ChartReader: A Unified Framework for Chart Derendering and Comprehension without Heuristic Rules

Zhi-Qi Cheng[1] Qi Dai[2*] Alexander G. Hauptmann[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Microsoft Research
zhiqic@cs.cmu.edu, qid@microsoft.com, alex@cs.cmu.edu

Abstract

Charts are a powerful tool for visually conveying complex data, but their comprehension poses a challenge due to the diverse chart types and intricate components. Existing chart comprehension methods suffer from either heuristic rules or an over-reliance on OCR systems, resulting in suboptimal performance. To address these issues, we present ChartReader, a unified framework that seamlessly integrates chart derendering and comprehension tasks. Our approach includes a transformer-based chart component detection module and an extended pre-trained vision-language model for chart-to-X tasks. By learning the rules of charts automatically from annotated datasets, our approach eliminates the need for manual rule-making, reducing effort and enhancing accuracy. We also introduce a data variable replacement technique and extend the input and position embeddings of the pre-trained model for cross-task training. We evaluate ChartReader on Chart-to-Table, ChartQA, and Chart-to-Text tasks, demonstrating its superiority over existing methods. Our proposed framework can significantly reduce the manual effort involved in chart analysis, providing a step towards a universal chart understanding model. Moreover, our approach offers opportunities for plug-and-play integration with mainstream LLMs such as T5 and TaPas, extending their capability to chart comprehension tasks.[1]

1. Introduction

The adage, "a picture is worth a thousand words," underscores the immense value of charts found on various websites and articles, which often depict knowledge that cannot be conveyed through words alone. Chart derendering, which refers to the conversion of charts into tables (i.e., Chart-to-Table [11, 41, 42]), is widely viewed as essential in facilitating a range of downstream tasks, such as chart question-answering (ChartQA [27, 45, 66]) and chart summarization (Chart-to-Text [7, 15, 48]). As shown in Figure 1, the Chart-to-Table task aims to recognize the chart as a machine-readable table, while ChartQA and Chart-to-Text tasks involve answering pre-set questions and summarizing the content of the chart, respectively. The interdependence and mutual value of these research tasks have also been emphasized in earlier studies [22, 58, 61], underscoring their critical role in chart comprehension research.

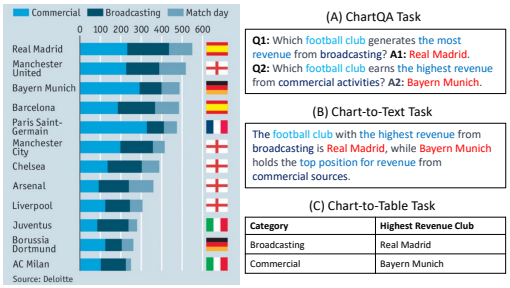


Figure 1. Illustration of chart derendering and comprehension tasks. The Chart-to-Table task aims to transform a chart into a machine-readable table, while ChartQA and Chart-to-Text tasks involve answering questions and summarizing the content of the chart, respectively. [Best viewed in color].

Despite the critical role of chart comprehension, existing research has failed to address the three sub-tasks separately, let alone propose a universal solution. As depicted in Figure 2, charts come in various types, each designed to convey domain-specific knowledge, and can exhibit intricate components, texture variations, and speckled backgrounds. Confronted with such complex charts, existing chart comprehension methods face two main problems.

Firstly, existing chart derendering approaches [4, 11, 18, 41, 42] (i.e., Chart-to-Table) resort to heuristic rules that demand extensive domain knowledge and effort to formulate. For example, ChartOCR [42], a pioneering method, requires chart classification to identify categories first and then detects different components using various pre-defined heuristic rules. To avoid complicated rule-making, some methods even try only one set of limited chart types, such as

*Corresponding author
[1]Code is at https://github.com/zhiqic/ChartReader

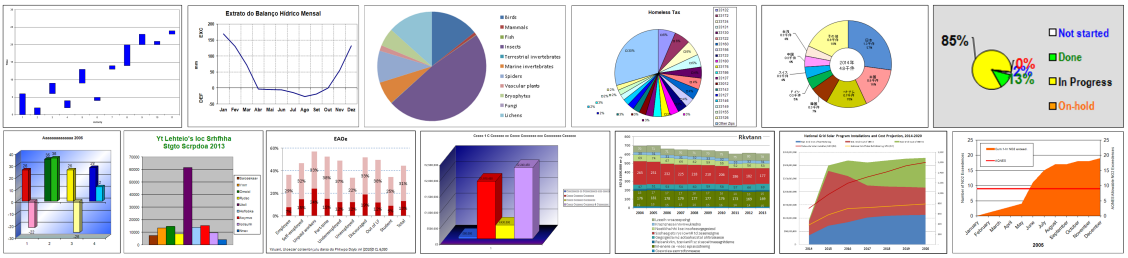


Figure 2. Demonstrates the complexity of charts in EC400K [42], which can vary in type, design, and visual properties. Charts can contain intricate components, texture variations, and speckled backgrounds, posing challenges for chart comprehension. [Best viewed in color].

bar [13, 50] and line [30, 43] charts. These limitations impede the ability to extract data for unknown categories. As a result, many latest methods [40, 45] even use tables directly from ground truth to complete the answers and summary tasks. It is evident that these studies are challenging to automate and struggle to extract data from real-world charts.

On the other hand, current chart comprehension methods, such as Chart-to-Text [7, 15, 48] and ChartQA [29, 36, 44, 45, 54], often heavily rely on off-the-shelf OCR systems or pre-extracted tables from the ground truth. By treating chart derendering as a black box, these methods neglect the visual and structural information of the charts, resulting in the following issues: (1) Chart-to-Text and ChartQA tasks devolve into text-only quizzes [39, 44], as they cannot extract visual semantics from chart derendering. This explains why OCR-based and end-to-end methods, such as LayoutLM [64], PresSTU [31], PaLI [8], CoCa [65], Donut [32], and Dessurt [14], have shown suboptimal results in chart understanding. (2) Chart-to-table tasks do not benefit from chart comprehension tasks. Due to the lack of understanding to the visual semantics in charts, existing systems, such as those using OCR systems [39, 42], struggle to accurately convert charts to tables. Overall, we contend that the problems with chart comprehension arise from an over-reliance on predefined rules and a lack of a universal framework to support multi-tasking.

In light of the previous analysis, it seems that a visual-language model is a promising direction for building a universal framework. However, while Pix2Struct [35], a pre-training strategy for visually-situated language, has shown superior performance over OCR-based models [2, 5, 37, 60], it is not suitable for chart derendering. Moreover, despite recognizing this issue, Metcha [40] had to rely on Pix2Struct as a backbone due to the lack of better visual-language models. Nonetheless, neither Pix2Struct nor Metcha address the two issues identified earlier: 1) excessive reliance on heuristic rules in table derendering, and 2) heavy reliance on existing OCR systems despite attempts to incorporate additional chart comprehension tasks.

To overcome these concerns, we introduce ChartReader, a unified framework that seamlessly integrates chart derendering and comprehension tasks. Our approach comprises a rule-free chart component detection module and an extended pre-trained vision-language model for chart-to-X (text/table/QA) tasks. Unlike heuristic rule-based methods, our approach leverages a transformer-based approach to detect the location and type of chart components, enabling automatic rule learning by processing existing annotated datasets. To enhance cross-task training, we extend the input and position embeddings of the pre-trained model and introduce a data variable replacement technique. Specifically, we standardize chart-to-X (table/text) tasks as question-answering problems, allowing us to solve multiple chart understanding tasks effectively. Additionally, the model generates the data variable instead of the actual value to avoid errors and hallucinations, which improves the consistency in multi-task training. Our approach represents a step towards a unified chart understanding model, as validated through experiments. The proposed framework has the potential to reduce the manual effort involved in chart analysis, paving the way for more efficient and accurate chart comprehension.

To summarize, our contributions are: 1) a unified framework that seamlessly integrates chart derendering and comprehension tasks; 2) a rule-free chart component detection module that leverages a transformer-based approach to automatically learn the rules of charts; 3) extending the input and position embeddings of the pre-trained LLMs and employing a data variable replacement technique to improve cross-task training; 4) validating our approach through experiments, demonstrating significant improvement over existing methods in chart understanding tasks.

2. Related Works

This section reviews related works on chart understanding, including chart derendering, question-answering, and summarization, highlighting the differences from previous work and the impact on visual-language research.

Chart Derendering, also known as Chart-to-Table, involves identifying the constituent components of an image of a chart, such as bars, pies, and legends, to extract the underlying data represented by the chart. Traditional methods [4, 18, 25, 49, 52, 53] relied on hand-designed rules based on edges and colors, which were time-consuming and not easily generalized to new chart types. Deep-learning based approaches [11, 41, 42] utilizing object detection and

text recognition networks have been shown to detect chart components accurately with better generalizability. However, some recent work [39] has attempted to use pre-trained large-scale language models (LLMs) for plot-to-table tasks, without recognizing the structure and components of the chart. Despite the progress made, the flagship methods, ChartOCR [42] and CHARTER [53], still rely heavily on type-specific rules and different networks for each chart type. In contrast, our approach is an end-to-end framework that integrates component detection and utilizes fine-tuned LLMs to eliminate heuristic rules and tackle various chart comprehending tasks.

Chart Question-Answering, or ChartQA, is the task of answering questions related to charts by utilizing both visual and textual information. Early methods [28] relied on relation networks to represent relationships between image objects or between visuals and questions, while dynamic encoding was introduced [26] to handle out-of-vocabulary candidate answers. Later approaches [27, 66] have leveraged multi-modal models that use LSTM [21] and DenseNet [23] to extract visual and text features and fuse embeddings to answer questions. Recent methods, such as OpenCQA [29], CRCT [36], ChartQA [44], PlotQA-M [45], and STL-CQA [54], have incorporated transformers to capture complex visual and text information when answering questions. Furthermore, some methods [40, 44] have attempted to use pre-trained LLMs for chart and language data modeling, but they often disregard chart characteristics and rely solely on ground-truth tables. In contrast, our work unifies chart summarization and derendering tasks into the ChartQA task, using a single sequence-to-sequence framework that elegantly combines chart characteristics and meaningful aspects of different tasks into LLMs.

Chart Summarization, refers to Chart-to-Text, aims to generate natural language summaries from visual charts. Traditional approaches [17, 46, 51] employed templates to provide brief descriptions of the chart's appearance, while others [16, 20] used heuristics or planning-based methods to create multimedia presentations. Recently, Natural Language Generation (NLG) techniques based on heuristic rules have been used, including statistical analysis [12, 55, 62] to infer insights and encoder-decoder architectures [7, 15, 48] to generate template-based captions. However, these methods have a common limitation in that they rely on predefined template-based approaches, which may lack generality and variation in grammatical style and lexical choices. In contrast, our research proposes a universal sequence-to-sequence chart understanding framework that utilizes data-driven models to generate more diverse and informative summaries.

Advancements in Vision-Language Model. Recently, vision-language pretrain model has primarily focused on natural images, relying on visually-grounded reasoning [9, 38, 56, 59, 63] and synthesized datasets [1, 10, 24, 57] for evaluation. However, these works do not capture the complexities of real-world visual language, especially in chart understanding tasks. While OCR-based and end-to-end methods, such as LayoutLM [64], PresSTU [31], PaLI [8], CoCa [65], Donut [32], Dessurt [14], and Pix2Struct [35] have been developed for visually-situated language, they do not specifically address the challenges posed by chart understanding. Despite the widespread use of the encoder-decoder framework, it still requires specific design considerations, such as determining which inputs are valuable and eliminating artificial rules. Without these modules, existing models, such as LaTr [5], GIT [60], DocFormer [2], and SelfDoc [37], cannot be directly applied to chart understanding. Our work achieves impressive results by unifying data extraction and understanding in an encoder-decoder framework, without requiring predefined heuristic rules or limitations.

3. ChartReader Framework

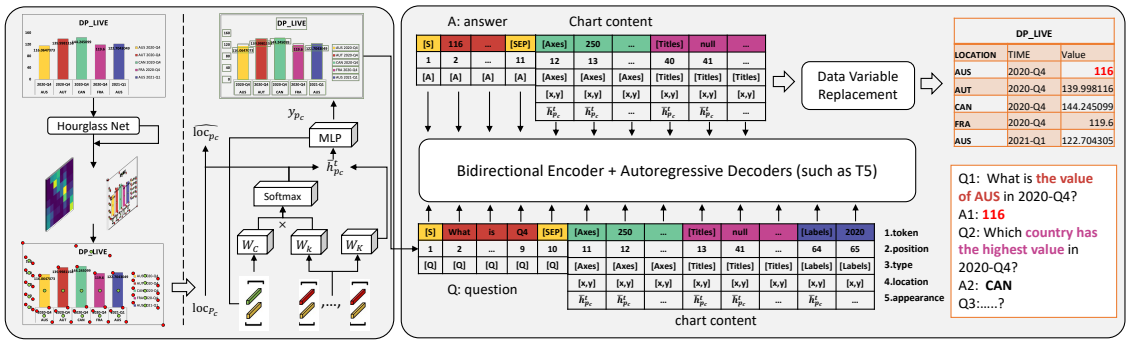
Our unified framework aims to support various chart analysis tasks, including chart-to-table, chart-to-text, and chartQA. As shown in Figure 3, it consists of two main components: (1) a rule-free chart component detection module, and (2) an extended pre-trained vision-language model for chart-to-X (text/table/QA) tasks. We will delve into the functions of each module in detail in the upcoming sections.

3.1. Chart Component Detection

We exploit a transformer-based approach to detect the location and type of chart components without relying on heuristic rules. Our approach consists of three main steps: 1) center/keypoint detection, 2) center/keypoint grouping, and 3) component position/type prediction.

Overcoming Heuristics. The motivation behind this is to overcome the limitations of heuristic rule-based methods in handling various chart styles that heavily rely on domain-specific knowledge and complex rule definitions. By processing annotated existing datasets, our model can automatically learn the rules of charts. Furthermore, the optimized framework can seamlessly be applied to other downstream chart understanding tasks. As illustrated in Figure 4, our updated model relies on center and key points inferred from existing datasets instead of heuristic rules to locate chart components and overcome the effects of style variations. We convert the upper left and lower right corners and position centers of each component into the center and key points, respectively. Although our approach still requires a large amount of annotated data, we do not need to design specific rules for each chart type. Our approach represents a step towards a unified chart understanding model and has been validated through experiments.

Step-1: Center/Keypoint Detection. We detect the centers $p_c \in \mathcal{C}$ and keypoints $p_k \in \mathcal{K}$ of chart components us-



(a) Chart Component Detection (b) Chart Derendering and Comprehension

Figure 3. illustrates the components of our proposed unified framework for various chart analysis tasks, including chart-to-table, chart-to-text, and chartQA. The framework is comprised of two main modules: (1) a transformer-based chart component detection module that eliminates the need for manual rule-making, and (2) an extended pre-trained vision-language model that enables chart-to-X (text/table/QA) tasks. The combination of these two modules enables seamless integration and efficient execution of chart comprehension tasks.

ing the Hourglass network [47] without the corner pooling layer to improve generalization across chart styles. The location $\widehat{\text{loc}}_p \in \mathbb{R}^2$ and component type $\hat{y}_p \in \{1, ..., \mathcal{T}\}$ for all centers and keypoints are predicted using the focal loss L_{focal} [34] and the smooth L_1 loss [19]. We use the central pooling layer for all component types, rather than different pooling strategies for each component type [42], to effectively detect center and keypoint of chart components in various styles.

Step-2: Center/Keypoint Grouping. For each detected center and keypoint, we extract the features of their corresponding positions to obtain initial embeddings. To better determine the chart component, we introduce a type token ϕ_{p_c} and ϕ_{p_k} and use multi-head attention to obtain the weights of the center point p_c and keypoint p_k as,

$$\mathcal{G}(p_c, p_k) = \left(\mathbf{W}_{\mathcal{C}} \left[h_{p_c}, \phi_{p_c}\right]\right)^T \mathbf{W}_{\mathcal{K}} \left[h_{p_k}, \phi_{p_k}\right], \quad (1)$$

where h_{p_c} and h_{p_k} respectively represent the hidden features of the center point and keypoint. The matrices $\mathbf{W}_{\mathcal{C}}$ and $\mathbf{W}_{\mathcal{K}}$ are projection matrices of the hidden features of the center point and the key point. We use fixed sinusoidal features to encode the absolute x- and y-axis positions and add ϕ_{p_c} and ϕ_{p_k} to the embedding before multiplication. After obtaining the weights of $\mathcal{G}$ using Eqn. 1, we compute the final grouping score by normalizing the weights with a softmax function over the entire set of keypoints as,

$$\text{attn}(p_c, p_k) = \frac{\exp\left(\mathcal{G}(p_c, p_k)\right)}{\sum_{\bar{k} \in \mathcal{K}} \exp(\mathcal{G}(p_c, p_{\bar{k}}))}, \quad (2)$$

where the softmax function sorts keypoints $\bar{k} \in \mathcal{K}$ from the same component and filters the most relevant ones (i.e., p_k) for each center point p_c. This approach enables effective grouping of centers and keypoints to their corresponding chart components.

Step-3: Component Position & Type Prediction. We predict the position and type of each chart component by optimizing the center point position using the grouping score and keypoint positions from the Hourglass network. We compute the weighted average of the keypoint positions $\widehat{\text{loc}}_{p_k}$ for each center point p_c using the grouping score as,

$$\widehat{\text{loc}}_{p_c} = \sum_{k \in \mathcal{K}} \text{attn}(p_c, p_k) \widehat{\text{loc}}_{p_k}. \quad (3)$$

To ensure that the predicted positions are close to the ground truth positions loc_{p_c}, we use the location loss L_{loc}, which is defined as,

$$L_{loc} = \sum_{c \in \mathcal{C}} |\widehat{\text{loc}}_{p_c} - \text{loc}_{p_c}|. \quad (4)$$

This supervises the optimization process and helps to accurately predict the position of each chart component.

To predict the type of chart component, we compute a weighted sum of the keypoint embeddings using the grouping score as,

$$\bar{h}_{p_c} = \sum_{k \in \mathcal{K}} \text{attn}(p_c, p_k) \mathbf{W}_{\mathcal{K}} [h_{p_k}, \phi_{p_k}]. \quad (5)$$

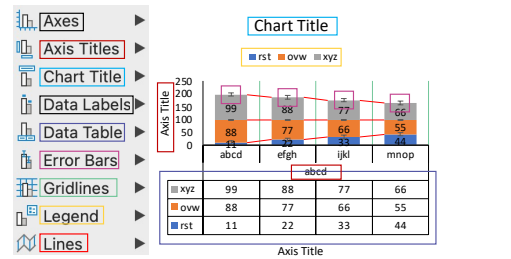
The resulting center point embedding $\bar{h}_{p_c}$ is then passed through an MLP layer to obtain the predicted probability distribution over all component types, which is then compared with the ground truth component type labels y_{p_c} using the cross-entropy loss L_{CLS} as,

$$L_{\text{CLS}} = -\sum_{c \in \mathcal{C}} y_{p_c} \log(\text{softmax}(\bar{h}_{p_c})). \quad (6)$$

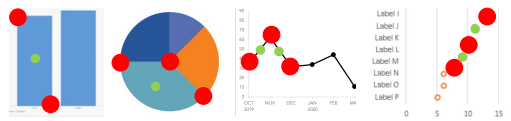
This supervises the optimization process and helps to accurately predict the type of each chart component.

3.2. Chart Derendering and Comprehension

We propose a unified chart understanding framework that handles Chart-to-X (text/table/QA) tasks as chartQA tasks, as illustrated in Figure X. To improve cross-task training, we extend (1) the input and position embeddings of the



(a) Example of char component



(b) Example of center and key point conversion

Figure 4. demonstrates the conversion of chart components into center and key points, which are inferred from existing datasets and used to locate chart components in our updated model. This approach eliminates the need for heuristic rules and allows the model to overcome the effects of style variations. Specifically, we convert the upper left and lower right corners of each component into center and key points, respectively. [Best viewed in color].

pre-trained model and (2) employ a data variable replacement technique.

Motivation & Reasoning. Our approach is motivated by two main reasons: Firstly, we treat both the chart-to-X (table/text) tasks as question-answering problems. Specifically, for chart-to-table, the combination of axis labels and legends forms the question, and the extracted components serve as the answer. Similarly, chart-to-text is treated as a Q&A task to fill in blank templates generated by removing redundant information. By standardizing these previously independent tasks as Q&A tasks, we can effectively train and solve multiple chart understanding tasks. Secondly, recent advancements in these tasks have adopted sequence-to-sequence models. To address all of them in one framework, we extend pre-trained LLMs to diverse chart comprehension tasks. However, integrating these tasks is challenging, and thus, we conduct numerous experiments to determine how to extend input and position encoding. Additionally, we propose a data variable replacement technique to enhance the consistency of multi-task training. Our findings provide possibilities for extending pre-trained LLMs to chart comprehension, and it can be seamlessly extended to mainstream LLMs such as T5 and TaPas.

Positional & Input Embedding. To support tasks such as Chart-to-Table, Chart-to-Text, and ChartQA, we have extended the input and positional embeddings from natural language to chart-style data. The fused embedding z_k at the k-th position of the sequence is defined as,

$$z_k = \mathrm{LN}(z_k^{\mathrm{pos}} + (z_k^{\mathrm{token}} + z_k^{\mathrm{typ}} + z_k^{\mathrm{loc}} + z_k^{\mathrm{app}})), \quad (7)$$

where $\mathrm{LN}()$ denotes layer normalization [3]. Firstly, the positional embedding z_k^{pos} is set to zero, as modern vision-language pre-trained models use relative position embeddings. Secondly, the input embedding includes the type z_k^{typ}, location z_k^{loc}, and appearance z_k^{app} of chart components, as well as the token z_k^{token} obtained from other textual information in the dataset.

Specifically, the z_k^{token} is generated by tokenizing the concatenation of textual words and is denoted as,

$$x^{\mathrm{token}} = \{ \texttt{[S]}, [\mathrm{w}_1], ..., [\mathrm{w}_\mathrm{m}], \texttt{[SEP]}, [\mathrm{y}^1_{\mathrm{pc}}], w_{1,1}, ..., w_{1,m}, [\mathrm{y}^2_{\mathrm{pc}}], \ldots, [\mathrm{y}^{\mathrm{N}}_{\mathrm{pc}}], w_{r_N,1}, ..., w_{r_N,m}, ..., \},$$

where [S] is used to distinguish between questions and answers, and [SEP] indicates the presence of chart context. To incorporate information about chart components, a special token $[\mathrm{y}^{\mathrm{i}}_{\mathrm{pc}}]$ is introduced to represent the chart component type $y^i_{p_c} \in \{1, ..., \mathcal{T}\}$ (such as [Axes]; see Figure 4). Other textual tokens obtained from each chart component are denoted as $\{w_{1,1}, ..., w_{1,m}\}$.

To capture semantic information about the chart, we incorporate a learnable one-hot embedding z_k^{typ} for chart component type. The location of the k-th token within the chart is denoted by x_k^{loc}, which is a 4-dimensional feature based on the relative bounding box coordinates as,

$$x_k^{\mathrm{loc}} = (x_k^{\mathrm{min}}/W_{\mathrm{im}}, y_k^{\mathrm{min}}/H_{\mathrm{im}}, x_k^{\mathrm{max}}/W_{\mathrm{im}}, y_k^{\mathrm{max}}/H_{\mathrm{im}}),$$

where $(x_k^{\mathrm{min}}, y_k^{\mathrm{min}})$ and $(x_k^{\mathrm{max}}, y_k^{\mathrm{max}})$ represent the coordinates of the top-left and bottom-right corners of the bounding box of the k-th token, while W_{im} and H_{im} represent the width and height of the chart, respectively. To take into account the appearance of each chart component, we concatenate the features of the center and corresponding keypoints to obtain the final appearance embedding z_k^{app}.

Data Variable Replacement Technique. We enhance the training process by incorporating the data variable substitution technique. Its pseudo-code is presented in Alg. 1. During training, the model generates the corresponding data variable instead of the actual value, avoiding errors and hallucinations that can result from treating data records as regular tokens. For example, numerical values in the table cells are replaced with data variables such as "product1," "product2," etc. This approach improves the accuracy and factual consistency of generated summaries, tables, and answers, particularly when multiple data records are involved.

To supervise the use of data variables, we introduce a new loss term that penalizes the model for generating tokens that do not match any data variable. The loss term uses a function $D(x)$ that returns the data variable that matches the token x if any and null otherwise. The loss term is defined as follows,

$$L_{\mathrm{var}} = -\frac{1}{T}\sum_{t}^{T}\sum_{i}^{N_t}\sum_{j \in V} I_{D(x_i)=v_j} \log P_{i,j}(t), \quad (8)$$

Algorithm 1 Data Variable Replacement Technique

Input: Data D, Labels Y, Model M,
Output: Trained model M
Initialize model parameters θ
Choose hyperparameter α
for each epoch **do**
  for each x, y in D, Y **do**
    Replace numerical values in x with data variables to create x'
    Get model prediction $M(x')$
    Calculate L_{ans} based on $M(x')$ and y
    Calculate L_{var} using Equation (8)
    Total loss $L = L_{\text{ans}} + \alpha L_{\text{var}}$
    Update θ using gradient descent on L
  end for
end for
procedure INFERENCE(x)
  Replace numerical values in x with data variables to create x'
  Get model prediction $M(x')$
  Replace data variables in $M(x')$ with original values
  return $M(x')$
end procedure

where T is the length of the generated output, N_t is the number of tokens in the t-th output, and $P_{i,j}(t)$ is the probability of generating the i-th token as the j-th data variable at the t-th time step. $I_{D(x_i)=v_j}$ is an indicator function that equals 1 if $D(x_i) = v_j$, and 0 otherwise.

The final optimization is to optimize the sum of the two losses, with α as a hyperparameter that balances the two losses,

$$L = L_{\text{ans}} + \alpha L_{\text{var}}, \tag{9}$$

where α is a hyperparameter that balances the two losses. The loss L_{ans} may be slightly adjusted depending on the specific chart understanding task.[2] In summary, our approach using data variable substitution can simultaneously support chart-to-text, chart-to-table, and chartQA tasks. The introduced loss term also ensures the correct use of data variables during training.

4. Experiments

We conducted experiments on multiple chart understanding tasks, including Chart-to-Table, ChartQA, and Chart-to-Text, as shown in Table 1.

4.1. Evaluation Tasks and Datasets

Chart-to-Table Task. To evaluate the effectiveness of our approach in the Chart-to-Table task, we used the EC400K dataset [42]. This dataset contains 386,966 real-world chart images from public Excel sheets and provides both bounding box locations and numerical readings of the charts. The EC400K dataset offers a wide variety of chart types and styles, surpassing previous datasets used in chart comprehension research. This enables us to validate the performance on diverse and challenging real-world chart data.

ChartQA Task. We evaluated our approach for ChartQA on four datasets: FQA [28], DVQA [26], PlotQA [45], and ChartQA [44]. The FQA dataset includes charts with templates for binary answer questions and has two validation sets and two non-publicly available test sets. DVQA is a synthetic dataset that provides precise location and appearance of visual elements and metadata, including two test tasks: Test-Familiar and Test-Novel. PlotQA is a large and publicly available dataset for chart comprehension tasks, containing charts generated from real-world data. It provides two benchmarks, PlotQA-V1 and PlotQA-V2, with the latter including the former as a subset. ChartQA is a recent open-domain chart Q&A dataset, evaluated on two subsets: augmented and human, where the augmented set is machine-generated and more extractive, while the human set is human-written and requires more complex reasoning.

Chart-to-Text Task. For the Chart-to-Text task, we used the C2T [48] dataset, which contains two subsets: Pew and Statista. The Pew subset consists of chart images from Pew Research Center with automatically extracted summaries, while the Statista subset consists of chart images from Statista with human-written summaries. This dataset provides a diverse set of chart styles and textual summaries, enabling the development of effective chart-to-text models.

Evaluation Metrics. We adopted task-specific metrics in our study. The Chart-to-Table task employed the metrics used in ChartOCR[42] to ensure fairness. For ChartQA, we used standard accuracy with a relaxed correctness criterion [44, 45] that permits a maximum of% numerical error. Additionally, BLEU4 was used to evaluate the Chart-to-Text task.

Table 1. Datasets used for multiple chart understanding tasks, including Chart-to-Table, ChartQA, and Chart-to-Text.

Tasks	Datasets	#Charts	#QAPs
Chart-to-Text	C2T [48]	83K	-
Chart-to-Table	EC400K [42]	387K	-
ChartQA	FQA [28]	180K	2.4M
ChartQA	PlotQA [45]	224K	28M
ChartQA	ChartQA [44]	22K	33K
ChartQA	DVQA [26]	300K	3.5M

4.2. Training Details for Each Task

In this section, we provide an overview of the training details for our method on different tasks. In training, we first independently train the chart component detection mod-

[2]For Chart-to-Text, L_{ans} is the cross-entropy loss between generated and reference summaries. In Chart-to-Table, it's the cross-entropy loss based on chart structure and cell value matching. In ChartQA, it's the cross-entropy loss between predicted and reference answers.

Table 2. Performance comparison of different methods on the ChartQA task. The table shows the results of our proposed models, TaPas (Ours) and T5 (Ours), on four datasets: FQA, DVQA, PlotQA, and ChartQA. Our T5-based model achieved state-of-the-art performance on all datasets, outperforming previous methods by a large margin. The relaxed correctness criterion permits a maximum of 5% numerical error. TF and TN stand for Test-Familiar and Test-Novel, respectively, for the DVQA dataset.

	FQA				DVQA		PQA		ChartQA	
Methods	**Val1**	**val2**	**Test1**	**Test2**	**TF**	**TN**	**Test V1**	**Test V2**	**Val**	**Test**
IMG+QUES [26]	59.41	57.14	-	56.04	21.06	21.00	-	-	-	-
PReFIL [27]	94.84	93.26	**94.88**	93.16	69.66	69.53	57.91	10.37	4.53	4.80
CRCT [36]	94.61	85.04	94.23	84.77	-	-	76.94	34.44	-	-
PlotQA [45]	-	-	-	-	57.99	59.54	53.96	22.52	36.15	38.00
TaPas [44]	90.32	90.43	89.52	89.57	48.82	48.68	15.09	12.90	39.68	41.28
V-TaPas [44]	91.46	91.45	90.68	90.64	94.43	94.54	65.30	42.50	42.60	45.52
T5 [44]	87.97	87.83	87.56	87.57	89.01	89.01	72.62	56.22	40.15	41.04
VL-T5 [44]	88.60	88.49	88.20	88.18	93.75	93.75	75.90	56.02	38.43	41.56
TaPas (Ours)	91.12	91.40	91.15	91.25	92.20	94.30	74.20	56.20	48.30	50.20
T5 (Ours)	**95.50**	**95.80**	94.40	**93.40**	**95.40**	**96.50**	**78.10**	**59.30**	**49.50**	**52.60**

ule, then integrate it with LLMs via extended embeddings, forming an end-to-end training paradigm. The chart component detection model was trained on 8×A6000 GPUs, while all other experiments based on pre-trained LLMs were fine-tuned using 64 GCP-TPUv3. Next we will introduce the training details of each task.

Chart-to-Table Task. We trained the chart component detection module on the EC400K dataset[42], which is currently the largest dataset for Chart-to-Table tasks. Ground truth labels for the position of the center and key points were derived from the bounding boxes labeled in the original dataset. During training, we used the Adam optimizer[33] with a learning rate of 2.5e-4 and reduced the learning rate to 2.5e-5 for the last 5,000 batches, with a batch size of 32. Soft-NMS[6] was used to merge key points from the heatmap. The hyper-parameters were set using a validation set, and early-stopping was used for end-to-end training. Although we did not fine-tune the chart component detection module on other tasks, the final model trained on other chart understanding tasks can still be used for extracting tabular data in Chart-to-Table tasks. The remaining settings were the same as in the previous work, ChartOCR [42].

ChartQA Task. We performed fine-tuning for the ChartQA task on different datasets. For FigureQA, we used binary cross-entropy loss and the Adam optimizer with a base learning rate of 5e-4. The learning rate decayed by a factor of 0.7 from the 15th to 25th epoch. For DVQA, we used multinomial cross-entropy loss and the Adam optimizer with a base learning rate of 7e-4. The learning rate decayed by a factor of 0.6 from the 15th to 25th epoch. For PlotQA, we used binary cross-entropy loss and the Adam optimizer with a base learning rate of 5e-4. Negative examples were generated by randomly assigning wrong answers to questions, and the model was trained for 20 epochs with a linear learning rate scheduler.

Chart-to-Text Task. We fine-tuned the model for 10,000 steps with a learning rate of 2e-5, a batch size of 16, and a maximum sequence length of 512.

Table 3. Comparison of ChartReader with previous state-of-the-art methods on EC400K for Chart-to-Table task. The reported GPU hrs refer to the total amount of GPU processing time used for evaluating on 8 ×A6000 GPUs, which is roughly equivalent to 8 times the running time, including time for IO and data preprocessing. Bold denotes the best performance.

Methods	**Bar**	**Pie**	**Line**	**GPU hrs**
Revision [52]	0.58	0.84	-	-
Faster-RCNN [41, 11]	0.80	-	-	-
Rotation-RNN [41]	-	0.80	-	-
ChartOCR [42]	0.92	0.92	0.96	57h
Ours	**0.95**	**0.95**	**0.97**	22h

4.3. State-of-the-Art Comparisons

Results of Chart-to-Table Task. In Table 3, we report the results of our proposed method on the EC400K dataset [42] for the Chart-to-Table task. Our approach achieved state-of-the-art performance, surpassing previous methods such as RotationRNN and Faster-RCNN, which rely solely on image classification and object detection. The comparison highlights the superiority of our key point detection approach over bounding box detection. Our approach also outperformed earlier attempts such as Revision and ChartOCR, which heavily rely on hand-crafted heuristic rules. Notably, ChartOCR represents the current state-of-the-art in the Chart-to-Table task. Our superior performance can be attributed to two factors. Firstly, we eliminated the need for heuristic rules and learned to recognize chart components by grouping center points and key points. Secondly, the recognized chart components are further utilized in subsequent chart-to-table and chartQA tasks, allowing our model

to better understand the structure and semantic information of charts, leading to improved numerical evaluation performance. Moreover, our method achieved this superior performance with significantly less GPU hours compared to ChartOCR, as shown in Table 3. Specifically, our method only required 22 GPU hours, while ChartOCR used 57 GPU hours. This indicates that our method not only outperforms previous approaches but also does so with more efficient use of computing resources.

Results of ChartQA Task. Table 2 shows the results of our proposed models, TaPas (Ours) and T5 (Ours), on all datasets in the ChartQA task. Our approach utilizes the structural and semantic information of charts to answer questions, making it highly suitable for complex chart understanding tasks such as ChartQA. Specifically, our T5-based model outperformed the previous state-of-the-art method, PReFIL, on the FQA dataset by a small margin, and achieved state-of-the-art performance on the other three datasets. Our model benefits from joint training on chart comprehension and extraction tasks, enabling it to better understand the semantic and structural information of charts. Our T5-based model achieved state-of-the-art performance on the DVQA and PlotQA datasets, outperforming previous methods, including the current state-of-the-art method, V-TaPas, on both test sets. Additionally, our T5-based model achieved state-of-the-art performance on the ChartQA dataset, outperforming previous methods by a large margin, indicating good generalization to unseen data.

Table 4. Results of Chart-to-Text task on the Pew and Statista datasets. Our T5-based model achieved state-of-the-art performance on both datasets, outperforming previous state-of-the-art methods.

Methods	Pew	Statista
T5 [40]	10.5	35.3
PaLI-17B (res. 224) [40]	10.0	40.2
PaLI-17B (res. 588) [40]	11.2	41.4
Pix2Struct [35]	10.3	38.0
MATCHA [40]	12.2	39.4
T5 (Ours)	**14.2**	**44.2**

Results of Chart-to-Text Task. Table 4 shows the results of our T5-based model on the Chart-to-Text task. Our model achieved the best performance on both datasets (Pew and Statista) and outperformed previous state-of-the-art methods PaLI-17B (res. 224) and Pix2Struct. We attribute our superior performance to the ability to generate diverse and informative summaries. Additionally, our T5-based model outperformed PaLI-17B (res. 588) on both datasets, despite having a smaller model size, demonstrating that our approach can achieve good performance even with a smaller model size.

5. Ablation Study

Chart Component Detection. We compare our full model with two variants that lack key point detection or group detection. The full model outperforms the ablated models on all three chart types, and key point detection improves performance on line charts, while the group module is more effective for bar charts. These results highlight the importance of both components for accurate chart component detection.

Table 5. Ablation study on chart component detection, comparing the performance of the full model with two variants that do not have either key point detection or group detection. The full model outperforms the ablated models on all three chart types, highlighting the importance of both components for accurate chart component detection.

Methods	Bar	Pie	Line
w/o Key Point	0.83	0.73	0.63
w/o Group	0.77	0.81	0.52
Ours	0.95	0.95	0.97

Input Encoding. We conducted an ablation study on input encoding methods, comparing full model performance with different ablated versions on PlotQA-V1 and PlotQA-V2 datasets. Table 6 shows that the full model outperforms all ablated versions, with location and appearance embeddings contributing the most to performance. This indicates that spatial and visual information is crucial for chart comprehension tasks.

Table 6. Ablation study on input encoding for the PlotQA-V1 and PlotQA-V2 datasets. The table shows that the full model outperforms all ablated versions, and the location and appearance embeddings contribute the most to the overall performance.

Ablation Token	PlotQA-V1	PlotQA-V2
w/o type	60.20	48.20
w/o location	64.20	51.20
w/o appearance	59.10	45.20
w/o CCD	52.30	40.20
Ours (TaPas)	74.20	56.20

Impact of Hyperparameter α on Data Variable Replacement. We conducted an ablation study to determine the optimal value of α in the Data Variable Replacement Technique. As shown in Table 5, the optimal α varied from 0.2 to 0.7 depending on the dataset complexity. Table 8 compares numerical value accuracy with and without data variable replacement (α set to 0 vs. optimal α). Figure 5 shows the performance variation with α for the T5 model.

Multi-Task Training. We conducted ablation experiments to evaluate the effectiveness of a multi-task training approach on the ChartQA task. Our model, trained on the

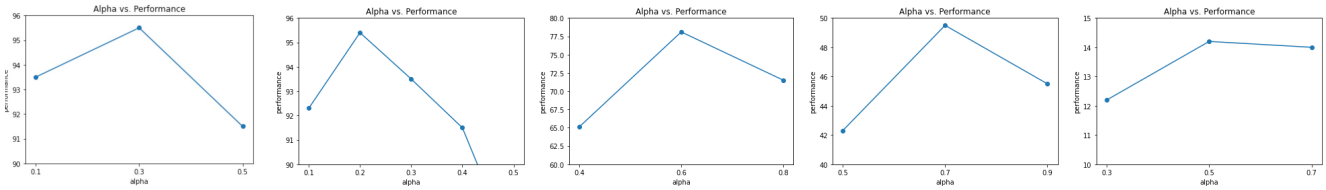


Figure 5. The impact of the hyperparameter α on model performance in chart understanding. The experimental results indicate that the weight of the α parameter increases as the number of open-ended questions grows, as the model needs to use variable replacement techniques more frequently to avoid errors caused by random guessing of unknown markers. Appropriate selection of the α parameter is crucial for achieving high performance and training efficiency of pretrained models.

Table 7. Impact of hyperparameter α in Data Variable Replacement Technique. The QA and Text tasks are divided into sub-datasets of varying difficulty levels, and the "Rate" column shows the fusion weight used for multi-task training.

	FQA	DVQA	PlotQA	CharQA	C2T
α	0.3	0.2	0.6	0.7	0.5
Rate	5%	5%	20%	40%	30%

Table 8. Ablation for Data Variable Replacement on FQA Val1, DVQA TF, PQA TestV1, ChartQA Val for chartQA task, and Pew metric for C2T on Chart-to-Text.

	FQA	DVQA	PQA	ChartQA	C2T
TaPas ($\alpha = 0$)	90.5	86.6	58.6	41.2	10.2
TaPas (optimal α)	91.1	92.2	74.2	48.3	12.8
T5 ($\alpha = 0$)	92.8	91.3	74.5	43.2	11.6
T5 (optimal α)	95.5	95.4	78.1	49.5	14.2

C4+ pre-training dataset with all three chart-related tasks, achieved the highest performance on both validation and test sets, as demonstrated in Table 9. These results highlight the significance of incorporating multi-task training.

Training Task Sequence. We analyzed the optimal sequence for training QA or Text tasks in Table 10. Training QA task last improved performance, especially with QA dataset included. Similarly, training Chart to Text task last resulted in better performance. Multi-task training produced even better performance due to complementarity between QA and Text tasks.

6. Conclusion

In this paper, we propose ChartReader, a framework that integrates chart derendering and comprehension tasks using a transformer-based chart component detection module and a pre-trained vision-language model. Our approach enhances accuracy and eliminates manual rule-making. Through experiments, we outperform existing methods in Chart-to-Table, ChartQA, and Chart-to-Text tasks. Our framework reduces manual effort in chart analysis and facilitates a universal chart comprehension model.

Table 9. Results of ablation study on the effect of pre-training dataset on the ChartQA task.

Model	PT	EF	MT	Val	Test
T5	C4+(PQA)	CQA	✗	38.4	39.2
T5	C4+(PQA)	CQA	✔	42.3	41.5
T5	C4+(PQA+CQA)	CQA	✗	39.2	39.5
T5	C4+(PQA+CQA)	CQA	✔	45.5	43.2
T5	C4+(PQA+CQA+C2T)	CQA	✗	41.2	42.2
T5	C4+(PQA+CQA+C2T)	CQA	✔	49.5	52.6

Table 10. Ablation for Pretraining Task Sequence. This table shows the details of the combined pretraining dataset and the training steps involved in the different tasks, including QA, Text, and Chart to Text tasks. The table also includes the fusion weights used for combining the different datasets within the QA task.

	ChartQA		Chart-to-Text	
Experimental Setting	**Val**	**Test**	**Pew**	**Statista**
QA->Text	46.2	47.2	13.4	43.2
Text->QA	48.2	51.6	12.7	41.2
QA by difficulty	48.3	51.2	-	-
QA with random order	47.5	50.4	-	-
Mixed training	**49.5**	**52.6**	**14.2**	**44.2**

Acknowledgement We extend our sincere gratitude to Siyao Li, Jingdong Sun, and Teruko Mitamura for their invaluable contributions to this paper. This work received support from the Air Force Research Laboratory under agreement number FA8750-19-2-0200, the Defense Advanced Research Projects Agency (DARPA) grants funded through the GAILA program (award HR00111990063), and the AIDA program (FA8750-18-20018).

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

References

[1] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016. 3

[2] Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R Manmatha. Docformer: End-to-end transformer for document understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 993–1003, 2021. 2, 3

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[4] Abhijit Balaji, Thuvaarakkesh Ramanathan, and Venkateshwarlu Sonathi. Chart-text: A fully automated chart image descriptor. *arXiv preprint arXiv:1812.10636*, 2018. 1, 2

[5] Ali Furkan Biten, Ron Litman, Yusheng Xie, Srikar Appalaraju, and R Manmatha. Latr: Layout-aware transformer for scene-text vqa. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16548–16558, 2022. 2, 3

[6] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017. 7

[7] Charles Chen, Ruiyi Zhang, Eunyee Koh, Sungchul Kim, Scott Cohen, Tong Yu, Ryan Rossi, and Razvan Bunescu. Figure captioning with reasoning and sequence-level training. *arXiv preprint arXiv:1906.02850*, 2019. 1, 2, 3

[8] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022. 2, 3

[9] Zhi-Qi Cheng, Qi Dai, Siyao Li, Teruko Mitamura, and Alexander Hauptmann. Gsrformer: Grounded situation recognition transformer with alternate semantic attention refinement. In *Proceedings of the ACM International Conference on Multimedia*, pages 3272–3281, 2022. 3

[10] Zhi-Qi Cheng, Xiao Wu, Siyu Huang, Jun-Xiu Li, Alexander G Hauptmann, and Qiang Peng. Learning to transfer: Generalizable attribute learning with multitask neural model search. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 90–98, 2018. 3

[11] Jinho Choi, Sanghun Jung, Deok Gun Park, Jaegul Choo, and Niklas Elmqvist. Visualizing for the non-visual: Enabling the visually impaired to use visualization. In *Computer Graphics Forum*, volume 38, pages 249–260. Wiley Online Library, 2019. 1, 2, 7

[12] Zhe Cui, Sriram Karthik Badam, M Adil Yalçin, and Niklas Elmqvist. Datasite: Proactive visual data exploration with computation of insight-based recommendations. *Information Visualization*, 18(2):251–267, 2019. 3

[13] Siri Chandana Daggubati, Jaya Sreevalsan-Nair, and Komal Dadhich. Barchartanalyzer: Data extraction and summarization of bar charts from images. *SN Computer Science*, 3(6):1–19, 2022. 2

[14] Brian Davis, Bryan Morse, Brian Price, Chris Tensmeyer, Curtis Wigington, and Vlad Morariu. End-to-end document recognition and understanding with dessurt. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IV*, pages 280–296. Springer, 2023. 2, 3

[15] Seniz Demir, Sandra Carberry, and Kathleen F McCoy. Summarizing information graphics textually. *Computational Linguistics*, 38(3):527–574, 2012. 1, 2, 3

[16] Massimo Fasciano and Guy Lapalme. Intentions in the coordinated generation of graphics and text from tabular data. *Knowledge and Information Systems*, 2(3):310–339, 2000. 3

[17] Leo Ferres, Gitte Lindgaard, Livia Sumegi, and Bruce Tsuji. Evaluating a tool for improving accessibility to charts and graphs. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 20(5):1–32, 2013. 3

[18] Jinglun Gao, Yin Zhou, and Kenneth E Barner. View: Visual information extraction widget for improving chart images accessibility. In *Proceedings of the IEEE International Conference on Image Processing*, pages 2865–2868. IEEE, 2012. 1, 2

[19] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 4

[20] Nancy L Green, Giuseppe Carenini, Stephan Kerpedjiev, Joe Mattis, Johanna D Moore, and Steven F Roth. Autobrief: an experimental system for the automatic generation of briefings in integrated text and information graphics. *International Journal of Human-Computer Studies*, 61(1):32–70, 2004. 3

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 3

[22] Danqing Huang, Jinpeng Wang, Guoxin Wang, and Chin-Yew Lin. Visual style extraction from chart images for chart restyling. In *Proceedings of the International Conference on Pattern Recognition*, pages 7625–7632. IEEE, 2021. 1

[23] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 3

[24] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017. 3

[25] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. Chartsense: Interactive data extraction from chart images. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 6706–6717, 2017. 2

[26] Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5648–5656, 2018. 3, 6, 7

[27] Kushal Kafle, Robik Shrestha, Scott Cohen, Brian Price, and Christopher Kanan. Answering questions about data visualizations using efficient bimodal fusion. In *Proceedings of the IEEE/CVF Winter conference on applications of computer vision*, pages 1498–1507, 2020. 1, 3, 7

[28] Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*, 2017. 3, 6

[29] Shankar Kantharaj, Xuan Long Do, Rixie Tiffany Ko Leong, Jia Qing Tan, Enamul Hoque, and Shafiq Joty. Opencqa: Open-ended question answering with charts. *arXiv preprint arXiv:2210.06628*, 2022. 2, 3

[30] Hajime Kato, Mitsuru Nakazawa, Hsuan-Kung Yang, Mark Chen, and Bj
ørn Stenger. Parsing line chart images using linear programming. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2109–2118, 2022. 2

[31] Jihyung Kil, Soravit Changpinyo, Xi Chen, Hexiang Hu, Sebastian Goodman, Wei-Lun Chao, and Radu Soricut. Prestu: Pre-training for scene-text understanding. *arXiv preprint arXiv:2209.05534*, 2022. 2, 3

[32] Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Ocr-free document understanding transformer. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, pages 498–517. Springer, 2022. 2, 3

[33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[34] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision*, pages 734–750, 2018. 4

[35] Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2struct: Screenshot parsing as pretraining for visual language understanding. *arXiv preprint arXiv:2210.03347*, 2022. 2, 3, 8

[36] Matan Levy, Rami Ben-Ari, and Dani Lischinski. Classification-regression for chart comprehension. In *Proceedings of the European Conference on Computer Vision*, pages 469–484. Springer, 2022. 2, 3, 7

[37] Peizhao Li, Jiuxiang Gu, Jason Kuen, Vlad I Morariu, Handong Zhao, Rajiv Jain, Varun Manjunatha, and Hongfu Liu. Selfdoc: Self-supervised document representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5652–5660, 2021. 2, 3

[38] Fangyu Liu, Emanuele Bugliarello, Edoardo Maria Ponti, Siva Reddy, Nigel Collier, and Desmond Elliott. Visually grounded reasoning across languages and cultures. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10467–10485, 2021. 3

[39] Fangyu Liu, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. Deplot: One-shot visual language reasoning by plot-to-table translation. *arXiv preprint arXiv:2212.10505*, 2022. 2, 3

[40] Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Martin Eisenschlos. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. *arXiv preprint arXiv:2212.09662*, 2022. 2, 3, 8

[41] Xiaoyi Liu, Diego Klabjan, and Patrick NBless. Data extraction from charts via single deep neural network. *arXiv preprint arXiv:1906.11906*, 2019. 1, 2, 7

[42] Junyu Luo, Zekun Li, Jinpeng Wang, and Chin-Yew Lin. Chartocr: data extraction from charts images via a deep hybrid framework. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1917–1925, 2021. 1, 2, 3, 4, 6, 7

[43] Anita Mahinpei, Zona Kostic, and Chris Tanner. Linecap: Line charts for data visualization captioning models. *arXiv preprint arXiv:2207.07243*, 2022. 2

[44] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022. 2, 3, 6, 7

[45] Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. Plotqa: Reasoning over scientific plots. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1527–1536, 2020. 1, 2, 3, 6, 7

[46] Vibhu O Mittal, Johanna D Moore, Giuseppe Carenini, and Steven Roth. Describing complex charts in natural language: A caption generation system. *Computational Linguistics*, 24(3):431–467, 1998. 3

[47] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 483–499. Springer, 2016. 4

[48] Jason Obeid and Enamul Hoque. Chart-to-text: Generating natural language descriptions for charts by adapting the transformer model. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 138–147, 2020. 1, 2, 3, 6

[49] Jorge Poco and Jeffrey Heer. Reverse-engineering visualizations: Recovering visual encodings from chart images. In *Computer Graphics Forum*, volume 36, pages 353–363. Wiley Online Library, 2017. 2

[50] Chinmayee Rane, Seshasayee Mahadevan Subramanya, Devi Sandeep Endluri, Jian Wu, and C Lee Giles. Chartreader: Automatic parsing of bar-plots. In *Proceedings of the International Conference on Information Reuse and Integration for Data Science*, pages 318–325. IEEE, 2021. 2

[51] Ehud Reiter. An architecture for data-to-text systems. In *proceedings of the eleventh European workshop on natural language generation (ENLG 07)*, pages 97–104, 2007. 3

[52] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. Revision: Automated classification, analysis and redesign of chart images. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology*, pages 393–402, 2011. 2, 7

[53] Joseph Shtok, Sivan Harary, Ophir Azulai, Adi Raz Goldfarb, Assaf Arbelle, and Leonid Karlinsky. Charter: heatmap-based multi-type chart data extraction. *arXiv preprint arXiv:2111.14103*, 2021. 2, 3

[54] Hrituraj Singh and Sumit Shekhar. Stl-cqa: Structure-based transformers with localization and encoding for chart question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 3275–3284, 2020. 2, 3

[55] Arjun Srinivasan, Steven M Drucker, Alex Endert, and John Stasko. Augmenting visualizations with interactive data facts to facilitate interpretation and communication. *IEEE transactions on visualization and computer graphics*, 25(1):672–681, 2018. 3

[56] Alane Suhr and Yoav Artzi. Nlvr2 visual bias analysis. *arXiv preprint arXiv:1909.10411*, 2019. 3

[57] Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 217–223, 2017. 3

[58] Chaoli Wang and Jun Han. Dl4scivis: A state-of-the-art survey on deep learning for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2022. 1

[59] Junke Wang, Dongdong Chen, Zuxuan Wu, Chong Luo, Luowei Zhou, Yucheng Zhao, Yujia Xie, Ce Liu, Yu-Gang Jiang, and Lu Yuan. Omnivl: One foundation model for image-language and video-language tasks. *Advances in neural information processing systems*, 35:5696–5710, 2022. 3

[60] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100*, 2022. 2, 3

[61] Qianwen Wang, Zhutian Chen, Yong Wang, and Huamin Qu. A survey on ml4vis: Applying machinelearning advances to data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2021. 1

[62] Yun Wang, Zhida Sun, Haidong Zhang, Weiwei Cui, Ke Xu, Xiaojuan Ma, and Dongmei Zhang. Datashot: Automatic generation of fact sheets from tabular data. *IEEE transactions on visualization and computer graphics*, 26(1):895–905, 2019. 3

[63] Zejia Weng, Xitong Yang, Ang Li, Zuxuan Wu, and Yu-Gang Jiang. Transforming clip to an open-vocabulary video model via interpolated weight optimization. *arXiv preprint arXiv:2302.00624*, 2023. 3

[64] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200, 2020. 2, 3

[65] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022. 2, 3

[66] Jialong Zou, Guoli Wu, Taofeng Xue, and Qingfeng Wu. An affinity-driven relation network for figure question answering. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1–6. IEEE, 2020. 1, 3