Visual Style Extraction from Chart Images for Chart Restyling

Danqing Huang[1], Jinpeng Wang[1], Guoxin Wang[2], Chin-Yew Lin[1]
[1] Microsoft Research, Beijing, China [2] Microsoft, Redmond, United States
{dahua, jinpwa, guow, cyl}@microsoft.com

***Abstract*—Creating a good looking chart for better visualization is time consuming. There are plenty of well-designed charts on the Web, which are ideal references for imitation of chart style. However, stored as bitmap images, reference charts have hinder machine interpretation of style settings and thus difficult to be directly applied.**

In this paper, we extract visual properties from reference chart images as style templates to restyle charts. We first construct a large-scale dataset of 187,059 chart images from real world data, labeled with predefined visual property values. Then we introduce an end-to-end learning network to extract the properties based on two image-encoding approaches. Furthermore, in order to capture spatial relationships of chart objects, which are crucial in solving the task, we propose a novel positional encoding method to integrate clues of relative positions between objects. Experimental results show that our model significantly outperforms baseline models. By adding positional features, our model achieves better performance. Finally, we present the application for chart restyling based on our model.

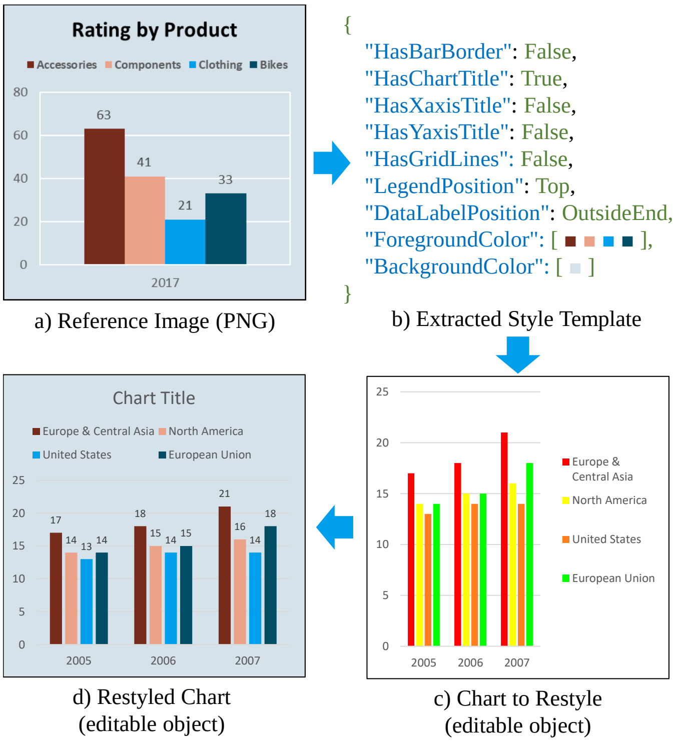
***Index Terms*—Chart Restyling, Chart Style Extraction**

I. Introduction

A chart is a graphical representation of data, and is commonly used in news articles, scientific papers, and on the Web. Charts with well-designed styles can not only gain more attentions, but also make data presentation more clearly. However, creating a well-designed chart takes time effort and requires designing skills. There are many good looking charts on the Web. Imitating styles from these reference charts is an efficient option. However, stored as bitmap images, reference charts have hinder machine interpretation of visual settings, and thus cannot be directly applied.

In this paper, we aim to recover the style template from a reference image. As shown in Fig. 1b, we first define the style template with nine common visual properties used in Excel[1]. By automatically extracting visual properties from reference images, we can collect large amount of well-designed templates, which has great potential applications, e.g., chart analysis or retrieval. In this paper, we show an example application for chart restyling. As shown in Fig. 1, the extracted style template is applied to a chart to imitate the style from the reference image.

In previous works [1]–[3] of chart proprieties extraction, data content such as texts or data values are extracted from chart images. Different from previous setting, we aim to recover style templates by extracting visual properties from chart images. Beyond text localization and object detection used in previous works, our extraction process requires inference among chart objects with their spatial relations. For example, in Fig. 1a), to predict the value of DataLabelPosition (i.e., the position of the data labels relative to the bars), the model needs to target to data labels and bars, and be aware that the data labels are on the top of the bars and therefore labeled as "OutsideEnd". This task is non-trivial, as it requires not only the ability of object detection but also the ability of spatial inference.



a) Reference Image (PNG)

b) Extracted Style Template

d) Restyled Chart (editable object)

c) Chart to Restyle (editable object)

Fig. 1: The workflow of chart restyling. To restyle c) with the style of a), the style template is first extracted from the reference image, then is applied to an editable chart object to render the restyled chart. c) and d) are editable objects in editor like Microsoft Excel, so their properties can be changed by APIs.

To accomplish this task, we introduce a neural-based end-to-end learning model, which adopts two encoding approaches for image representation, given the raw image and chart objects

[1] https://en.wikipedia.org/wiki/Microsoft_Excel

as input respectively to capture different aspects of features. To enhance the encoder with the ability of spatial inference, we propose a novel approach for feature encoding of relative position. Specifically, for each object, its neighbouring objects are encoded as context information for representation. In addition, as most publicly available chart datasets are synthetic or with limited size, to facilitate the research on this domain, we construct a large-scale dataset of 187,059 bar chart images based on real-world data, labeled with visual properties and chart object bounding boxes.

Our experimental results show that our model achieves promising results. By adding the positional feature, our model reduces the position-related errors and leads to improved performance.

The main contributions of this paper are:

- We provide an end-to-end solution for extracting visual properties from chart images.
- We propose a novel encoding approach to integrate relative position clues, which is crucial for spatial relation inference and visual understanding.
- We construct a large-scale dataset containing 187,059 chart images, labeled with visual properties and chart object bounding boxes. The dataset will be publicly released.
- We present an application of our model: extract visual properties from reference image as style template for chart redesign.

The remainder of this paper is organized as follows. Section II introduces the dataset. Section III describes in detail the design of the proposed approach, and Section IV is the experiments. Finally, we discuss related work in Section V and make our conclusions in Section VI and discuss future directions.

II. Dataset

In this section, we introduce our chart image dataset.

A. Dataset Property

Each chart image in the dataset is labeled with:

- **Visual Properties.** Definitions of each visual property are given in Table I. Furthermore, our dataset provides extra properties such as text and data contents.
- **Bounding Boxes.** We define 8 object classes in the chart: background, plot area, bar, x-axis title, y-axis title, chart title, legend and data label.

B. Construction & Statistics

Manual annotation of chart images is time consuming. Instead of collecting chart images and manually labeling them, we first crawl large amounts of Microsoft Excel files (.xlsx) from the Web, which already contain meta information. We currently only consider bar charts in these files and proceed the following steps to construct the dataset:

- Annotation Extraction. Stored as Excel format, the charts contain the information of visual properties and chart objects that can be directly used as annotation. We use a third-party tool to parse the Excel files to retrieve the meta information.
- Text Anonymization. For data compliance consideration, we replace texts in the charts with randomly generated strings.
- Chart Image Generation. Finally, we export the charts as bitmap images, which will be used as the input of our model.

Statistics. Following the above procedures, we collect **187,059** chart images in total. We randomly split the data into 173,168 / 6,928 / 6,963 charts in the training, development and testing sets respectively. We show the visual property statistics with `LegendPosition` in Table II, and `DataLabelPosition` in Table III. From these tables, we can see the distributions are highly skewed.

Dataset Comparison. It can be seen in Table IV that existing datasets are either sythentic or with limited size. The largest ones (FigureQA and DVQA) are targeted for Visual Question Answering (VQA) without visual property annotation, as well as synthetic with low style variations. Being compared, our dataset provides rich annotation of chart images from large-scale real-world data.

III. Approach

Given a chart image x_0, our goal is to extract its visual properties. Model overview is shown in Fig. 2.

We describe details of each component in this section.

A. Image Encoder

Our image encoder is based on DenseNets [6]. It consists of several major components as briefly described in the following.

Dense Blocks. A dense block is a group of non-linear transformation functions:

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]) \tag{1}$$

Specifically, x_l denotes the output of the l^{th} layer. The dense block has access to the feature maps its produced in its proceeding layers $0, ..., l-1$ as input. Following [6], H_l is composed of three consecutive operations: batch normalization, rectified linear unit and a convolution.

Transition Layers. Since the inputs of Eq. 1 have different feature-map sizes, transition layers are used between adjacent dense blocks to change the feature-map sizes via convolution and pooling.

Growth Rate g. Let H_l produces g feature maps, it follows that the l^{th} layer has $g_0 + g \times (l-1)$ input feature maps, where g_0 is the number of channels in the input image. This hyper-parameter regulates how much new information each new layer contributes to the global state.

After l layers of transformation, the image encoder outputs a hidden representation x_l for the image.

B. Object Encoder

Given a set of objects in the input image, the object encoder represents the image dynamically according to different visual properties.

TABLE I: Chart visual properties definition.

Visual Property	Labels	Description
`HasBarBorder`	True, False	The bars in the chart has borders or not.
`HasChartTitle`	True, False	The chart has title or not.
`HasXaxisTitle`	True, False	The x-axis has title or not.
`HasYaxisTitle`	True, False	The y-axis has title or not.
`HasGridLines`	True, False	The chart has grid lines or not.
`LegendPosition`	None, Top, Right, Bottom, Top	Legend position in the chart.
`DataLabelPosition`	None, OutsideEnd, InsideEnd, InsideBase, Center	Data label position relative to the bar.
`ForegroundColor`	#000000-#ffffff	Colors of the bars in a chart.
`BackgroundColor`	#000000-#ffffff	The background color of a chart.

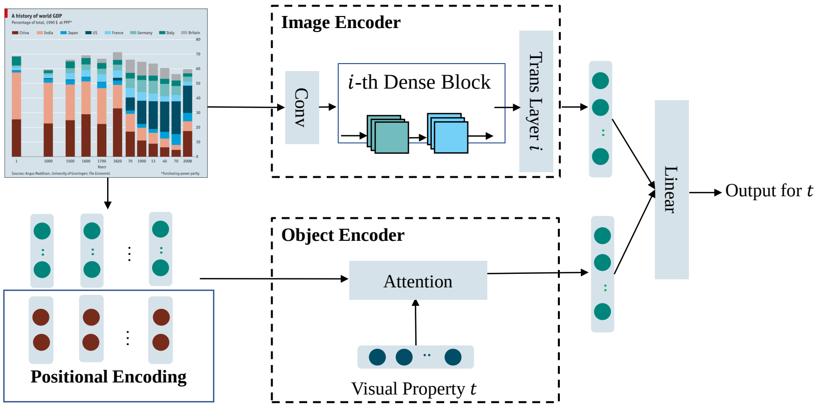


Fig. 2: Model overview.

TABLE II: `LegendPosition` class distribution (%) in train, dev and test.

`LegendPosition`	Train	Dev	Test
`None`	28.12	26.78	27.79
`Top`	13.43	15.91	14.82
`Bottom`	26.73	26.35	25.83
`Left`	1.87	2.76	2.41
`Right`	29.84	28.21	29.14

TABLE III: `DataLabelPosition` class distribution (%) in train, dev and test.

`DataLabelPosition`	Train	Dev	Test
`None`	69.92	63.59	63.52
`InsideEnd`	0.93	1.93	2.08
`OutsideEnd`	22.57	25.02	24.46
`InsideBase`	0.36	0.53	0.79
`Center`	6.23	8.92	9.16

Object Inputs. Objects are represented as region features. The input is K region features $V = \{v_1, \dots v_K\}$ $(v_k \in \mathcal{R}^D)$. In this work, we follow [7] to train a Faster-RCNN [8] on our dataset for chart object detection, select all regions with class detection probability exceeds a certain threshold, and use the final feature layer output for each region as the input.

Visual Property Embedding. Visual properties are embedded as:

$$\{\phi(sty_1), \phi(sty_2), \dots, \phi(sty_T)\} \tag{2}$$

where ϕ maps the visual property sty_t to a fixed dimensional vector.

Visual Property Aware Attention. Attention mechanism [7], [9] has been widely used in many tasks as it can explicitly weight inputs with importance. Our object encoder applies a soft attention over all the input objects, and weight objects according to different visual properties. For each visual property as a querying vector, an attention function is applied to attend over the objects:

$$e_{tk} = w_a^T \tanh(W_{va} v_k + W_{sa} \phi(sty_t) + b_{attn}) \tag{3}$$

$$a_{tk} = \frac{\exp(e_{tk})}{\sum_{k'=1}^{K} \exp(e_{tk'})} \tag{4}$$

$$c_t = \sum_{k=1}^{K} a_{tk} h_k \tag{5}$$

Intuitively, a_{tk} defines the probability distribution of attention over the input objects. They are computed from the unnormalized attention score e_{tk}. c_t is the context vector for visual property t, which is the weighted sum of the input region features.

C. Positional Encoding

Some visual properties are position-sensitive, which require spatial relationship inference with other objects to predict the

TABLE IV: Comparison of different datasets.

Dataset	Source	# Charts	Type	Annotation
Revision [1]	Web	2,601	bar, pie, line, scatter plot	bounding box, text
[2]	synthetic, docs	5,125	bar, line, scatter plot	bounding box, text and role labels
FigureQA [4]	synthetic	140,000	bar, line, pie	bounding box, chart data, QA pairs
DVQA [5]	synthetic	3,487,194	bar	bounding box, chart data, QA pairs
Our dataset	Web	187,059	bar	bounding box, visual properties, chart data

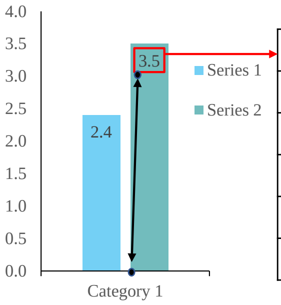
values. For example, DataLabelPosition in Fig. 3 is labeled as "InsideEnd" since the data label is on the top inside the bar.

In preliminary experiments, we observed that the two encoders cannot capture the position information well, and therefore does not perform well on LegendPosition and DataLabelPosition. There are existing works [10]–[12] to incorporate absolute position into the model. However, absolute coordinates cannot explicitly encode the spatial relations, which does not fit our case.

To address this issue, and to more efficiently learn the object spatial relations, we propose a novel method to encode the relative positional features. For each input object k, we introduce a positional indicator $p_k \in \mathcal{R}^5$. The five dimensions indicate the distance to the nearest object respectively in the relation of (1) up; (2) left; (3) down; (4) right; (5) contained. For example, the nearest object to the data label (red box) in Fig. 3 in the **right** direction is legend, and their normalized distance is 0.27. Also, the data label is **contained** in a bar (overlap area rate $>$ 0.7), and their normalized distance is 0.96. The distance function is defined as:

$$d_{norm} = \frac{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}{\sqrt{img_w^2 + img_h^2}} \tag{6}$$

given the bottom-left coordinates of two objects $(x_1, y_1), (x_2, y_2)$. Specifically, img_w, img_h represents the width and height of the input image. d_{norm} is the re-scaled distance function ranged from $[0, 1]$.



Spatial Relation	Object (distance)
Up	None (0)
Left	None (0)
Down	X-axis title (0.87)
Right	Legend (0.27)
Contained	Bar (0.96)

Fig. 3: Spatial relations of a data label object (in red box) with other objects. We calculate the normalized distance between the data label and its nearest objects in the direction of up, left, down, right and contained.

Please note that the choice of distance function depends on different scenarios. In our case, to differentiate the data labels inside the bar (InsideEnd, Center, InsideBase), we choose the distance between two objects' bottom-left points instead of center points[2].

Then the positional indicator is mapped to a vector, and we obtain the positional feature of the kth object as:

$$h_k^p = V p_k \tag{7}$$

where V is the model parameter. Next, the positional feature is concatenated to the region feature v_k as the input.

D. Visual Property Prediction Layer

For each visual property t, the model concatenates the image encoder output x_l and the object encoder output c_t to obtain the final representation o_t. The probability distribution over all possible values for the corresponding visual property is:

$$P_t = \text{softmax}(W_t o_t) \tag{8}$$

where W_t is the model parameter and $o_t = [x_l, c_t]$.

As for BackgroundColor and ForegroundColor, we extract colors directly from the bars detected by Faster-RCNN with a tool[3]. This simple approach for color extraction has achieved an ForegroundColor accuracy of 85.5% on development set.

E. Objective

Given a target ground truth property y_t^* and a model with parameters λ, we minimize the standard entropy loss:

$$\mathcal{L}_{std}(\theta) = -\sum_{\forall t \in \mathrm{T}} \log p(y_t^* | x) \tag{9}$$

where we sum up the losses for T visual property predictions.

Imbalanced Classification. We discover that some visual properties suffer from highly-skewed class imbalance during training. To address the issue, we apply weights to the loss function:

$$\mathcal{L}(\theta) = -\sum_{\forall t \in T} \sum_{\forall i \in N_t} w_{ti} \log p(y_{ti}^* | x) \tag{10}$$

where w_{ti} is calculated according to inverse frequency of class i:

$$w_{ti} = \frac{n_{ti}}{\min_{j \in N_t} n_{tj}} \tag{11}$$

$$\text{s.t.} \qquad n_{tj} > 0 \tag{12}$$

[2] distance between center points cannot differentiate data label of InsideEnd and InsideBase, since they are symmetric and thus have the equal distance to the center point.

[3] https://pypi.org/project/extcolors/

n_{ti} is the number of instances labeled with class i in visual property t, and N_t is the number of classes in property t.

IV. Experiment

A. Settings

Here we describe the baselines and implementation details in the experiments.

1) Baselines: We consider the following baselines:

- **Rule System.** As we have trained the Faster-RCNN with chart object bounding boxes for the Object Encoder input, we implement a rule-based system with its detection output. For example, if Faster-RCNN detects a chart title, then HasChartTitle is set to True. If the Faster-RCNN detects a legend, we use its predicted bounding box to decide the value of LegendPosition with predefined rules. The rule-based system cannot handle certain properties (e.g., HasBarBorder, HasGridlines), since no object outputs can be related to them.
- **Image Encoder-Classification (IEC).** The input image is encoded only by the Image Encoder previously described, and fed into the classification layer for the final prediction. Please note that under this setting, the architecture is the same as DenseNets [6].
- **Object Encoder-Classification (OEC).** The input image is encoded only by the Object Encoder previously described (without positional feature), and fed into the classification layer for the final prediction.

2) Implementation Details: The chart images are resized to short side 800px [13]. The Image Encoder has four dense blocks with numbers of layer (6, 12, 24, 16) respectively. Kernel size of Conv layers is 3×3. The growth rate is 8. For the Object Encoder, we first train a Faster-RCNN[4] to get region features. $K = 100$ for object input. We select regions with confidence > 0.5. The dimension of region features, style embedding, position feature is 2048, 30 and 300. We train for 5 epochs with batch size 64. We use the Adam [14] optimizer with its hyper-parameters set as: learning rate $\alpha = 0.001$, momentum parameters $\beta_1 = 0.9$ and $\beta_1 = 0.999$, and $\epsilon = 10^{-8}$. We apply gradient clipping [15] with range $[-5, 5]$.

B. Results

1) Overall Performance: Table V shows the overall evaluation results. For rule system, we show the detection results of Faster-RCNN in Table VII. From the table, we can see that the plot area and legend classes are easier to be detected. The performance of x-axis title and y-axis title is not promising, which leads to error propagation of the rule system baseline. The rule system performs badly on most properties, and cannot handle certain properties (HasBarBorder and HasGridLines) as no chart objects are related to them. Both IEC and OEC outperform the rule system on most of visual properties, and OEC is better than IEC except on HasGridLines where no chart objects are related to grid lines for the region features. This result indicates that IEC and OEC can capture different level of features from the image. By combining IEC and OEC, our model significantly outperforms the baseline models on all visual properties. In addition, the positional feature is useful, since under the ablation setting (w/o pos.), the performance drops consistently on all visual properties.

2) DataLabelPosition Performance: We report accuracy of different classes in Table VI. Our model achieves the best results on {None, InsideEnd, OutsideEnd}, and obtains the highest macro accuracy. Please note that there are only a few instances (0.79%) labeled as InsideBase in the test, thus small margin of performance gain/loss on this class might not be statistically convincing. It is not surprised that the rule system has poor performances. Prediction of DataLabelPosition requires to first pair up data label object and bar object. When rule system needs complex rules to implement this while being fragile, our model has encoded the object relations and therefore performs significantly better.

3) LegendPosition Performance: We further investigate the accuracy of different classes in LegendPosition. From Table VIII, we can see that our model outperforms the baselines and the ablation setting on most classes, and our model achieves the best macro accuracy. As for the class of {Bottom, Left}, through our error analysis, we figure out that some x-axis title and y-axis title objects are wrongly detected as legend, which might misleads the prediction for these classes, as well as HasXaisTitle and HasYaxisTitle in Table V.

C. Effect of Weighted Loss

Next, we investigate the effect of weighted loss designed for imbalanced classification. Fig. 4 shows the DataLabelPosition accuracy with the standard loss function and weighted loss function respectively. From the figure, we can see that by using the weighted loss function, the model performs significantly better on minority classes while preserving promising accuracy on the dominant class (i.e. None). Especially, the accuracy of both“InsideEnd” and “InsideBase” drop to 2.8% and 0.14% using standard loss function, which implies that standard loss cannot provide sufficient training signals to minority classes.

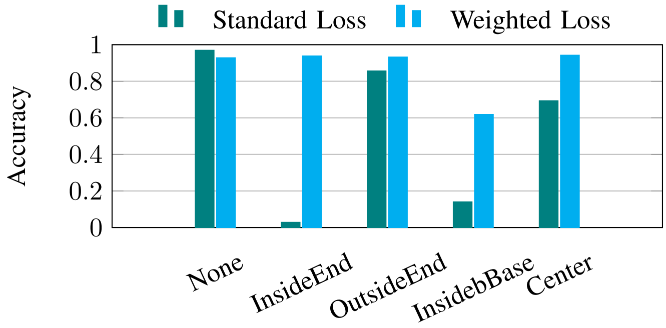


Fig. 4: DataLabelPosition classification accuracy with different loss function strategy.

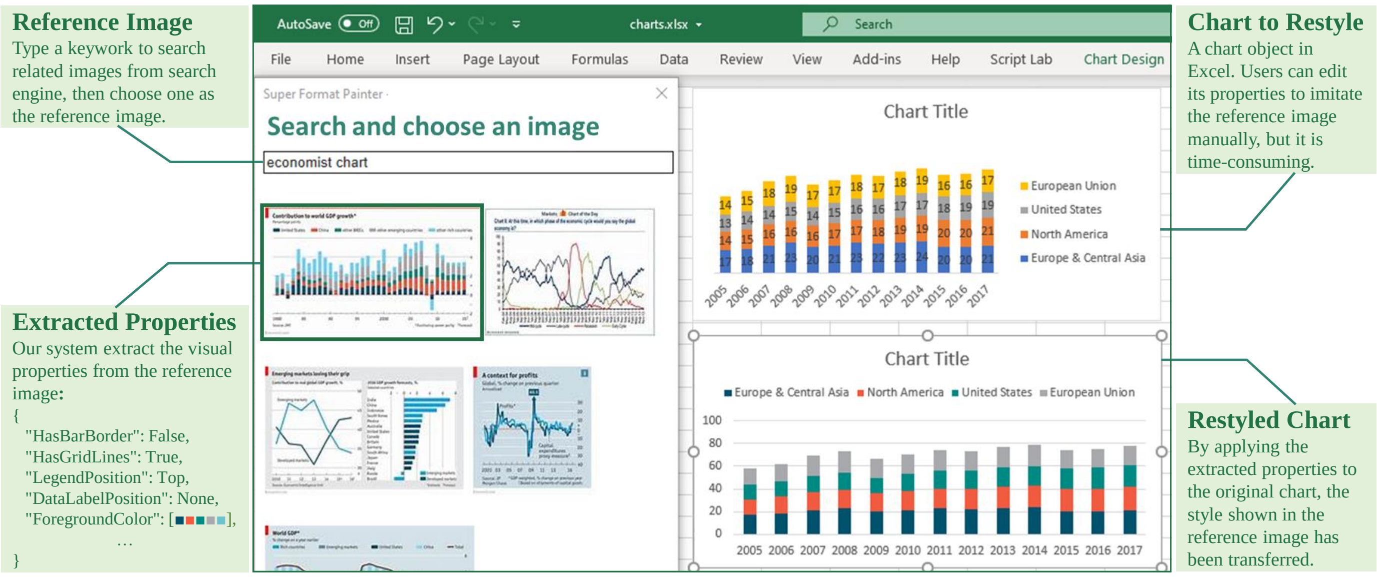
[4] https://github.com/facebookresearch/maskrcnn-benchmark

TABLE V: Accuracy (%) of different visual properties. “w/o pos.” means ablation setting of positional feature. “-” means that the system does not have an output.

Models	HasBarBorder	HasChartTitle	HasXaxisTitle	HasYaxisTitle	HasGridLines	LegendPosition	DataLabelPosition
Rule	-	91.44	16.62	36.66	-	74.39	75.52
IEC	79.75	88.58	89.60	94.80	93.98	90.08	84.07
OEC	80.18	91.69	88.02	95.03	84.26	90.03	91.66
Our model	**88.07**	**92.06**	**93.06**	**95.94**	**94.50**	**94.71**	**92.14**
w/o pos.	80.61	90.16	88.74	93.84	94.37	92.76	91.70

TABLE VI: Accuracy (%) of each class in `DataLabelPosition`. acc_M means macro accuracy.

Models	None	InsideEnd	OutsideEnd	InsideBase	Center	acc_M
Rule	86.66	2.08	81.91	0	4.70	35.06
IEC	92.43	75.69	78.51	14.55	48.90	62.02
OEC	91.61	81.94	91.66	**67.27**	92.24	85.75
Our model	**92.77**	**93.75**	**93.19**	61.82	94.20	**86.95**
w/o pos.	92.72	91.66	88.67	60.09	**94.67**	85.56



Reference Image
Type a keyword to search related images from search engine, then choose one as the reference image.

Extracted Properties
Our system extract the visual properties from the reference image:

```
{
  "HasBarBorder": False,
  "HasGridLines": True,
  "LegendPosition": Top,
  "DataLabelPosition": None,
  "ForegroundColor": [■■■■■],
          …
}
```

Chart to Restyle
A chart object in Excel. Users can edit its properties to imitate the reference image manually, but it is time-consuming.

Restyled Chart
By applying the extracted properties to the original chart, the style shown in the reference image has been transferred.

Fig. 5: The restyling application which is integrated with Microsoft Excel as a plugin. When a user is working on chart styling, he/she can type a keywork to search reference image from the search engine. After that, the system extracted the visual properties from the reference image and then transfer the extracted style to user’s original chart.

TABLE VII: Detection results on our test set using Faster-RCNN.

Object Class	mAP (%)
Plot Area	80.2
Bar	68.1
X-axis Title	43.9
Y-axis Title	51.6
Chart Title	62.1
Legend	75.4
Data Label	51.1
Average	68.1

D. Attention Visualization

In order to demonstrate that our model attends to the correct chart objects given different visual properties, we show the attention visualization in Fig. 6. The lighter the object is, the higher attention score it has obtained. In the figure, we can see that while predicting the property `LegendPosition`, the attention is strongly focused on the legend object (the light region on the right) with the highest attention score of 0.9999.

TABLE VIII: Accuracy (%) of each class in `LegendPosition`. acc_M means macro accuracy.

Models	None	Top	Bottom	Left	Right	acc_M
Rule	95.04	70.16	50.78	**94.05**	76.15	77.23
IEC	91.78	77.54	92.49	82.73	93.30	87.57
OEC	90.54	84.61	94.55	76.19	89.45	87.07
Our model	**95.50**	**87.12**	93.45	93.45	**95.61**	**93.81**
w/o pos.	93.18	85.87	**94.22**	92.86	94.58	92.14

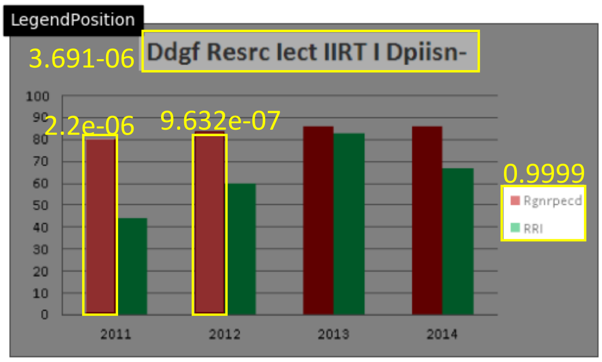


Fig. 6: The visualized attention when predicting `LegendPosition`. We only show 4 objects with highest attention scores. The lighter the object is, the higher attention score it has obtained.

E. Error Analysis

We have observed two main error types in our model predictions:

- **Charts with Complicated Relations.** Figure 7a is a stacked bar chart, where each bar has its corresponding data labels with complicated relative position information. We will improve the relationship modeling for more diverse charts in the future.
- **Object Detection Error Propagation.** Our model depends on object detection of Faster-RCNN. Some objects in chart domain are challenging, such as Figure 7b. The errors from Faster-RCNN would propagate to our model inference.

F. Application: Chart Restyling

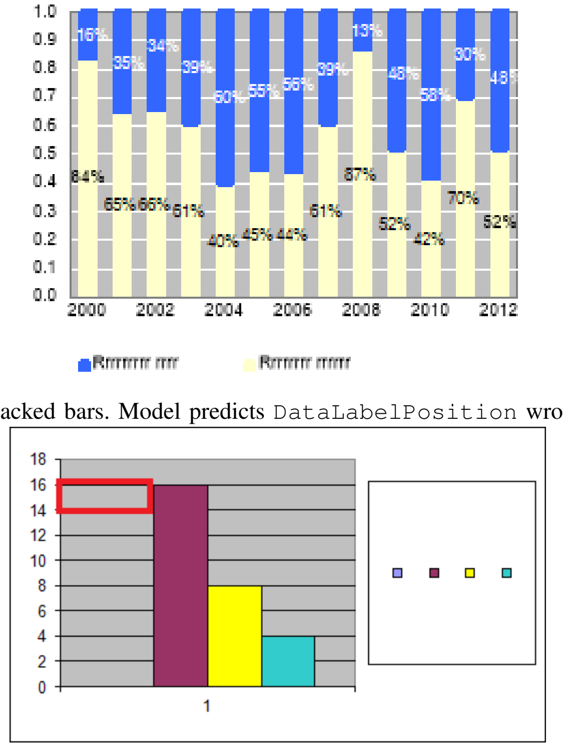
Figure 5 shows the application of chart restyling which is integrated with Microsoft Excel as a plugin. In this application, when a user is working on chart styling, he/she can type a keyword to search related image from the search engine and choose on as the reference image. After that, the system extracts the visual properties from the selected reference image and then applies the extracted style to user's original chart for restyling.

For this application, in addition to the visual properties defined in Table I, we also extract colors of bar objects as another visual property. Specifically, from the bar objects detected by Faster-RCNN, we extract their colors using a third-party tool to get the value of colors. This simple implementation for color extraction is effective, with an accuracy of 85.5% on our development set.

V. Related Work

A. Chart Redesign

Previous works mainly follow the approach of recovering data content in a chart image, then fit to a predefined style template. They are mostly rule-based and make strong simplifying assumptions. [1] locates graphical marks (e.g., bar, pie location) using connected component algorithm [16] and extracts the underlying data with heuristic rules. [2] extracts text elements in the chart with Optical Character Recognition (OCR) and a pipeline of post-processing. [17] extracts data from D3 (a JavaScript library) visualization with the assumption that direct access to the generating code of visualization is available. [18] only extracts color given the legend.



(a) Stacked bars. Model predicts `DataLabelPosition` wrongly.

(b) Faster-RCNN wrongly detects the highlight box as a bar object.

Fig. 7: Error cases.

This paper considers a different scenario to generate style template by extracting visual properties from reference images. Beyond data extraction, it requires visual understanding of chart images (e.g., spatial inference), which is more challenging. Compared with previous work, our task does not take assumption of any ground-truth object inputs (e.g., legend region). Also, this paper proposes a learning-based network on a large-scale real-world dataset, surpassing the limits of synthetic or small-size data.

B. Visual Question Answering

The task of visual question answering (VQA) is also related to our task, which deals with answering questions given visual clues. VQA focuses on open-domain questions for natural images. Many models [19]–[21] have been proposed based on VQA datsets [22]–[25]. Please refer to the more extensive reviews [26], [27].

Recently, several VQA datasets for charts have been introduced. For example, FigureQA [4] is a synthetic chart dataset with QA pairs. The questions in the dataset are generated from 15 templates with binary answer type (i.e. True/False).

DVQA [5], which can be seen as an enriched version of FigureQA in terms of variations of visual style and question types. Among the questions, the type of structure understanding can be seen as most related to visual property prediction (e.g., "How many bars are there?"). Questions under this category are simple and can be mostly solved by object detection and heuristic rules. Being compared, the visual properties defined in this paper require more visual understanding and inference.

VI. Conclusion

In this paper, we explore the task of extracting visual properties from chart images. An end-to-end network has been proposed. Furthermore, we propose a novel mechanism to encode relative positional features. To facilitate research on this domain, we construct a large-scale dataset of chart images from real-world data. Experimental results show that our model achieves the state-of-the art performance. Finally, we present an add-on application of our model for chart restyling. In the future, we plan to consider more chart types and more visual properties.

References

[1] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "Revision: Automated classification, analysis and redesign of chart images," in *ACM User Interface Software & Technology (UIST)*, 2011.

[2] J. Poco and J. Heer, "Reverse-engineering visualizations: Recovering visual encodings from chart images," *Comput. Graph. Forum*, vol. 36, no. 3, pp. 353–363, Jun. 2017.

[3] J. Choi, S. Jung, D. G. Park, J. Choo, and N. Elmqvist, "Visualizing for the non-visual: Enabling the visually impaired to use visualization," in *Computer Graphics Forum*, vol. 38, no. 3. Wiley Online Library, 2019, pp. 249–260.

[4] S. E. Kahou, V. Michalski, A. Atkinson, Á. Kádár, A. Trischler, and Y. Bengio, "Figureqa: An annotated figure dataset for visual reasoning," *ArXiv*, vol. abs/1710.07300, 2017.

[5] K. Kafle, S. Cohen, B. Price, and C. Kanan, "Dvqa: Understanding data visualizations via question answering," in *CVPR*, 2018.

[6] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2018.

[7] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *CVPR*, 2018.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015, pp. 91–99.

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR 2015*, 2014.

[10] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 4055–4064.

[11] Y. Wang, H. Yang, X. Qian, L. Ma, J. Lu, B. Li, and X. Fan, "Position focused attention network for image-text matching," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 2019, pp. 3792–3798.

[12] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," *ArXiv*, vol. abs/1712.06080, 2017.

[13] R. Girshick, "Fast r-cnn," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[14] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of 3rd International Conference for Learning Representations*, San Diego, 2015.

[15] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks." *ICML (3)*, vol. 28, pp. 1310–1318, 2013.

[16] M. B. Dillencourt, H. Samet, and M. Tamminen, "A general approach to connected-component labeling for arbitrary image representations," vol. 39, no. 2, 1992.

[17] J. Harper and M. Agrawala, "Deconstructing and restyling d3 visualizations," in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '14. New York, NY, USA: ACM, 2014, pp. 253–262. [Online]. Available: http://doi.acm.org/10.1145/2642918.2647411

[18] J. Poco, A. Mayhua, and J. Heer, "Extracting and retargeting color mappings from bitmap images of visualizations," *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2018. [Online]. Available: http://idl.cs.washington.edu/papers/extracting-color-mappings

[19] K. Kafle and C. Kanan, "Answer-type prediction for visual question answering," 06 2016, pp. 4976–4984.

[20] Z. Yang, X. He, J. Gao, L. Deng, and A. J. Smola, "Stacked attention networks for image question answering." in *CVPR*. IEEE Computer Society, 2016, pp. 21–29.

[21] M. Acharya, K. Kafle, and C. Kanan, "Tallyqa: Answering complex counting questions," in *AAAI*, 2019.

[22] M. Ren, R. Kiros, and R. S. Zemel, "Exploring models and data for image question answering," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015, pp. 2953–2961.

[23] M. Malinowski and M. Fritz, "A multi-world approach to question answering about real-world scenes based on uncertain input," in *NIPS*, 2014.

[24] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Parikh, and D. Batra, "Vqa: Visual question answering," *Int. J. Comput. Vision*, vol. 123, no. 1, pp. 4–31, 2017.

[25] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[26] Q. Wu, D. Teney, P. Wang, C. Shen, A. Dick, and A. Hengel, "Visual question answering: A survey of methods and datasets," *Computer Vision and Image Understanding*, 2016.

[27] K. Kafle and C. Kanan, "An analysis of visual question answering algorithms," in *ICCV*, 2017.