Hybrid Cascade Point Search Network for High Precision Bar Chart Component Detection

Junyu Luo
Pennstate University
Pennsylvania, USA
junyu@psu.edu

Jinpeng Wang
Microsoft Research
Beijing, China
jinpwa@microsoft.com

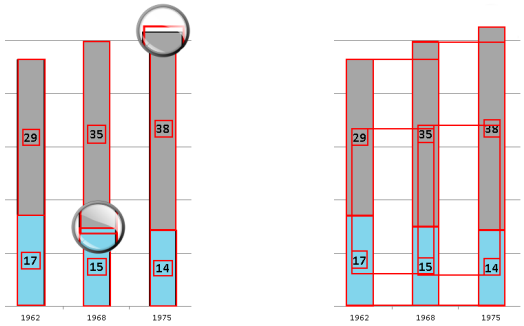
Chin-Yew Lin
Microsoft Research
Beijing, China
cyl@microsoft.com

***Abstract*—Bar charts are commonly used for data visualization. One common form of chart distribution is in its image form. To enable machine comprehension of chart images, precise detection of chart components in chart images is a critical step. Existing image object detection methods do not perform well in chart component detection which requires high boundary detection precision. And traditional rule-based approaches lack enough generalization ability. In order to address this problem, we design a novel two-stage component detection framework for bar charts that combines point-based and region-based ideas, by simulating the process that human creating bounding boxes for objects. The experiment on our labeled ChartDet dataset shows our method greatly improves the performance of chart object detection. We further extend our method to a general object detection task and get comparable performance.**

I. Introduction

Bar charts are common forms of human communication. They are ubiquitous and are easily found in news, web pages, company reports, scientific papers, etc. The automatic analysis of those data can bring us huge commercial values. In many cases, these charts are published as images. Humans can readily decode the underlying data in chart images, but machines have no way to understand them [1]. Some methods [1]–[3] have been proposed to address this challenge. Despite promising performance on their reported datasets, these previous work highly rely on manually defined features which are applicable only to certain types of bar chart. The diversity in chart designs and styles make rule-based chart component detection methods difficult to scale, e.g., adding texts or patterns to the bars of a bar chart always make hand-craft heuristics failed to detect the bars. Hence, a more general approach is desired to detect chart components to enhance machine interpretation of charts. The recent advanced natural object detection methods attract our attention. By casting chart component detection as an object detection problem, we can leverage the existing work in image object detection. However, due to the special precision requirement of chart components, the existing mainstream object detection methods can't solve this task easily.

For region-based methods, e.g, Faster-RCNN [4], the high precision detection of the boundary is hard to achieve. As shown in Fig. 1, most boundaries are shifted a few pixels. Although some works have been proposed to improve precision by eliminating error detection [5], [6] or better loss function



(a) Faster-RCNN (b) CornerNet

Fig. 1: Typical mistakes produced by existed object detection apporaches. (a) the common mistake of Faster-RCNN: most of boundaries shifted a few pixels. (b) the common mistake of CornerNet: miss-pairing different objects to one object, although the detected boundaries are precise.

[7], [8], the resolution problem still exists as described below. In order to capture the whole structure information of the target object, the region-based method usually needs to encode a large region. However, due to the high computational cost, the sampled resolution is usually very low, e.g, 7×7 in Ren [4]. It's not a surprise that region-based methods do not work well in the terms of precision.

For key-point based methods like CornerNet [9] are likely to mistakenly pair points belong to different objects as one object. CornerNet directly works on the highest resolution scale and has a good boundary detecting ability. However, the ConerNet only uses the local feature information around key points to pair the points. When a large number of similar and homogeneous objects appeared in one chart image. As shown in Fig. 1, the miss-pairing problems for CornerNet on chart images are severe. Some methods [10], [11] try to use the category information of the central point to eliminate the miss-pairing problem, but due to a large number of homogeneous components in one chart image, it is not a proper choice as we will show in the experimental results.

To solve the above-mentioned problems, we propose a new two-stage object detection framework that combines advantages from the both region and point-based methods. As shown in Fig. 2, our method imitates the process that human

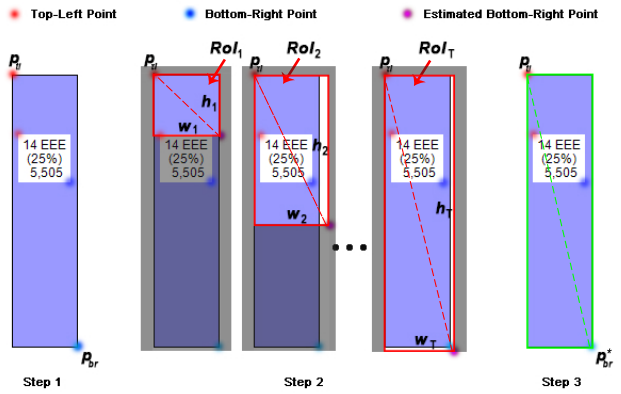


Fig. 2: The workflow of our method. **Step 1**: The detected top-left points and bottom-right points. **Step 2**: Starting with one top-left point, the main region for its corresponding object are estimated iteratively. **Step 3**: pair top-left points and bottom-right points based on the estimated regions.

creating bounding boxes for objects: the first step is to generate unpaired top-left and bottom-right points with a point-based method; Then, for each top-left point, a network is designed to search the main region for its corresponding object; at last, we pair top-left points and bottom-right points based on the regions for objects. It utilizes not only the ability of point-based methods for predicting precise key points of boundaries, but also the ability of region-based methods for predicting boundaries based on sampled regions. On the ChartDet data set collected by us which targets precise object detection, our method achieved the state-of-the-art on all metrics, especially when precise boundaries are required.

In summary, we make the following contributions:

- Propose a hybrid two-stage object detection framework that utilizes not only the ability of point-based methods for predicting key points of boundaries, but also the ability of region-based methods for predicting boundaries based on sampled regions.
- Achieve state-of-the-art on bar chart component detection, especially when precise boundaries are required.
- A new annotated data set for bar chart component detection with precise boundaries.

II. Related Work

A. Feature Based Chart Components Extraction

Since charts are highly structured, the feature based methods [1], [12], [13] for extracting chart components from chart images are the mainstream methods. Those methods highly rely on hand-crafted heuristics and low-level pre-defined features. They are efficient for the data with targeted design styles, but lack of generalization ability when minor changes are made to the input image. For example, adding texts or patterns to the bars always make such rules failed to detect the bars or the rules designed for detecting vertical bars are hard to work for horizontal bars. The styles and designs of charts change quickly today, which brings difficulty to the industrial use of those feature-based methods. More details can be found in Section V

B. Region-Based Object Detection

Region-based methods first sample a small region, then generate the bounding box based on the feature. According to the rounds of sampling, it can be divided into one-stage and two-stage methods. For one-stage methods, e.g. YOLO [14] and SSD [15], after the initial sample, the network needs to give out the final detection results directly, and the limitation of fixed sample region is the major disadvantage of such methods. Later, feature pyramid network (FPN) [16] solves this problem with multi-scale features. For two-stage methods like R-CNN [17], SPP [18], and Fast-RCNN [19] have an initial sampling procedure to dynamically generate sample regions, and the adaptive sample region greatly improves the precision. Recent work like deformable convolution network (DCN) [20] further improves the performance of these methods.

C. Point-Based Object Detection

The point-based methods are free of the limitation of the initial sample region, thus can achieve high precision in boundary detecting. For these methods, detection starts with key points, e.g., top-left points and bottom-right points in CornerNet [9] or peak points in ExtremeNet [11]. After the detection of the key points, a pairing algorithm is applied to pair the key points. Semantic embedding is used in CornerNet, and other methods like central heat map [11] or the combination of them [10] are also proposed. The grouping steps are performed outside the network, hence they can not be optimized through training and usually takes more than $O(n^2)$ in terms of time complexity. For dense scenes, e.g., tens of bars on a chart, the time cost is huge and impractical.

III. Method

The framework of Hybrid Cascade Pairing Network (HCPN) consists of two major networks. It is inspired by the process that when annotators are creating bounding boxes for objects. First, identify the top-left point of an object, then fit the top-left point and draw a bounding box that covers the main region of the current object, finally, adjust the boundary slightly to make it more precise. In our framework, as shown in Fig. 3, hourglass network (HG) [21] is adopted as the feature backbone network, then **Step 1** is executed by a fully convolutional key point proposal network, which aims to find the potential unpaired top-left and bottom-right points by predicting a heat map of points. In **Step 2**, given the initial top-left point on the input image, the object search network is designed to find the main region of the current object iteratively. In **Step 3**, the point pairing module pairs the top-left point with its corresponding bottom-right point produced by Step 1, based on the detected region of the object of Step 2. In this section, we will show more details about these steps.

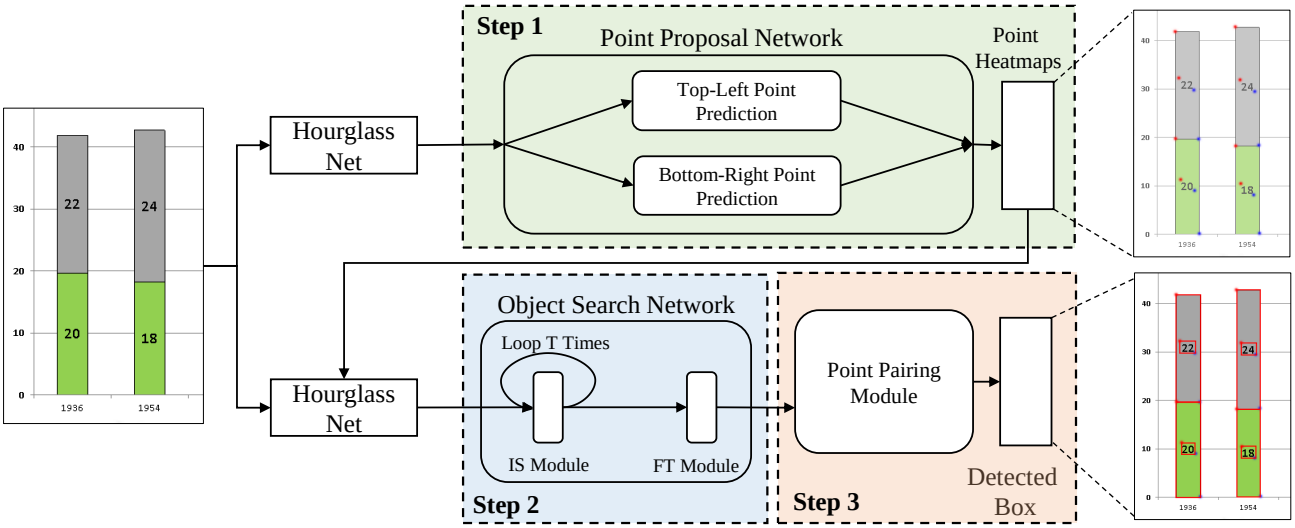


Fig. 3: A diagram of the hybrid cascade pairing network. Point Proposal Network (PPN), Object Search Network (OSN) and Point Pairing Module (PPM) work in a cascade order.

A. Step 1: Point Proposal Network (PPN)

In this paper, we adopt a simplified CornerNet [9] as our point proposal network. Follow the same setting, we predict two sets of heatmaps, i.e., the heatmap of top-left points which will be used as the initial search anchors in Step 2, and the heatmap of bottom-right points which will be paired with corresponding top-left points in Step 3. Each heatmap has C channels, where C denotes the number of categories, and with a size of $H \times W$. For each key point, there is one ground-truth positive location and those of negative locations are augmented with unnormalized Gaussians. Let y_{cij} be the ground-truth at location (i, j) for class c and $\hat{y}_{cij}$ be the corresponding predicted score. The loss of a heatmap is:

$$L_{heat} = \frac{-1}{N} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \begin{cases} (1-\hat{y}_{cij})^{\alpha} log(\hat{y}_{cij}), & \text{if } y_{cij} = 1 \\ (1-y_{cij})^{\beta} (\hat{y}_{cij})^{\alpha} log(1-\hat{y}_{cij}), & \text{else} \end{cases} \tag{1}$$

where α, β are the hyperparameters that control the contribution of each point (we follow the same settings with CornerNet), and N is the number of the target objects in the image. To eliminate the precision loss involved by downsampling, we follow CornerNet to add the offset loss L_{off}, then the loss for PPN is:

$$L_{\text{PPN}} = L_{heat} + L_{off} \tag{2}$$

Solving the above equation we get two heatmaps for top-left points and bottom-right points. For each heatmap, we keep the top-k points as candidate points according to their predicted scores $\hat{y}_{cij}$, i.e., a set of top-left points $\{p_{tl}\}$ which will be used as initial search anchors in Step 2 and a set of bottom-right points $\{p_{br}\}$ which will be paired with corresponding top-left points in Step 3. For each point, $p.score$ indicates its score and $p.label$ indicates its class. To enhance the representation of the input image, we also integrate these two heatmaps with the original input image.

B. Step 2: Object Search Network (OSN)

With the enhanced input image and the predicted top-left points $\{p_{tl}\}$, the target of OSN is to find the main region for each top-left point.

1) Iterative Search Module (ISM): Starting from each top-left point p_{tl} at location $(p_{tl}.i, p_{tl}.j)$, ISM searches the main region of the current object $p_{tl}.lable$ iteratively. At iteration t, $(p_{tl}.w_t, p_{tl}.h_t)$ is the predicted width and height of the object[1]. ISM makes predictions based on three factors: a) RoI_t^{ISM}: the image features of the current region started from $(p_{tl}.i, p_{tl}.j)$ with size $(p_{tl}.w_t, p_{tl}.h_t)$. RoIAlign [22] algorithm is used to sample the RoI into a fixed size feature $RoI_t^{\text{ISM}} \in \mathbb{R}^{H_{\text{ISM}} \times W_{\text{ISM}} \times C_f}$, where $W_{\text{ISM}}, H_{\text{ISM}}$ are the sampled resolutions and C_f is the number of channels of the feature layer; b) $p_{tl}.label$: the label information of p_{tl} which is critical because it can serve as an indicator to guide the network to focus on label related objects. The label of top-left point is represented in one-hot format $p_{tl}.label \in \mathbb{R}^{1 \times C}$; c) $Scale_t$: the scale information of sampled region at the current interation t. Specifically, $Scale_t = (\frac{p_{tl}.w_t}{W_{\text{ISM}}}, \frac{p_{tl}.h_t}{H_{\text{ISM}}})$. The outputs of ISM at interaction t are shown as the following equations:

$$(f_w, f_h) = \text{ISM}\big(RoI_t^{\text{ISM}}, p_{tl}.label, Scale_t\big) \tag{3}$$

$$p_{tl}.w_{t+1} = p_{tl}.w_t \times f_w, \qquad p_{tl}.h_{t+1} = p_{tl}.h_t \times f_h \tag{4}$$

Notice that this network predicts the relative ratios (f_w, f_h) instead of directly predicting the size of region $(p_{tl}.w_{t+1}, p_{tl}.h_{t+1})$, which makes the prediction of network scale irrelevant. Hence, the module can share parameters from

[1] $(p_{tl}.w_0, p_{tl}.h_0) = (W_{\text{ISM}}, H_{\text{ISM}})$ for initialization.

different iterations which with different scales. And the loss for ISM is:

$$L_{\text{ISM}} = \frac{1}{T}\sum_{t=1}^{T}\Big(\text{RegLos}\big(\log(f_w), \log(\frac{p_{tl}.w_g}{p_{tl}.w_t})\big) + \text{RegLos}\big(\log(f_h), \log(\frac{p_{tl}.h_g}{p_{tl}.h_t})\big)\Big) \quad (5)$$

where T is the number of iterations, $p_{tl}.w_g, p_{tl}.h_g$ are the ground-truth size of region, and RegLos is the regression loss function where smooth L1 loss is selected. In training, p_{tl} and $p_{tl}.label$ are provided by ground-truth annotations, while in testing, they are predicted by PPN[2].

2) Fine Truing Module (FTM): We add the FTM to further refine the predictions by ISM. It is different from ISM, as FTM focuses on fine-tuning the predicted region while ISM focuses on finding targeted regions in a few iterations. This is also inspired by [23] where a sequence of detectors trained with increasing IoU thresholds to improve performance. Therefore it's better to model FTM as a separated module that focus on the final stage where the IoU threshold is high. Specifically, FTM predicts a linear offset to adjust the previous estimated region from $(p_{tl}.w_T, p_{tl}.h_T)$ to $(p_{tl}.w, p_{tl}.h)$:

$$(d_w, d_h) = \text{FTM}(RoI^{\text{FTM}}, p_{tl}.label, Scale_T) \quad (6)$$

$$p_{tl}.w = p_{tl}.w_T \times (1 + d_w) \quad (7)$$

$$p_{tl}.h = p_{tl}.h_T \times (1 + d_h) \quad (8)$$

where $RoI^{\text{FTM}} \in \mathbb{R}^{H_{\text{FTM}} \times W_{\text{FTM}} \times C_f}$ and the sampled resolution is also higher compared with those for ISM. Again, Eq. 6 predicts the relative ratios instead of directly predicting the size of region which makes the prediction of network scale irrelevant. The loss of FTM is:

$$L_{\text{FTM}} = \text{RegLos}\big(d_w, \frac{p_{tl}.w_g - p_{tl}.w_T}{p_{tl}.w_T}\big) + \text{RegLos}\big(d_h, \frac{p_{tl}.h_g - p_{tl}.h_T}{p_{tl}.h_T}\big) \quad (9)$$

The final training loss of one top-left point for OSN is:

$$L_{\text{OSN}} = L_{\text{ISM}} + L_{\text{FTM}} \quad (10)$$

C. Step 3: Point Pairing Module (PPM)

The accuracy of the region estimated by Step 2 is not precise enough, since OSN is also a region based method, hence face the same drawback due to the limitation of sampled resolution as illustrated in Introduction. To address this problem, we propose to pair the predicted top-left point based on the corresponding bottom-right points predicted by Step 1 and the regions predicted by Step 2, as shown in Algorithm 1. In Line 3, we first select the subset from bottom-right points $\{p_{br}\}$ where p_{br} close to p_{obj} the bottom-right point of the predicted region and has the same label with p_{tl}. For each candidate p_{br}, in Line 5, we calculate the IoU between bounding box constructed by (p_{tl}, p_{obj}) and that by p_{tl}, p_{br}. After that, if the IoU and the weighted score in Line 7 reach corresponding

[2]We expand the ground-truth bbox with a fix exp pixel for OSN.

Algorithm 1 Point Pairing Algorithm

Input: top-left point p_{tl}, top-k bottom-right points $\{p_{br}\}$, size of predicted region $(p_{tl}.w, p_{tl}.h)$, threshold of IoU T_{IoU}, threshold of the score T_{score}, ratio of candidate region γ

Output: the paired bottom-right point p^*_{br}

1: initial the max score $S_{max} = 0$
2: $p_{obj} = (p_{tl}.i + p_{tl}.w,\ p_{tl}.j + p_{tl}.h)$
3: $\{p_{br}\}'$ = **select** p_{br} **from** $\{p_{br}\}$
where p_{br} **in** $(p_{obj}.i \pm \gamma \cdot p_{tl}.w,\ p_{tl}.j \pm \gamma \cdot p_{tl}.h)$
and $p_{br}.label == p_{tl}.label$
4: **for** $p_{br} \in \{p_{br}\}'$ **do**
5: $S_{IoU} = \text{IoU}\big(\text{bbox}(p_{tl}, p_{obj}), \text{bbox}(p_{tl}, p_{br})\big)$
6: **if** $S_{IoU} > T_{IoU}$ **then**
7: $S_{cur} = S_{IoU} \times p_{br}.score$
8: **if** $S_{cur} > S_{max}$ and $S_{cur} > T_{score}$ **then**
9: $S_{max} = S_{cur}$
10: $p^*_{br} = p_{br}$
11: **end if**
12: **end if**
13: **end for**

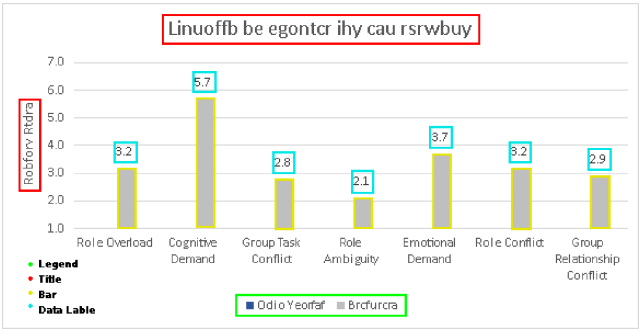


Fig. 4: An example image from ChartDet val set.

thresholds, we update the answer of the bottom-right point which should pairs with input p_{tl} in Line 10. By using Quad Tree index [24] the search of candidate points $\{p_{br}\}'$ takes $O(log4n)$, and the algorithmic complexity for fixing n points is $O(nlog4n)$.

IV. ChartDet Data Set

To evaluate the ability of object detection under a high precision setting, a dataset contains boundaries with pixel-level accuracy is required. The existing object detection datasets cannot satisfy this request as most of them are annotated by human annotators who hard to annotate objects with pixel-level accuracy. In this paper, we present the ChartDet dataset which is a labeled chart image dataset for high precision object detection. On one hand, the boundaries and labels are exported from editable charts with computer API automatically which are precise. On the other hand, the task of chart object detection has application values, e.g., extracting semantic information or data values from chart images with high precision object detection, to enable machine readability for charts or

visualizing data for visually impaired users. In addition, the ability to detect chart objects can be easily adapted to other manual created image data like tables, posters, or web pages.

A. Dataset Collection

We first crawl public available Excel sheets from Internet. For each chart in Excel sheets, we capture the image of chart with Excel APIs and export 4 categories of boundaries, i.e., Legend, Title, Bar and Data Label as shown in Fig. 4.We compared our data set with the general object detection data set COCO [25] in Table I.

TABLE I: Dataset statistics. #cat. is the number of categories.

Data Set	train	val	test	#cat.	#objects per image	precise boundary
COCO	118,287	5,000	20,288	81	7	×
ChartDet	158,140	6,121	6,262	4	25	✓

V. Experimental Results

A. Implementation Details and Parameters

All experiments are conducted in the same environment with 4 P100 GPUs. Batch size is chosen by the maximum running size, and Soft-NMS is applied for all final results. For OSN, DCN [20] block is included to align the features. The selection of hyperparameters is confirmed by val set for HCPN and the baseline methods. For PPN, we follow the same setting of CornerNet: training iteration=100,000; batch size=26. For OSN: training iteration=40,000; batch size=26; learning rate=1e-4; the exp=3 pixels; iteration time T is set to 3. For sample resolution, the ISM module is 20×20 and the FTM module is 40×40. For PPM, γ is set to 0.3, T_{IoU} is set to 0.5 and T_{score} is set to 0.25.

B. Compared Methods

For **region approaches**, we use Faster-RCNN [4], Retinanet [26] and the state-of-the-art Cascade-RCNN [23]. For **point approaches**, we select CornerNet [9] and CenterNet [10]. For Faster-RCNN, Cascade-RCNN, and Retinanet, the batch size is set to 20, and training iteration is set to 14 epochs of data set. Learning rates are set to their default best learning rate. SGD optimizer is used and the learning rate is reduced 10 times at 10 and 12 epoch to guarantee the network can converge to its best fit. No pre-training for the backbone networks for fairness. For CornerNet and CenterNet, we follow the same set of OSN. We include DCN blocks for all methods that contain region sampling, and FPN is used to provide multi-scale features. For **rule-based baseline**, we select Revison [12][3]. For **our approaches**, HCPN is the full model which described in Section III, and HCPN w/o PPM is our ablation method without the point pairing module.

[3]ReVision only includes the extraction of bar components, so we only compare it on this category.

C. Evaluation Metric

We adopt the COCO Average Precision (AP) across IoU thresholds from 0.5 to 0.95 with an interval of 0.05. We also report the AP scores from 0.5 to 0.95 to measure the high precision boundary detecting ability. In the meanwhile, we conduct additional studies to analyze the grouping ability and the effectiveness of different steps in OSN.

D. Overall Performance

Table II shows the performance for all methods on the test set of ChartDet, and Fig. 5 visualizes their examples. We can see that the rule-based method Revision cannot detect any stacked elements because it doesn't have related rules. For other charts, Revision is always disturbed by texts or patterns and work poorly. In terms of AP, Cascade-RCNN, CornerNet, and CenterNet are comparable, while our HCPN achieves the best performance among all methods, i.e., HCPN outperforms CenterNet with 6.0% improvement on AP.

We further zoom in the results with another two metrics to analyze the problems discussed in Section I: $AP_{0.5}$ which mainly reflects the ability to detect the general target objects and isn't sensitivity to the precision of boundary; and $AP_{0.95}$ which is sensitivity to the precision of boundary in addition to the ability to find the target objects. We can see that Faster-RCNN works well on $AP_{0.5}$ while poor on $AP_{0.95}$, and CornerNet has a relatively low score in terms of $AP_{0.5}$, despite CornerNet has a good AP score. This confirms with our finding in Section I that Faster-RCNN suffers from the problem of precision and CornerNet easy to make mistakes when pairing top-left points and bottom-right points. CenterNet tries to solve this problem by adding a center heatmap, but due to a large number of homogeneous components in chart images, the improvement of CenterNet to CornerNet is small. Moreover, CenterNet also drops on $AP_{0.95}$, which shows that its precision is hurt. The above results show that our method effectively combines the advantage of region methods and point-based methods by using region structure information to group points and using key points to keeps the precision of boundary, and Fig. 5 further shows that the problems of miss-pairing and boundary missing are solved effectively. In the meanwhile, by comparing HCPN with HCPN w/o PPM, we find that PPM effectively improved the performance on $AP_{0.95}$ by improving the precision of the bottom-right points, which also proves our assumption of precision loss in region sampling.

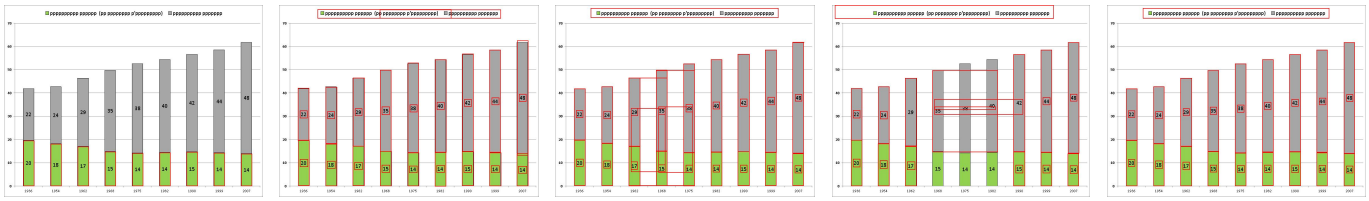
E. Error Analysis

Fig. 6.a, Fig. 6.c and Fig. 6.e show the error analysis of the rule-based method Revision. We can see that Revision cannot handle stacked charts (e.g., Fig. 6.a) and always be disrupted by the texts (e.g., Fig. 6.a and Fig. 6.e) or the background patterns (e.g., Fig. 6.c).

Fig. 6.g, Fig. 6.i and Fig. 6.k show the error analysis of Faster-RCNN for chart component detection. We can see that Faster-RCNN in Fig. 6.g, the boundaries shifted a few pixels, and in Fig. 6.i and Fig. 6.k, Faster-RCNN is struggle to assign correct bounding boxes for larger components.

TABLE II: The overall performance of chart component detection on ChartDet test set. $AP_{0.5}$ mainly reflects the ability of detecting the general target objects and isn't sensitivity to the precision of boundary; $AP_{0.95}$ is sensitivity to the precision of boundary in addition to the ability of finding the target objects.

Method	AP	$AP_{0.5}$	$AP_{0.75}$	$AP_{0.8}$	$AP_{0.85}$	$AP_{0.9}$	$AP_{0.95}$
Retinanet 101	0.459	0.729	0.497	0.389	0.253	0.110	0.012
Faster-RCNN 101	0.580	0.805	0.664	0.578	0.434	0.233	0.056
Cascade-RCNN 101	0.647	0.831	0.723	0.660	0.552	0.375	0.149
CornerNet	0.646	0.783	0.717	0.674	0.587	0.429	0.225
CenterNet	0.666	0.820	0.742	0.685	0.592	0.429	0.205
HCPN w/o PPM	0.697	**0.870**	0.775	0.714	0.610	0.429	0.222
HCPN	**0.706**	0.868	**0.778**	**0.723**	**0.623**	**0.457**	**0.261**
HCPN (Bar Only)	0.810	0.934	0.873	0.837	0.757	0.635	0.397
Revision (Bar Only)	0.330	0.598	0.316	0.217	0.112	0.032	0.002



(a) Revision (b) Faster-RCNN (c) CornerNet (d) CenterNet (e) HCPN

Fig. 5: Visualization examples on ChartDet val set (IoU Threshold 0.4). a) The rule-based method Revision cannot handle stacked bars; b) the boundaries detected by Faster-RCNN shifted a few pielxs; c) CornerNet miss-pairs different objects to one object; d) CenterNet alleviates the miss-pairing problem; e) Our HCPN doesn't have the above problems.

TABLE III: The ability of pairing top-left points and bottom-right points for CornerNet and HCPN. w/ GT masks the mistakes on predicting key points by substitute the top-left points, bottom-right points and object categories with corresponding ground-truths.

$AP_{0.5}$	w/ GT	w/o GT
CornerNet	0.903	0.783
HCPN	0.956	0.868

Fig. 6.m and Fig. 6.o show the error analysis of CornerNet for chart component detection. We can see that CornerNet miss-pairing different objects to one object, although the detected boundaries are precise, especially when detecting the chart images with crowd objects.

F. Pairing Ability Study

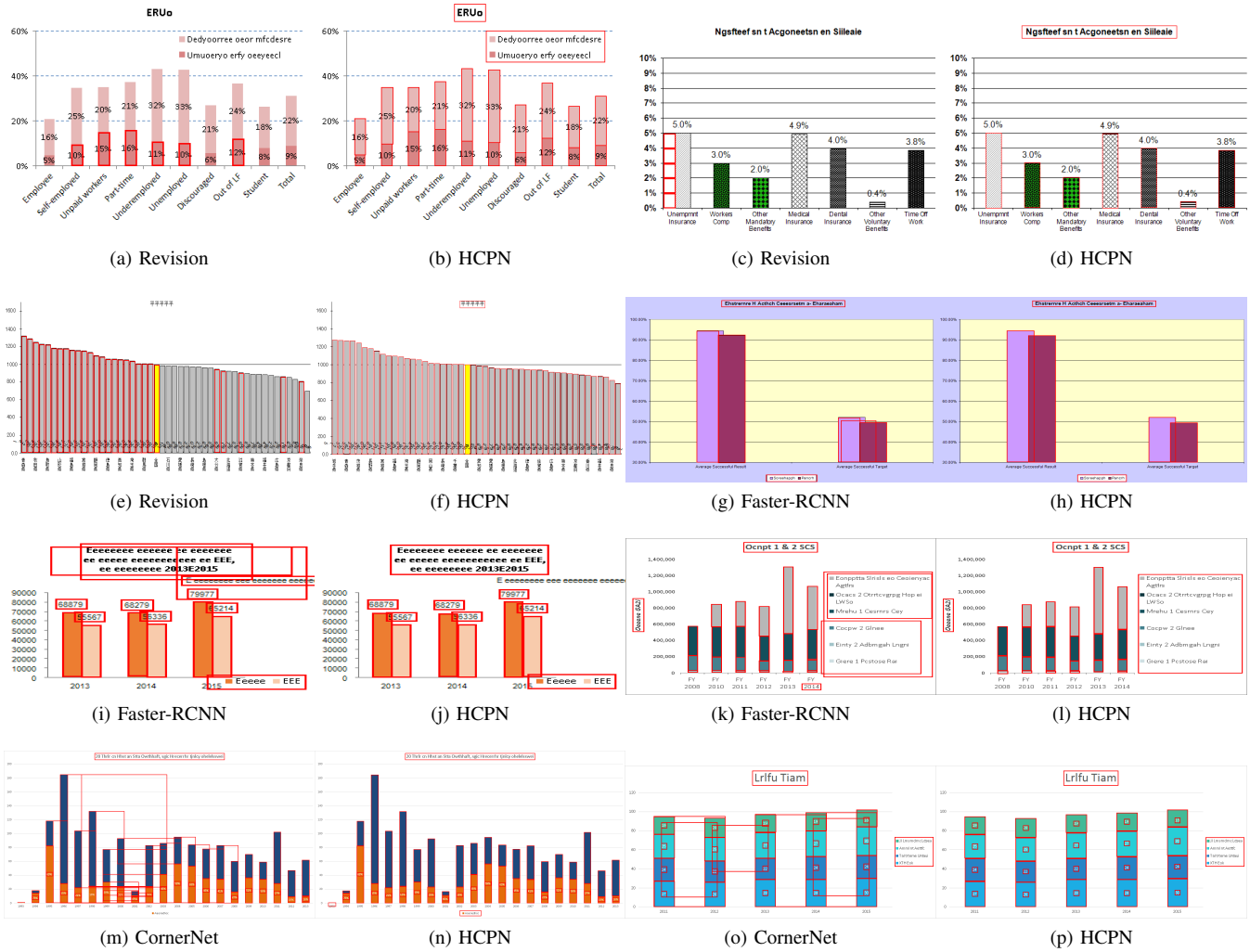
To analyze the pairing ability for top-left points and bottom-right points, we mask the mistakes in predicting key points by substitute the top-left points, bottom-right points, and object categories with corresponding ground-truths. Here, we also choose to report $AP_{0.5}$, as it mainly reflects the ability to find the general target objects correctly. From Table III, we can see that, with ground-truth information, both methods got significant improvements, and HCPN achieve 0.956 on $AP_{0.5}$ which is nearly perfect while that for CornerNet is only 0.903. This result proves the improvement of our approach on point pairing ability. It also shows that the major bottleneck of HCPN is the point proposal network.

TABLE IV: Model inference time and the corresponding chart component detection performance for HCPN w/o PPM.

#Iteration	1	2	3	4	CenterNet	Revision
AP	0.580	0.683	0.697	0.700	0.666	-
$AP_{0.5}$	0.813	0.867	0.870	0.870	0.820	-
FPS	2.99	2.85	2.82	2.74	3.26	0.05

G. OSN Search Step Analysis

Fig. 7 visualizes the boundaries under each search step of OSN, we can see that ISM first make a rough estimation of the target object and then adjust the prediction according to the new region information, and FTM makes the final decisions with more precise boundaries. We also report the AP scores and inference time under different count of search iterations T for ISM in Table IV. As we can see, the iteration procedure of ISM is necessary: the initial steps of ISM usually cannot cover the ground-truth and have low AP scores; more iterations bring higher precision; and the improvement is tiny after the third iteration. With more iterations, the cost of inference also goes up, but the FPS is still acceptable under the high precision requirement. Compared with CenterNet, despite CenterNet is a one-stage method, the FPS of CenterNet is closed to ours, which is due to its high complexity of the grouping algorithm. In further work, with shared backbone network acceleration, the time consuming of our method can be further reduced.



(a) Revision (b) HCPN (c) Revision (d) HCPN

(e) Revision (f) HCPN (g) Faster-RCNN (h) HCPN

(i) Faster-RCNN (j) HCPN (k) Faster-RCNN (l) HCPN

(m) CornerNet (n) HCPN (o) CornerNet (p) HCPN

Fig. 6: Error analysis. For the rule-based method Revision, in a) it failed to detect the bar components for stacked charts; in a) and e) it is disrupted by the texts; in c) it is disrupted by the background pattern. For Faster-RCNN, in g) the boundaries shifted a few pixels; in i) and k), it is struggle to assign correct bounding boxes for larger components. For CornerNet, in m) and o) it miss-pairing different objects to one object, although the detected boundaries are precise, especially when detecting the chart images with crowd objects.

TABLE V: Performance of object detection on COCO val set

Method	AP	$AP_{0.5}$	$AP_{0.75}$	AP_S	AP_M	AP_L
HCPN	0.390	0.556	0.409	0.205	0.417	0.550
CornerNet	0.391	0.546	0.414	0.193	0.416	0.546

In addition, we also report the inference time of Revision here. Revision can run inference at only 0.05 FPS while our HCPN can run at 2.82 FPS, which makes Revision far from commercialization.

H. Performance on General Object Detection

We also report the performance of HCPN on COCO data set in Table V. As we can see, compared with its backbone CornerNet, HCPN gets a litter improvement, especially under a lower IoU setting, where the PSN step can eliminate the impossible point grouping results. However, HCPN is designed for dense scenes that include multiple homogeneous objects at the same image, which is quite different from COCO where the average number of objects in a per image is less than 7. In addition, the miss-pairing problem on the COCO dataset is not significant, as most of the objects in one image usually belong to different categories. It's not surprising that the improvement of HCPN on general object detection is limited.

VI. Conclusion

We make the initial attempt on charts and offer the ChartDet dataset, a new benchmark of such human-created chart images, labeled with precise bounding boxes. Our tests on this benchmark show that such images challenge state-of-the-art detectors. The performance of modern object detectors

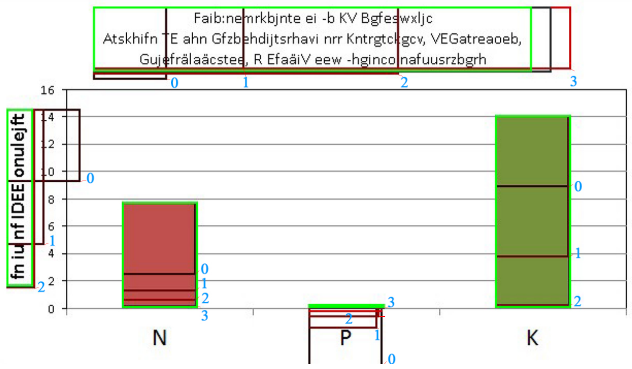


Fig. 7: Iteration visualization of OSN. Boxes from dark to bright with blue numbers represent the search steps of ISM and the green one represents the FTM.

on bar chart component detection is remarkable yet still limited. To address these challenges, we presented HCPN, a new framework for precise object detection. The experiments proved that our method effectively combining the strengthens of the region and point-based methods on chart component detection tasks. For future work, we will further expand this work for more types of charts and visualized data.

REFERENCES

[1] J. Poco and J. Heer, "Reverse-engineering visualizations: Recovering visual encodings from chart images," in *Computer Graphics Forum*, 2017, pp. 353–363.

[2] R. A. Al-Zaidy and C. L. Giles, "A machine learning approach for semantic structuring of scientific charts in scholarly documents," in *Twenty-Ninth IAAI Conference*, 2017.

[3] J. Poco, A. Mayhua, and J. Heer, "Extracting and retargeting color mappings from bitmap images of visualizations," *IEEE Transactions on Visualization & Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[4] S. Ren, K. He, R. Girshick, and S. Jian, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.

[5] E. Goldman, R. Herzig, A. Eisenschtat, J. Goldberger, and T. Hassner, "Precise detection in densely packed scenes," in *CVPR*, 2019.

[6] A. Neubeck and L. V. Gool, "Efficient non-maximum suppression," in *International Conference on Pattern Recognition*, 2006.

[7] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *CVPR*, 2019, pp. 658–666.

[8] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding box regression with uncertainty for accurate object detection," *arXiv preprint arXiv:1809.08545*, 2018.

[9] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 734–750.

[10] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," *arXiv preprint arXiv:1904.08189*, 2019.

[11] X. Zhou, J. Zhuo, and P. Krahenbuhl, "Bottom-up object detection by grouping extreme and center points," in *CVPR*, 2019, pp. 850–859.

[12] M. Savva, N. Kong, A. Chhajta, F. F. Li, M. Agrawala, and J. Heer, "Revision: automated classification, analysis and redesign of chart images," in *Acm Symposium on User Interface Software & Technology*, 2011.

[13] J. Gao, Z. Yin, and K. E. Barner, "View: Visual information extraction widget for improving chart images accessibility," in *IEEE International Conference on Image Processing*, 2013.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[16] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. PP, no. 99, pp. 2999–3007, 2017.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision & Pattern Recognition*, 2014.

[18] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 37, no. 9, pp. 1904–16, 2014.

[19] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[20] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," in *CVPR*, 2019, pp. 9308–9316.

[21] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*. Springer, 2016, pp. 483–499.

[22] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2017.

[23] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154–6162.

[24] H. Samet, "The quadtree and related hierarchical data structures," *Acm Computing Surveys*, vol. 16, no. 2, pp. 187–260, 1984.

[25] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," 2014.

[26] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.