Hindawi
Discrete Dynamics in Nature and Society
Volume 2024, Article ID 9835062, 1 page
https://doi.org/10.1155/2024/9835062

Retraction

Retracted: ChartMaster: An End-to-End Method to Recognize, Redraw, and Redesign Chart

Discrete Dynamics in Nature and Society

Received 23 January 2024; Accepted 23 January 2024; Published 24 January 2024

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope

(2) Discrepancies in the description of the research reported

(3) Discrepancies between the availability of data and the research described

(4) Inappropriate citations

(5) Incoherent, meaningless and/or irrelevant content included in the article

(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] 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, Article ID 9559430, 14 pages, 2021.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2021, Article ID 9559430, 14 pages
https://doi.org/10.1155/2021/9559430

Research Article

ChartMaster: An End-to-End Method to Recognize, Redraw, and Redesign Chart

Lingmei Zhang,[1,2] **Guangxia Wang**,[1] **and Lingyu Chen**[1]

[1]*Information Engineering University, Zhengzhou 450000, China*
[2]*Zhengzhou University of Aeronautics, Zhengzhou 450000, China*

Correspondence should be addressed to Lingmei Zhang; zhanglingmei@zua.edu.cn

Received 9 September 2021; Revised 16 October 2021; Accepted 25 October 2021; Published 7 December 2021

Academic Editor: Ahmed Farouk

Chart is one kind of ubiquitous information, which is widely utilized and easy for people to understand. Due to there are so many different kinds and different styles of charts, it is not an easy task for a computer to recognize a chart, as well as to redraw the chart or redesign it. This study proposes a three-stage method to chart recognition: analyze the classification of charts, analyze the structure of charts, and analyze the content of charts. When classifying charts, we choose ResNet-50. When recognizing the structure and content of charts, we use different deep frameworks to extract key points based on different types of charts. Besides, we also introduce two datasets, UCCD and UCID, to train deep models to classify and recognize charts. Finally, we utilize some traditional geometric methods to obtain an original table of a chart, so we can redraw it.

1. Introduction

The era in which we have gone through is called the age of the data. A vast amount of data are produced every day. People spread and communicate all aspects of content through the ubiquitous network, which is called the "ubiquitous network." Through the ubiquitous network, anyone can obtain any information they need at any place, time, and location, which is called ubiquitous information. Ubiquitous information is mainly divided into charts, texts, images, audios, and videos according to the form of expression [1].

The chart has been widely used for data visualization. It is very intuitive, people can read the data of a chart without effort, even more, people can read some other information from a chart at a glance without calculation, such as the longest and shortest ones, the trend of the data [2].

A chart is always created from a table; once a chart has been painted to image, the original table information is lost. Though it is a very easy task for people to read and understand the chart, a very hard task for a computer. To automatically analyze a chart, then getting information (usually original table data) from it can bring us huge benefits. For example, we want a deep learning algorithm to help us predict the future price of some stocks. The box chart of stock price cannot be ignored during such a task. So, to recognize a chart, and get the information from that, is very meaningful. The information that we get from the chart can provide data support for the specific applications of ubiquitous information [3].

To get what a chart means, many people have done some amazing jobs related to research areas. ReVision, presented by Savva et al. [4], aims at classifying and extracting data from charts. They used 2500 chart images that were collected from the Internet and were divided into 10 groups, which are called area, bar, Pareto, and pie charts, curve, radar, and scatter plots, tables, and Venn diagrams. Image classification with 96% accuracy is achieved by ReVision. While marks are extracted from 79% of bar charts, they are extracted from 62% of pie charts. When these charts are utilized, while the data are extracted from 71% of bar charts, it is extracted from 64% of pie charts, which is not a very good performance.

A semimodernized work process, BarChartAnalyzer, is proposed in [5] for data extraction from diagram pictures. This work cycle joins the going with tasks in the game plan: chart type gathering, picture clarification, object recognizable proof, text area and affirmation, data table extraction,

and text layout, and on the other hand, outline overhaul. The data extraction uses second-demand tensor fields from tensor popularity based used in PC vision. The results show that the work process can effectively and unequivocally eliminate data from pictures of different objectives and of different subtypes of visual diagrams. However, this method can fail in some applications.

ChartSense, proposed by Jung et al. [6], also provided a method of chart classification and extraction of data. The chart type of a given chart image is first determined by the ChartSense utilizing a classifier based on the deep learning method. Then, the underlying data of the chart image are extracted utilizing algorithms called semiautomatic and interactive extraction that are optimized based on each type of chart. But it was a semiautomatic algorithm, involved with a mixed-initiative interaction, that needed people to do some labeling works, so it lacks efficiency.

In [7], a new methodology called Chartem is proposed to address an information implanting composition to encode a lot of data away from the plain sight of a diagram picture without meddling with the human impression of the graph. The inserted data, when removed from the picture, can empower an assortment of representation applications to reuse or repurpose outline pictures. However, this methodology cannot be applicable to all three purposes of identification, redrawing, and redesign.

WebPlotDigitizer, presented by Rohtagi [8], is an Internet application having two different ways of implementation called automatic and manual. While automatic implementation functions based on color detection discriminating the data points from the background image and data points are retrieved, manual one needs users to employ some information to obtain data.

A new framework called DataQuilt is introduced in [9] that empowers perception creators to iteratively plan pictorial representations as arrangements. Genuine pictures (e.g., works of art, photos, and portrays) go about as the two motivations and as an asset of visual components that can be planned to information. The inventive pipeline includes the semidirected extraction of applicable components of a picture (subjective locales, customary shapes, shading ranges, and surfaces) helped by PC vision procedures, the limiting of these graphical components and their provisions to information to make significant perceptions, and the iterative refinement of the two elements and representations through direct control. However, the scalability of this methodology is one of the main challenges.

A neural network design is proposed in [10] that is prepared to recognize among various sorts of outlines, for example, line diagrams or dissipate plots, and anticipate the amount they will have partaken in social communities. This stances critical difficulties due to the shifting configuration and nature of the outlines that are posted and the restrictions in existing preparing information. The proposed framework beats related work in graph type grouping on the ReVision corpus. Besides, publicly support is utilized to assemble another corpus comprising of graph pictures shared by information columnists on Twitter. However, this methodology cannot be applicable to all fields.

ChartOCR, introduced by Luo et al. [11], proposed a combination of both rule-based methods and deep neural networks to extract precise data of chart images, which deals with the key points defining chart components. Different chart types could utilize the same framework by tuning the prior rules. Experiments suggest that a fast processing speed is achieved by our method as a state-of-the-art performance utilizing two public datasets. A novel benchmark dataset ExcelChart400K is also introduced. But it only included three types of charts: bar, line, and pie.

In this study, first, we studied the classification of charts. It is very important, and all other jobs are based on chart classification, including the analysis of the chart structure, the recognition of chart patterns, the data extraction, and so on.

Second, we worked out a way to recognize a pattern of a chart. To get things simple, we need to analyze the structure of the chart first. With structure analysis, a title, the titles of x and y axes, legend, painting area, and including labels painting area can be obtained (the detailed explanation of painting area and including labels painting area are in Section 3). With painting area and including labels painting area, along with OCR technologies, we can get labels. Inside the painting area, is where chart patterns are painted. Based on chart classification, we can now use various ways to recognize vertical bar, horizontal bar, box, line, wedge, and radar. All those patterns are defined in their drawing properties such as bounding boxes to define bar, a group of points to define line, start and ending points of the arc, and the degree to define a wedge.

With all the above works done, we can redraw and obtain original tables. We need to associate labels with chart patterns and then use labels to calculate data. Finally, we can obtain a table chart and redraw or redesign the chart. Therefore, in this study, a new approach dealing with the most commonly used charts is proposed for an end-to-end chart recognition, redraw, and redesign. The proposed approach can solve the problems of the previous approaches. In addition, it can provide higher accuracy to increase efficiency.

2. Overview

This section contains three sections: chart classification, chart structure analysis, and chart content analysis. The three sections are also ChartMaster's three main tasks. We use image classification technologies to classify charts and use object detection and OCR technologies to study chart structure analysis and chart content analysis. When pairing labels to chart patterns, we use traditional geometric methods.

2.1. Chart Classification

2.1.1. Chart Types. As we can see from the last chapter, most previous jobs simply defined the chart type just as bar, line, and pie. This has no problem when detecting and recognizing chart patterns, but we want to get the original table to redraw or redesign a chart, which is not an enough method.

For example, the methods to get the original table data are different in similar charts. Take vertical bars, for example, for a simple one (Figure 1); we just need to pair every bar with an x-axis label, and its original table is very simple just one row. But for a complex one (Figure 2), besides pairing every bar with the specific x-axis label, we need further to pair that bar with legend.

To recognize the drawing pattern of the chart, as long as to get the original table to redraw the chart, we need more types when classifying the chart. In this study, we will study 15 most commonly used charts, namely, area, horizontal group bar, line, radar, vertical single bar, bar and line, horizontal single bar, donut and pie, scatter, vertical and horizontal stacked bars, donut, pie, vertical group bar, and vertical box (Figure 3).

2.1.2. The Method of Chart Classification. Chart classification is an image classification task. Image classification has been revolutionized and propelled technological advancements. The CNN (convolutional neural network) is the primary neural network used in computer vision and image classifiers. Some of the widely used CNN architectures are called AlexNet [12], VGG [13], Inception [14], and ResNet [15]. We choose ResNet-50 for chart classification in this manuscript.

A residual mapping is explicitly fitted by few stacked layers using a residual network (ResNet). Figure 4 shows shortcut connections that skip one or more layers. Identity mapping is executed by the shortcut connections whose outputs are adjoined to the outputs of the stacked layers.

A residual network, ResNet-50, has 50 layers depth, where Table 1 provides its architecture. A pretrained version of the network that is trained with more than a million images from the ImageNet database [16] is loaded.

2.2. Chart Structure Analysis. Chart structure analysis is particularly important for chart recognition. With chart structure analysis, we extract some common elements from a chart and can turn a complex chart recognition problem into several simple tasks.

2.2.1. The Definition of the Chart Structure. Although there are many types of charts, almost all charts contain several of the following elements: title, titles of x and y axes, legend, painting area, and including labels painting area. The painting area (Figure 5) is where all bars, lines, and wedges are painted; when a chart has axes, this area is always the bounding box of the axes. And including labels painting area (Figure 6) is little larger; it contains all labels in the chart, and here are some examples (Figure 7).

2.2.2. CornerNet. The detection of those elements is conducted by chart structure analysis. This is a typical object detection problem. Identifying and locating objects within an image as a technique is called object detection. One of the state-of-the-art approaches to detect objects is called the convolutional neural network (CNN). This manuscript

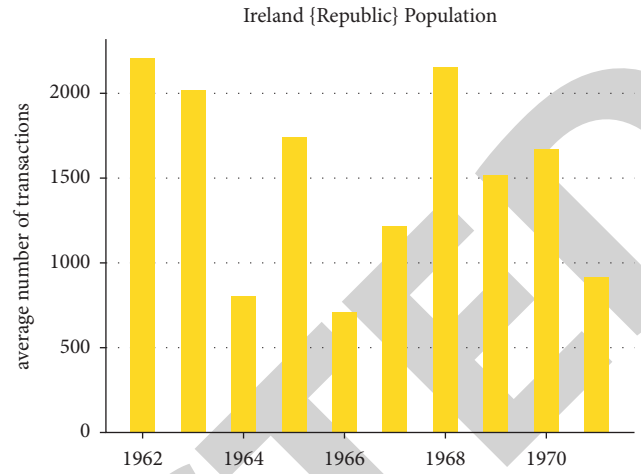

Figure 1: Pair bar with x-axis label to get data.

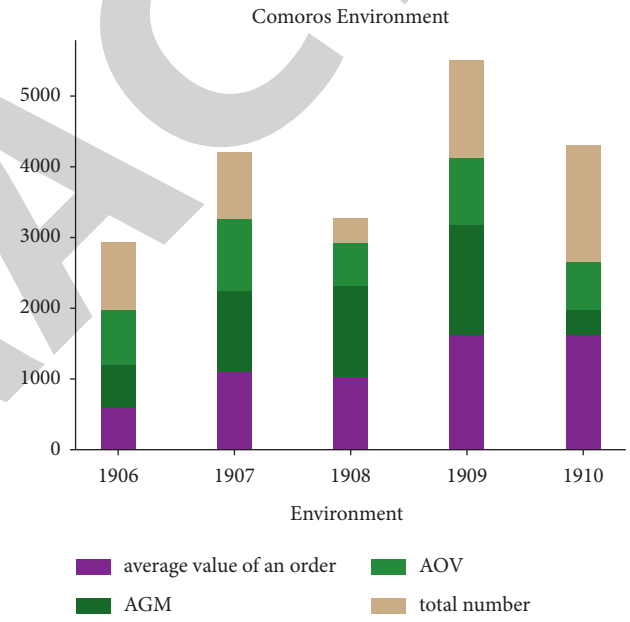

Figure 2: Pair bar with x-axis label and legend to get data.

utilizes CornerNet, which is a key point detection technology, for chart-related object detection [17]. Key point detection consists of locating key object parts. An object bounding box as a pair of key points is detected by CornerNet, which is called the top-left and the bottom-right corners.

A heat map for the top-left corners of all instances of the same object category is predicted by CornerNet, employing a single convolutional network. Also, it can be utilized to obtain a heat map for all bottom-right corners and an embedding vector for each detected corner. Grouping a pair of corners belonging to the same object is served by embedding. Similar embedding is predicted by the trained network. Corner pooling, another novel component of CornerNet, is a new type of pooling layer that aids a convolutional network to better localize corners of bounding boxes.

Figure 3: 15 most commonly used types of charts.

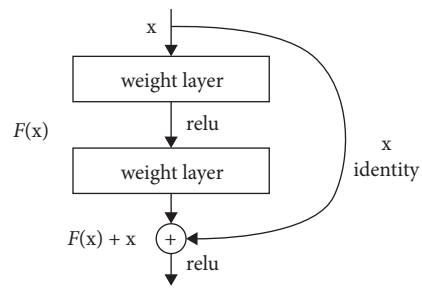

Figure 4: Block of residual.

The hourglass network [18], called the backbone network, is utilized by CornerNet, which is followed by two prediction modules. While the top-left corners are predicted by one module, the bottom-right corners are predicted by the other. A corner pooling module for each module before predicting the heat maps, embedding, and offsets is utilized to pool features of the hourglass network. It differs from other object detectors since features from different scales to detect objects of different sizes are not employed by CornerNet. Both modules only used by CornerNet to the output of the hourglass network. Figure 8 shows its architecture.

Table 1: ResNet-50 architecture.

Layer name	Output size	18-layer	34-layer	50-layer	101-layer	152-layer
Conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
Conv2.x	56×56	$\begin{bmatrix} 3\times3, & 64 \\ 3\times3, & 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, & 64 \\ 3\times3, & 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 64 \\ 3\times3, & 64 \\ 1\times1, & 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 64 \\ 3\times3, & 64 \\ 1\times1, & 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 64 \\ 3\times3, & 64 \\ 1\times1, & 256 \end{bmatrix}\times3$
Conv3.x	28×28	$\begin{bmatrix} 3\times3, & 128 \\ 3\times3, & 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, & 128 \\ 3\times3, & 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, & 128 \\ 3\times3, & 128 \\ 1\times1, & 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, & 128 \\ 3\times3, & 128 \\ 1\times1, & 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, & 128 \\ 3\times3, & 128 \\ 1\times1, & 512 \end{bmatrix}\times8$
Conv4.x	14×14	$\begin{bmatrix} 3\times3, & 256 \\ 3\times3, & 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, & 256 \\ 3\times3, & 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, & 256 \\ 3\times3, & 256 \\ 1\times1, & 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, & 256 \\ 3\times3, & 256 \\ 1\times1, & 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, & 256 \\ 3\times3, & 256 \\ 1\times1, & 1024 \end{bmatrix}\times36$
Conv5.x	7×7	$\begin{bmatrix} 3\times3, & 512 \\ 3\times3, & 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, & 512 \\ 3\times3, & 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 512 \\ 3\times3, & 512 \\ 1\times1, & 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 512 \\ 3\times3, & 512 \\ 1\times1, & 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 512 \\ 3\times3, & 512 \\ 1\times1, & 2048 \end{bmatrix}\times3$
	1×1	Average pool, 1000-d fc, softmax				
FLOPs		1.8×109	3.6×109	3.8×109	7.6×109	11.3×109

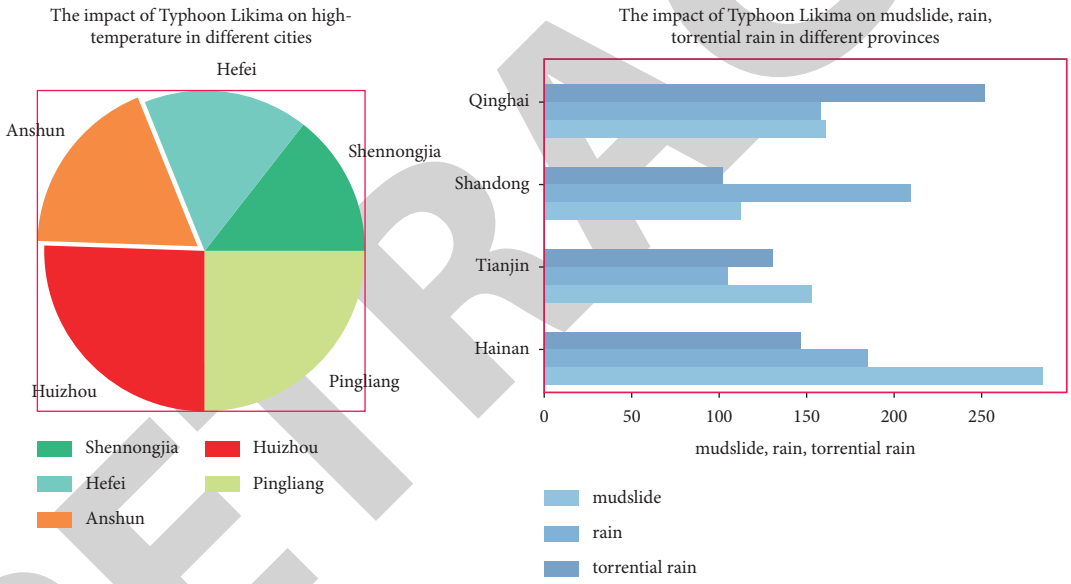

Figure 5: Painting area.

The structural elements of the chart (Figure 9) called title, titles of x and y axes, legend, painting area, and including labels painting area are bounding boxes. The top-left and bottom-right corner points are key points. We pair the top-left point to the closest bottom-right point (Figure 10).

2.3. Chart Content Analysis. Besides chart structure elements, inside including labels painting area, there are labels and chart patterns that need to be detected too. Furthermore, we need to recognize all the texts that we have detected so far. All those jobs will be done in this section: first, we will also use key point detection technologies to recognize chart patterns painting; then, we will discuss OCR technologies that help to detect texts (labels and texts in legend) and recognize them (title, titles of x and y axes, labels, and texts in legend); at last, we will talk about the way we pair chart patterns and labels.

Based on this work, we can get the original table of the chart; then, we can redraw the chart in different types using third-party chart tool libraries, such as matplotlib [19] and eChart [20].

2.3.1. The Detection of a Chart Pattern. We still use CornerNet to detect chart patterns. Defining key points are realized differently based on a chart type.

(1) Just Points Charts. For charts with just points, for example, scatter, the key points are the center points. To detect the scattered points, we add a convolutional layer to CornerNet (Figure 11).

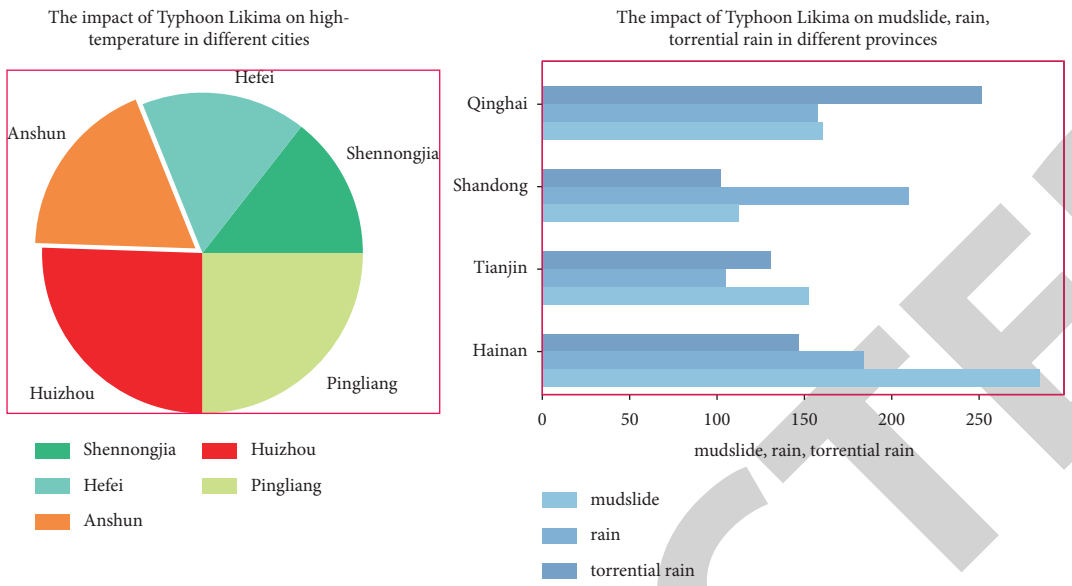

Figure 6: Including labels painting area.

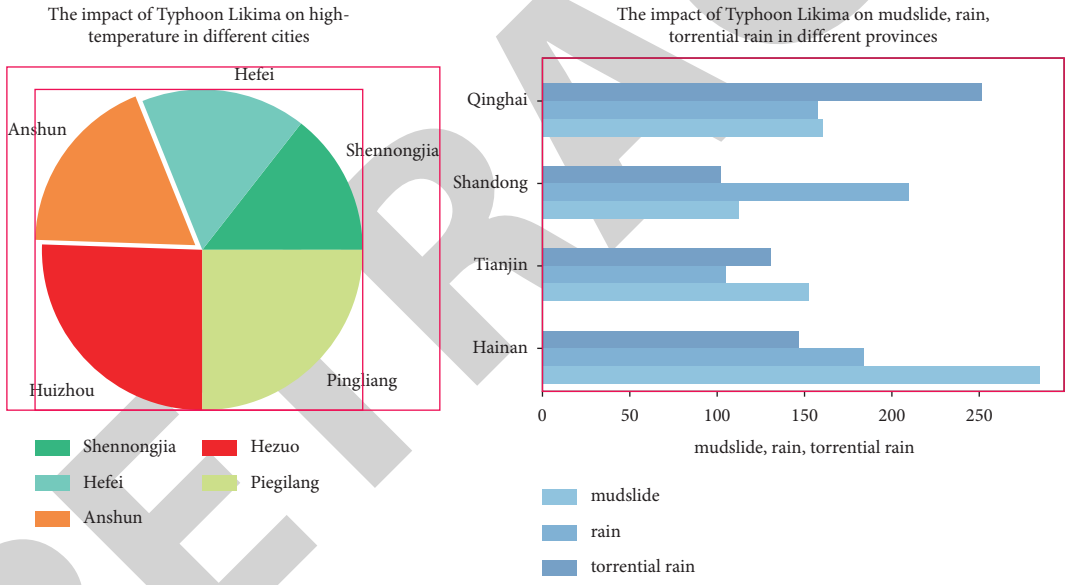

Figure 7: Painting area and including labels painting area.

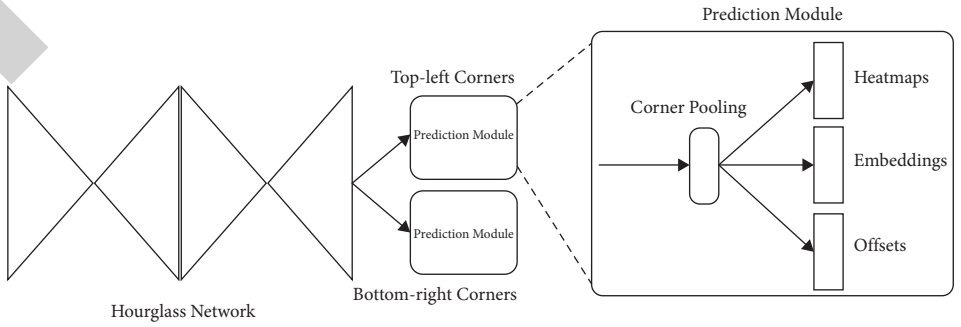

Figure 8: CornerNet architecture.

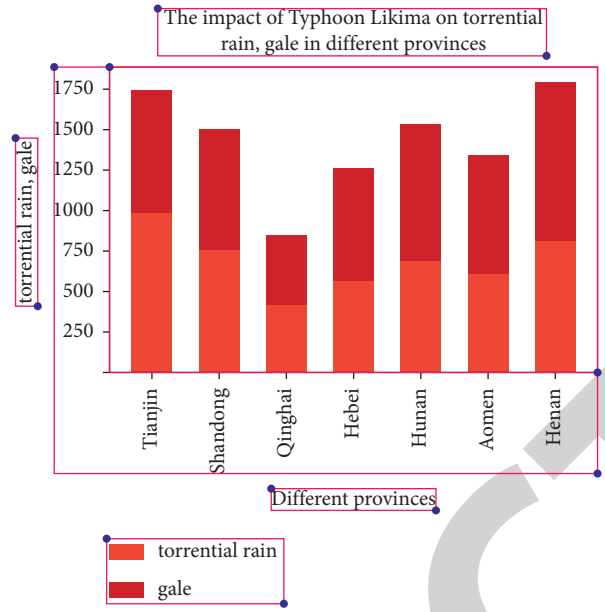

Figure 9: Chart structure elements.

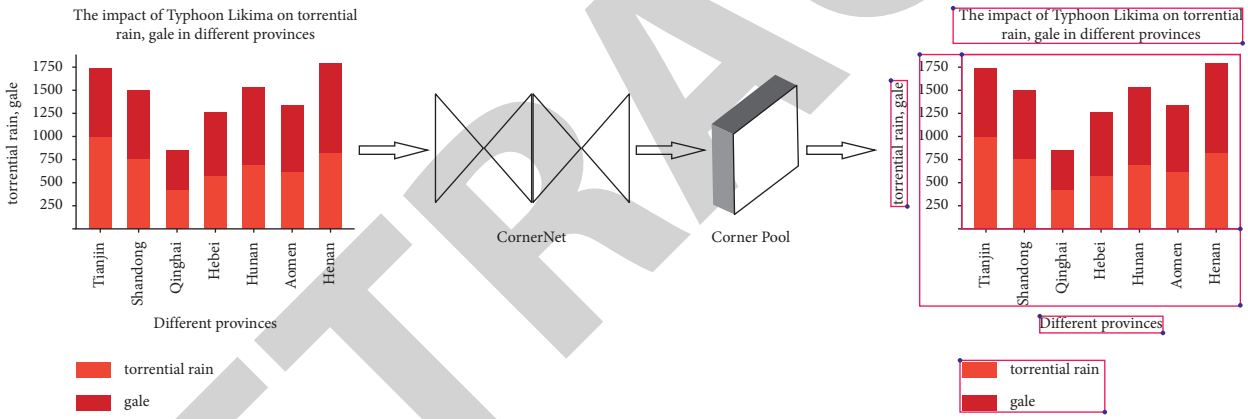

Figure 10: Chart structure elements detection network.

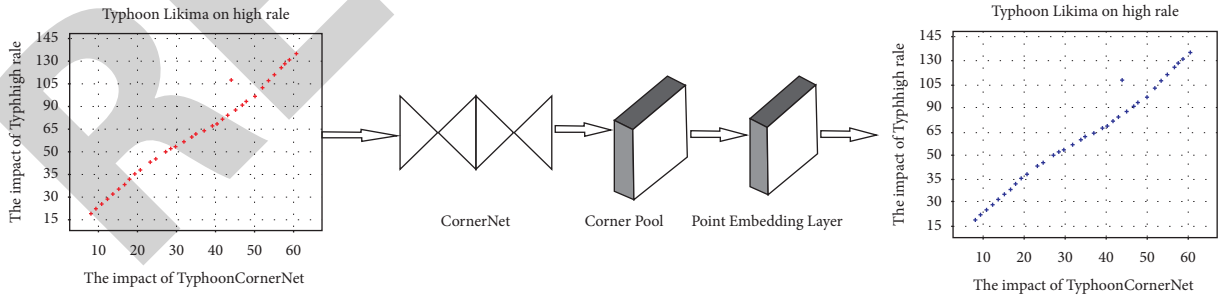

Figure 11: Points detection network.

(2) Bar-Related Charts. For bar-related charts, such as horizontal group bar, vertical single bar, horizontal single bar, vertical stacked bar, horizontal stacked bar, and vertical group bar, each separate bar has key points that are called the top-left and bottom-right corners. The network is the same as the structural elements of the detection network (Figure 12).

(3) Line-Related Charts. For line-related charts, such as area and line, the pivot points on the line are called the key points. We applied the ChartOCR's line chart algorithm, adding a convolutional layer to CornerNet (Figure 13).

(4) Wedge-Related Charts. For wedge-related charts, such as pie, donut, donut and pie, and radar, the center point, start

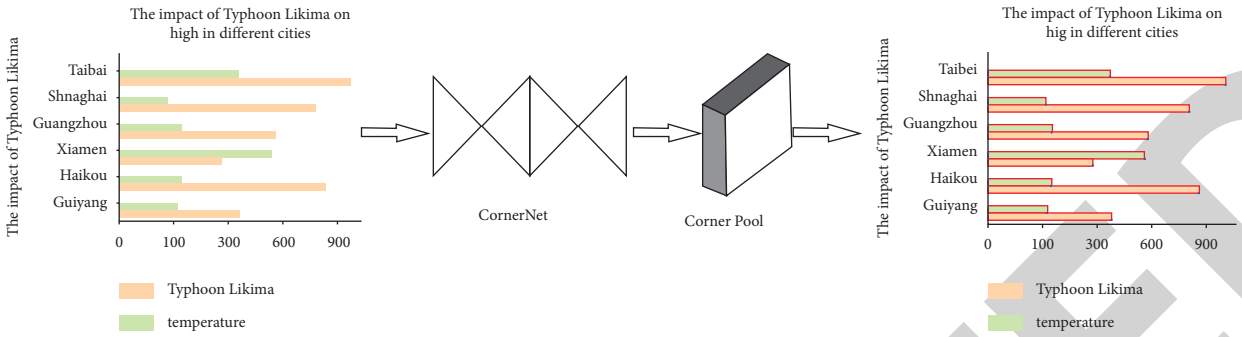

Figure 12: Bar detection network.

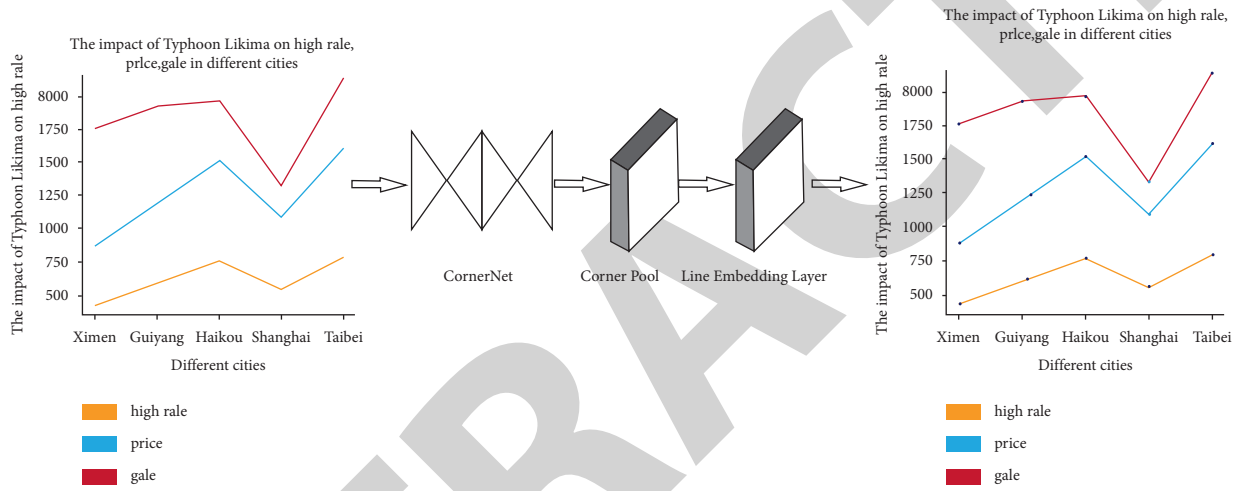

Figure 13: Line detection network.

and endpoints of the arc, and degree of the wedge are called the key points. We also applied the ChartOCR's pie chart algorithm to detect wedges, replacing the corner pooling layer with the center pooling layer (Figure 14).

(5) Mixed Charts. We have two mixed types of charts in our 15 types of charts, bar, and line, vertical box. Just as we have discussed above, each separate bar having the top-left and bottom-right corners is called the key points. Besides, the pivot points on the line are called the key points too. To detect bar and line, we also add a convolutional layer to CornerNet (Figure 15).

2.3.2. OCR Technology. When we are concerned with obtaining labels or titles that we talked about in the last section, OCR (optical character recognition) cannot be ignored. Unlike titles, which area has been detected in the last section, we need to detect labels through OCR technologies, and we choose DB. Differentiable binarization (DB) [21], performing the binarization process in a segmentation network, is optimized along with a DB module (Figure 16). The thresholds for binarization can be adaptively set by a segmentation network that not only simplifies the postprocessing but also improves the performance of text detection.

After we get text region bounding boxes, we need OCR technologies to recognize the text, in which we choose the CRNN (Figure 17). Diagnosing various faults of the HST bogie is conducted by a convolutional recurrent neural network (CRNN) [22]. Both capabilities of the CNN and RNN are inherited simultaneously. Features are filtered out from the original data by the CRNN utilizing convolutional layers based on the novel architecture.

2.3.3. Label Exaction and Chart Pattern Matching. We first need to obtain labels and then match those labels with specific chart patterns. There are two main kinds of methods based on our method of chart classification.

(1) Chart with Axes Label Exaction and Matching. For those charts that have axes (or painted without axes, but we can infer there are, such as bar-related chart, line chart, and area chart, Figure 18) between painting areas and including labels painting area, we can easily get labels. Because those labels always lay outside the painting area but lay inside including labels painting area. Using OCR technologies that we just talked about above, we can easily obtain labels, and those labels we call axis labels.

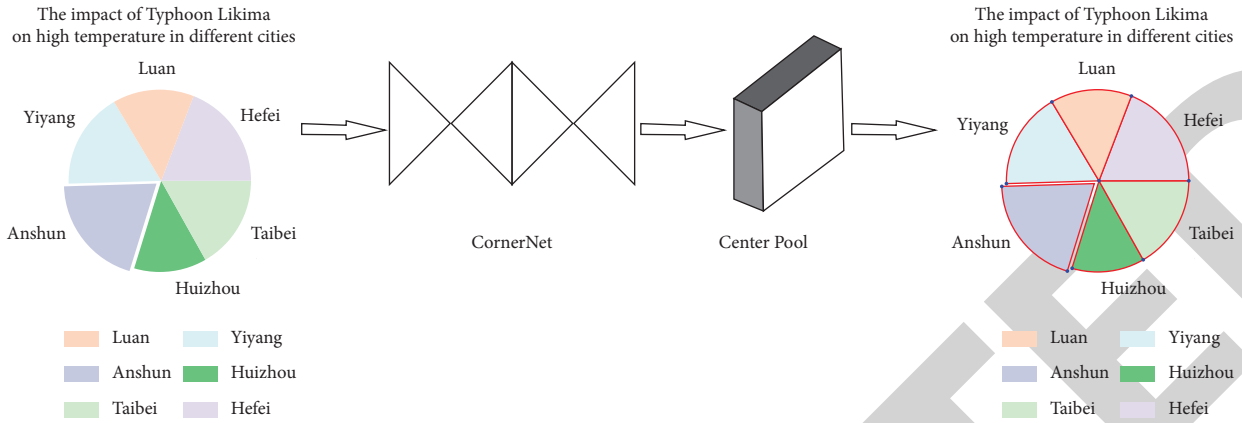

Figure 14: Wedge detection network.

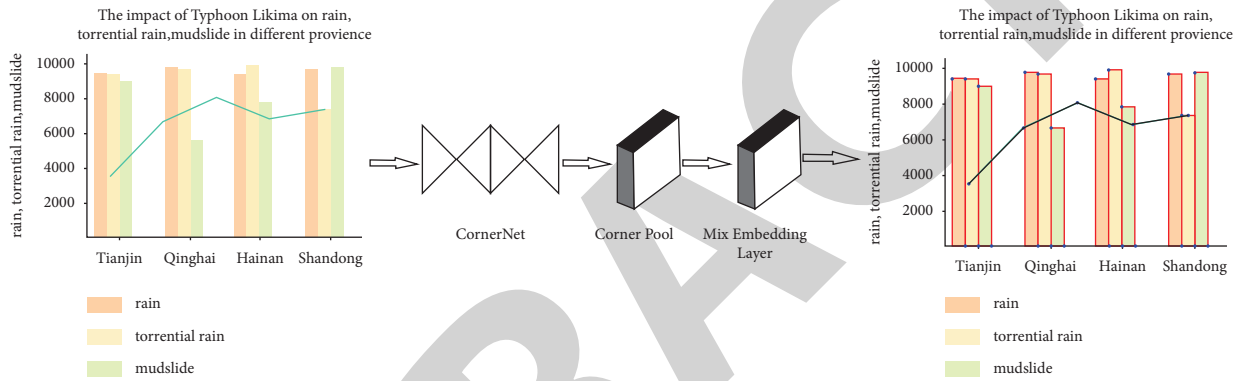

Figure 15: Bar and line chart detection network.

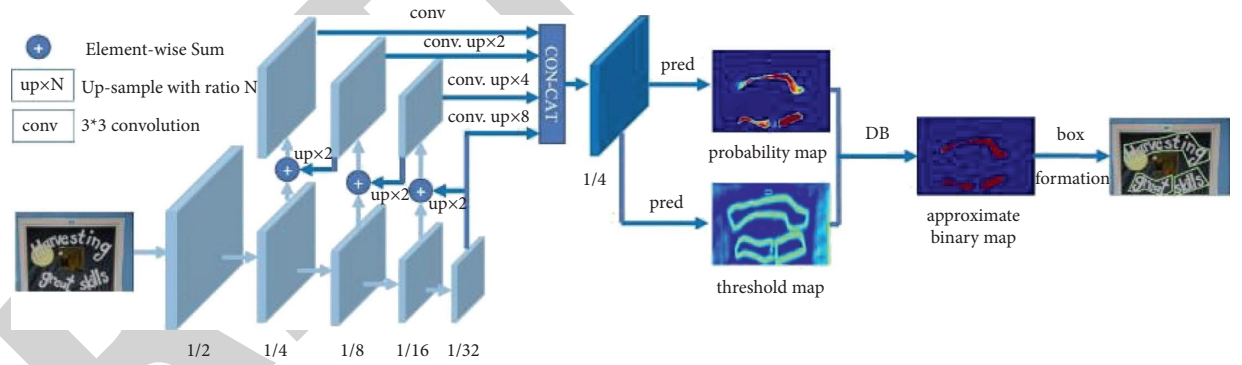

Figure 16: Differentiable binarization architecture.

Step 1. Label analysis

There are two kinds of axis labels, one is the horizontal label (x-axis label) and the other is the vertical label (y-axis label). The x-axis labels always lay left or right beside the painting area, and y-axis labels always lay above or below beside the painting area. For vertical single bar, vertical stacked bar, vertical group bar, line, and area charts, y-axis labels are value-related labels and x-axis labels are property-related labels. For horizontal single bar, horizontal stacked bar, and horizontal group bar, the situation is the opposite.

Step 2. Value calculating

We should convert value-related labels to numbers and use their coordinates to calculate corresponding chart pattern values. We just use the top-bottom labels (y-axis labels are value-related labels) and left-right labels (x-axis labels are value-related labels), which means the biggest and smallest value, along with the pixel coordinates, and we can get every chart pattern's value in that chart.

Step 3. Pairing value, label, and legend

Finally, we need to pair values to properties. For single bar charts, we just pair chart patterns with the corresponding property-related label; for every property-related label, there

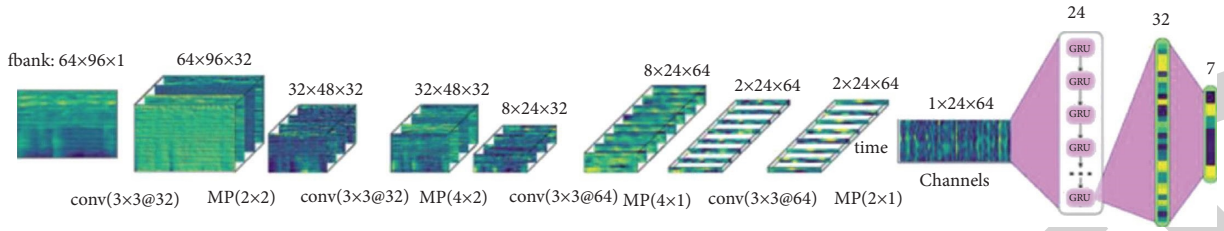

Figure 17: CRNN architecture.

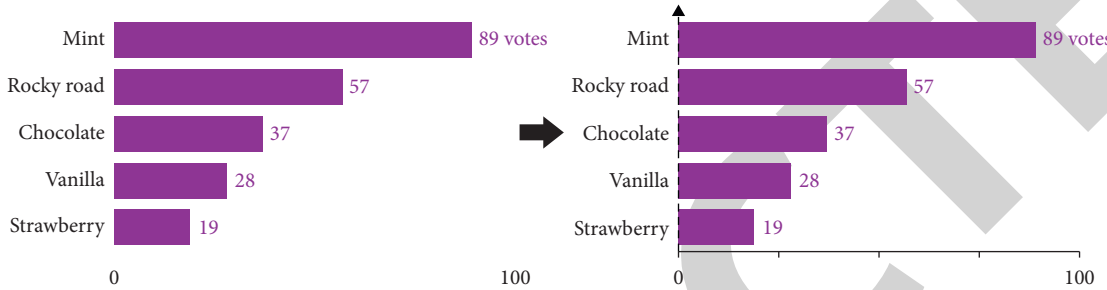

Figure 18: The chart that can infer axes.

is just one chart pattern (in those charts' original table, there is just one row). But for those charts (such as stacked bar, group bar, line, and area charts), whose property-related label may have several corresponding chart patterns, we need to further analyze the legend, to pair property, legend, and value to every specific chart pattern; then, we can get the multiple rows original table (Figure 19).

(2) Chart without Axes Label Exaction and Matching. For those charts that do not have axes (such as pie, donut, donut and pie, and radar), getting labels out is a little bit more complex. We still need a painting area and including label painting area, and the labels are strictly inside included labels painting area, but not strictly outside the painting area, for the painting area is a rectangle (Figure 20).

Step 4. Label analysis

We need to find all the center points of the bounding boxes of texts inside including the labels painting area. As we have analyzed chart patterns, we can use the chart pattern information to get labels out from as pie, donut, donut and pie, and radar charts. We can employ the winding number algorithm [23] to check whether a center point is inside or outside a pattern, and then get all outside chart pattern labels out (Figure 21).

Since those labels in those types of charts are always just one kind, property-related labels, so we now get property-related labels out (there may be property-value-related labels; in this situation, we need a word segmentation algorithm to obtain property-related labels out).

Step 5. Value calculating

We no longer need value-related labels to calculate values, values are simply the percentage of that pattern area divided by the whole group area.

Step 6. Pairing value, label, and legend

For each chart pattern, we use the start and endpoints of the arc of that chart pattern to draw a line; then, we calculate the center point of that line. We use that center point to calculate all distances of all labels' center points, the shortest one match the chart pattern.

If there are no property-related values in Step 1, then we pair that chart pattern with legend (Figure 22).

3. Dataset and Training Details

The dataset, FQA [24], has 100 synthetic images for bar, pie, and line charts whose chart style does not have large variations.

The size of WebData [24], whose images are crawled from the web, is the same as that of FQA. Nevertheless, it has a much larger variation in chart style than does FQA.

The dataset, ExcelChart400K [11], is collected as a large-scale dataset including 386,966 chart images. To protect privacy, random characters are utilized to overwrite texts in charts. Three types of charts, which are called vertical bar, line, and pie, are used for this dataset.

In this study, we created two kinds of chart datasets. One is for classification and the other is for detection and recognition of chart patterns. To generate a dataset, we first searched a lot of topics from the Internet, such as economy, weather, and natural disaster statistics; then, we create location and time lists for properties; we then randomly generated data. We applied various matplotlib settings when creating charts to generate different kinds of styles.

We first created a classification dataset that contains 9000 images of 15 types of charts, we call it Ubiquitous Chart Classification of Dataset (UCCD). After training for 50 epochs, we archived a result of 99.7% accuracy.

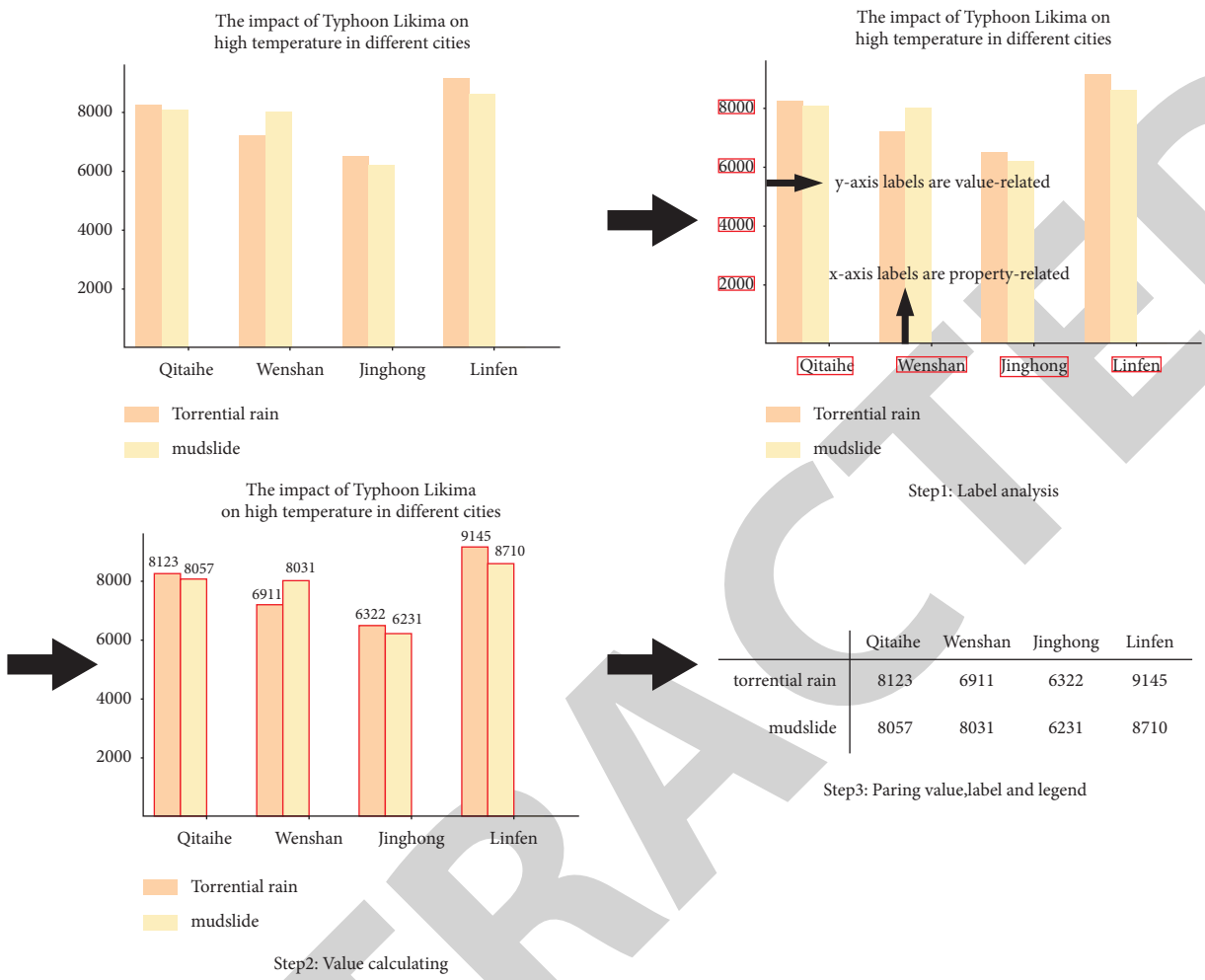

Figure 19: Chart with axes label exaction and matching example.

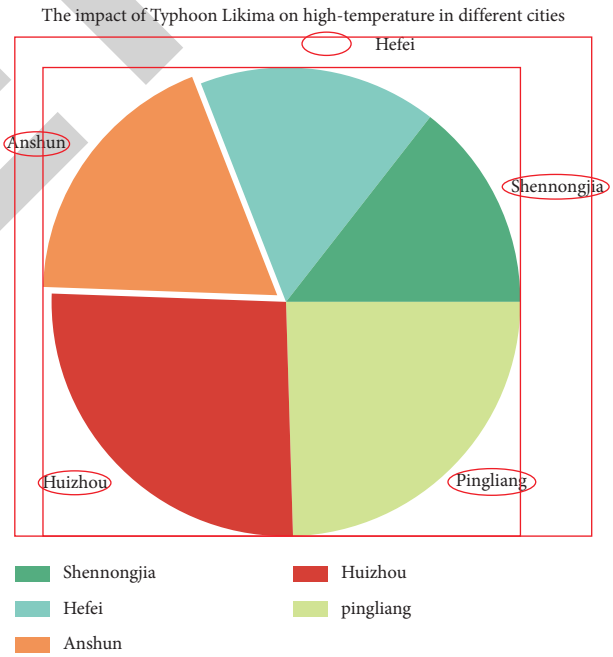

Figure 20: Labels are not strictly outside the painting area.

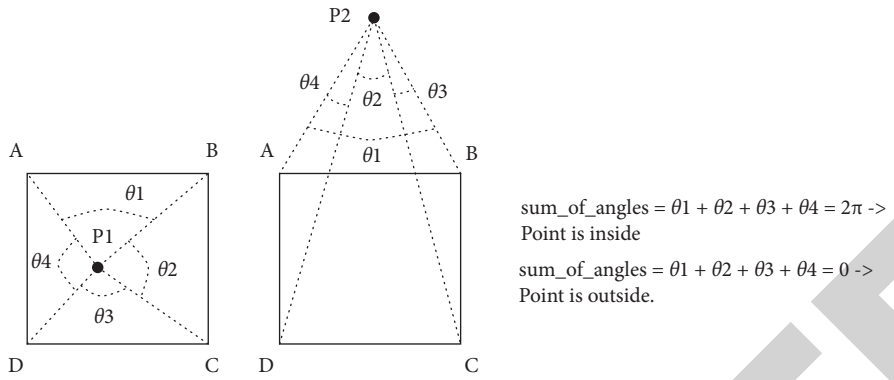

Figure 21: Winding number algorithm judging whether a point is inside or outside.

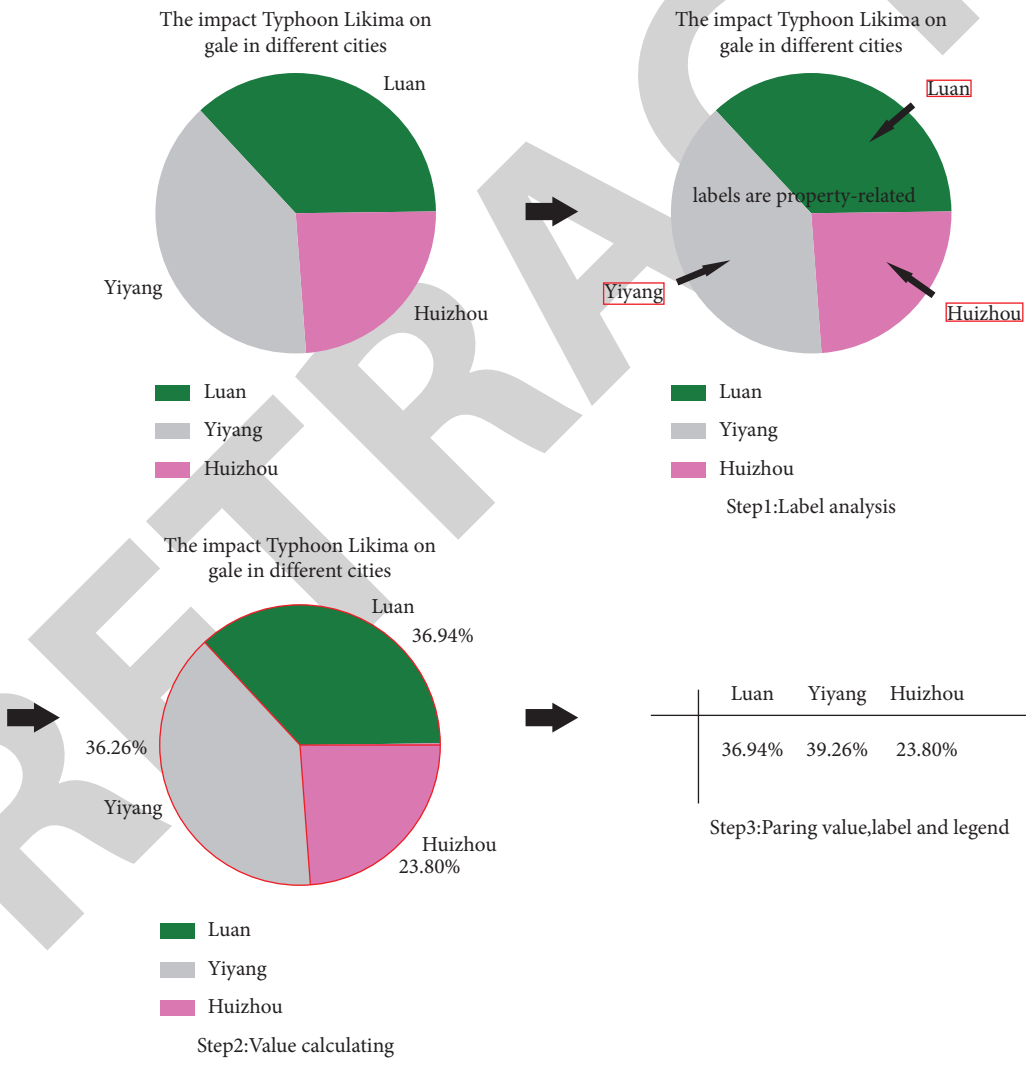

Figure 22: Chart without axes label exaction and matching example.

Then, we created 5 types of as many as 276,536 charts called Ubiquitous Chart Information of Dataset (UCID), in which each type of chart has a similar number. The reason why we just create 5 types is that some types of charts' recognition methods are the same, just as we have talked about in the last section. The 5 types are as follows: scatter, horizontal bar (horizontal group bar, horizontal single bar, and horizontal stacked bar), vertical bar (vertical single bar,

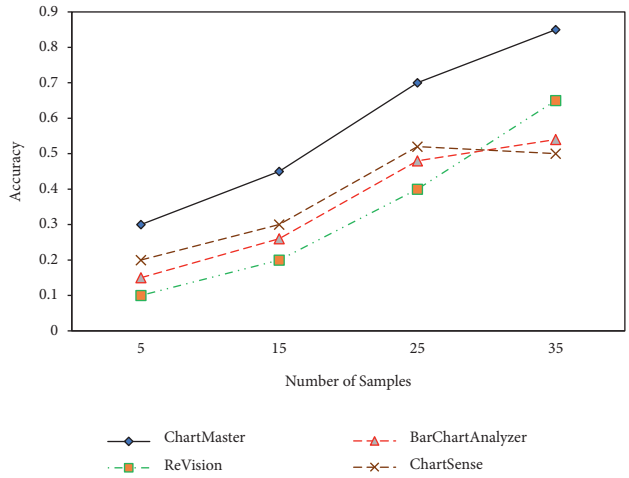

Figure 23: Accuracy as a function of the number of samples.

vertically stacked bar, and vertical group bar), line (area and line), pie (radar, donut and pie; scatter, donut and pie), bar and line (bar and line, vertical box).

UCID contains bounding boxes of title, titles of x and y axes, legend, labels, painting area and including label painting area, and the value of titles and labels. For chart pattern, our dataset contains bars' bounding boxes. lines' group coordinates, wedges' start points of arcs, endpoints of arcs, and degrees.

The Adam optimizer having the learning rate of $2.5e-4$ is utilized in the training where the size is set to be 25.8 Tesla V100. GPUs are conducted with all 5 types in the same environment. The accuracies of scattering, horizontal bar, vertical bar pie, and bar and line are 93.8%, 99.1%, 98.9%, 98.3%, and 94.6%, respectively.

Finally, the accuracy of the proposed approach is compared with some similar approaches. The results are shown in Figure 23. As Figure 23 shows, the proposed approach (ChartMaster) provides higher accuracy than the similar approaches.

4. Conclusions and Prospective Research

The recognition process of the chart is divided into two phases called the classification of the chart and the extraction of the data from it. We use ResNet-50 to classify charts and proposed 5 different networks for data extraction. Based on those two tasks, we introduced two datasets: UCCD and UCID. The information that we get from the chart can provide data support for future applications of ubiquitous information.

Though we have achieved good performance result on our dataset, there are some works need to be done in future:

(1) We have analyzed the 15 most commonly used types of charts, and there are many more types of charts that need to be researched.

(2) Besides, we just studied a 2D chart; but in reality, there are a lot of charts represented by 3D, which is more elegant.

(3) As we have introduced in Section 3, we use DB and CRNN to detect and recognize text, and we use PaddleOCR's [25] models. But to get high accuracy, we need to train chart-specific text models. There are a lot of scenes that in normal models may occur error: 1, text with axis tick; 2, text in the different color background; 3, text with lines interrupted; 4, text in switched degree.

(4) The charts that we have discussed so far are original images, which need not be corrected by perspective. But when we want to recognize a chart from a scanned paper, there are two problems to solve: first to detect the chart in the document and second to do some perspective corrections.

Data Availability

The data used to support this study are included within this article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the National Key Research and Development Program of China (2017YFB0503500) and the Youth Program of the National Natural Science Foundation of China (41801317).

References

[1] H. Zhang, Y. Li, Z. Lv, A. K. Sangaiah, and T. Huang, "A real-time and ubiquitous network attack detection based on deep belief network and support vector machine," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 790–799, 2020.

[2] X. Qin, Y. Luo, N. Tang, and G. Li, "Making data visualization more efficient and effective: a survey," *The VLDB Journal*, vol. 29, no. 1, pp. 93–117, 2020.

[3] Badam, S. Karthik, A. Mathisen, R. Rädle, C. N. Klokmose, and N. Elmqvist, "Vistrates: a component model for ubiquitous analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 586–596, 2018.

[4] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "Revision: automated classification, analysis, and redesign of chart images," in *Proceedings of the 24th annual ACM Symposium on User Interface Software and Technology*, Santa Barbara, CA, USA, October 2011.

[5] K. Dadhich, S. C. Daggubati, and J. Sreevalsan-Nair, "BarChartAnalyzer: digitizing images of bar charts," in *IMPROVE*, pp. 17–28, Springer Nature, Urdorf, Switzerland, 2021.

[6] D. Jung, W. Kim, H. Song et al., "ChartSense: interactive data extraction from chart images," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, Denver, CO, USA, May 2017.

[7] J. Fu, B. Zhu, W. Cui et al., "Chartem: reviving chart images with data embedding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 337–346, 2020.

[8] A. RohatgiWebPlotDigitizer, https://automeris.io/WebPlotDigitizer/ Version 4.1, 2018.

[9] J. E. Zhang, S. Nicole, A. Bezerianos, and F. Chevalier, "DataQuilt: extracting visual elements from images to craft pictorial visualizations," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, Honolulu HI USA, April 2020.

[10] P. Vougiouklis, L. Carr, and Elena Simperl, "Pie chart or pizza: identifying chart types and their Virality on twitter," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, pp. 694–704, 2020.

[11] J. Luo, Z. Li, J. Wang, and C.-Y. Lin, "ChartOCR: data extraction from charts images via a deep hybrid framework," in *Proceedings of the 2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, HI, USA, January 2021.

[12] A. Krizhevsky, I. Sutskever, and E. H. Geoffrey, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, 2012.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 2015.

[14] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016.

[16] "ImageNet," https://www.image-net.org.

[17] H. Law and D. Jia, "CornerNet: Detecting Objects as Paired Keypoints," *International Journal of Computer Vision*, vol. 128, 2018.

[18] A. Newell, K. Yang, and D. Jia, "Stacked Hourglass Networks for Human Pose Estimation," in *Proceedings of the 15th European Conference on Computer Vision*, Munich, Germany, September 2014.

[19] https://echarts.apache.org/.

[20] https://matplotlib.org/.

[21] M. Liao, Z. Wan, C. Yao, K. Chen, and B. Xiang, "Real-time Scene Text Detection with Differentiable Binarization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2019.

[22] B. Shi, B. Xiang, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, 2015.

[23] https://en.wikipedia.org/wiki/Point_in_polygon.

[24] J. Choi, S. Jung, D. Gun Park, J. Choo, and N. Elmqvist, "Visualizing for the non-visual: Enabling the visually impaired to use visualization," in *Proceedings of the Eurographics Conference on Visualization (EuroVis) 2019*, Porto, Portugal, June 2019.

[25] https://github.com/PaddlePaddle/PaddleOCR.