracting and retargeting color mappings from bitmap images of visualizations. *IEEE Transaction on Visualization and Computer Graphics*, 24(1):637–646, 2018.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Sun Jian. Faster r-cnn: Towards real-time object detection with region proposal networks. 2015.

[23] Manolis Savva, Nicholas Kong, Arti Chhajta, Fei Fei Li, Maneesh Agrawala, and Jeffrey Heer. Revision: Automated classification, analysis and redesign of chart images. In *ACM Symposium on User Interface Software & Technology*, 2011.

[24] Sudhindra Shukla and Ashok Samal. Recognition and quality assessment of data charts in mixed-mode documents. *International Journal of Document Analysis and Recognition (IJDAR)*, 11(3):111, 2008.

[25] Yue Wu, Tal Hassner, KangGeon Kim, Gerard Medioni, and Prem Natarajan. Facial landmark detection with tweaked convolutional neural networks. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 40(12):3067–3074, 2017.