CHART-Info 2024: A dataset for Chart Analysis and Recognition

Kenny Davila[1 [0000−0001−6308−7113]], Rupak Lazarus[2 [0009−0008−9482−3230]], Fei Xu[3 [0000−0002−9353−9528]], Nicole Rodríguez Alcántara[4 [0000−0002−4405−7063]], Srirangaraj Setlur[3 [0000−0002−7118−9280]], Venu Govindaraju[3 [0000−0002−5318−7409]], Ajoy Mondal[2 [0000−0002−4808−8860]], and C. V. Jawahar[2 [0000−0001−6767−7057]]

[1] School of Computing, DePaul University, Chicago IL 60604, USA
kdavila@depaul.edu
[2] International Institute of Information Technology, Hyderabad, India
rupak.lazarus@research.iiit.ac.in, {ajoy.mondal,jawahar}@iiit.ac.in
[3] CSE, University at Buffalo, Buffalo NY 14260, USA
{fxu3,setlur,govind}@buffalo.edu
[4] Facultad de Ingenieria, Universidad Tecnológica Centroamericana, Honduras
nicole.rodriguez@unitec.edu

Abstract. Charts are tools for data communication used in a wide range of documents. Recently, the pattern recognition community has shown interest in developing methods for automatically processing charts found in the wild. Following previous efforts on ICPR's CHART-Infographics competitions, here we propose a newer, larger dataset and benchmark for analyzing and recognizing charts. Inspired by the steps required to make sense of a chart image, the benchmark is divided into 7 different tasks: chart image classification, chart text detection and recognition, text role classification, axis analysis, legend analysis, data extraction, and end-to-end data extraction. We also show the performance of different baselines for the first five tasks. We expect that the increased scale of the proposed dataset will enable the development of better chart recognition systems.

Keywords: Charts · Dataset · Graphic Recognition.

1 Introduction

Charts are tools for data communication used in a wide range of documents. The pattern recognition community has displayed a significant interest in methods for analyzing and recognizing charts in the wild [8]. There are rules and regular patterns that define how data can be converted into charts of different types. Yet, current models still struggle with complex graphics, especially when these do not adhere strictly to these rules and conventions. In contrast, humans can still successfully interpret the data encoded in these images.

Recent years have seen the development of very deep networks which can solve a variety of tasks as long as they are trained with enough data. For chart

recognition, most available large-scale datasets are synthetic, often created using an artificial process to generate charts based on data from real sources [10,32]. However, users can employ many tools to create charts with diverse visual styles. Often, they do not just convert data tables arbitrarily into charts, but instead use domain knowledge to choose appropriate chart types and conventions and carefully communicate their message [8]. Therefore, using a single tool to create large synthetic datasets does little to capture the diversity of charts in the wild.

Large-scale datasets are also needed to evaluate chart recognition systems. The CHART-Infographics competitions [10,9,11] were proposed with the goal of becoming the go-to chart recognition benchmark. Earlier editions included synthetic datasets, but systems trained only on synthetic charts performed very poorly on real charts seen in documents in the wild. Therefore, recent editions of CHART-Info have focused on providing large chart datasets based on real charts. This work presents the next iteration of this effort, the *CHART-Info 2024* dataset, which provides a major increase in the amount of training data facilitating training of larger models. At the same time, we provide a brand new test set with non-disjoint splits to evaluate each task using far more images.

CHART-Info defines six functional tasks critical to the chart recognition process: Chart Image Classification (Task 1), Text Detection and Recognition (Task 2), Text Role Classification (Task 3), Axis Analysis (Task 4), Legend Analysis (Task 5), and Data Extraction (Task 6). To facilitate the development of task-specific models, each task receives as input the ideal outputs from some of the previous tasks. An additional task is formulated for end-to-end data extraction (Task 7), where only the chart image is provided. Tasks 6 and 7 must produce an approximation of the data table used to create the chart. In this work, we provide baselines for all tasks except 6 and 7. These baselines are based on open-source models, but multiple domain adaptations and heuristic rules have been used to make them work well on chart-specific tasks. We plan to release all chart images, annotations, evaluation tools and the custom baseline code. To get a sense of the complexity and diversity of charts in the dataset, reviewers can access a preliminary copy of it.[5].

2 Related Works

Multiple datasets have been created for different chart-related tasks. Earlier datasets were created by collecting chart images using web engines and manual filtering [34,5]. In this category we find the Revision dataset [34] (2,500 images of 10 chart types) and the dataset by Chagas et al. [5] (4,837 images of 10 chart types). However, web images are often available under restrictive licenses.

Other works have collected charts directly from data-oriented web sources. ExcelChart400k [30] was created by collecting Excel spreadsheets from the web. The dataset contains a total of 386,966 charts extracted from these spreadsheets

[5] https://www.dropbox.com/scl/fo/lk9csn1z2hnws8f1nbpjs/ANNPdRx5ChvoxITRHMO1b_o?rlkey=hlr7lxqhrrvy1njbnv0hsv4b5&dl=0

along with the tabular data used to create them (no manual annotation required). Nevertheless, all charts are created using a single tool, thus they lack visual diversity. The chart text was also replaced with random characters, affecting the ability to incorporate multi-modal methods that use text. Another example is the ChartQA [31] dataset which has 21,945 charts (mostly in vector format) of 3 types (bar, line and pie), extracted from 4 websites. Annotations for visual question answering (VQA) tasks were collected via crowdsourcing.

Some synthetic chart datasets have been proposed for tasks such as data extraction [3,10] and VQA [22,32]. Some important advantages of generating synthetic data include: better scalability, cleaner and more detailed annotations, and relatively low cost. The AdobeSynth dataset [10] has 202,550 chart images (10 classes) generated with Matplotlib using data from multiple web sources. Bajić and Job [3] used Plotly to create a dataset with over 120K images from 20 chart classes. The DVQA [22] dataset contains 300K images of synthetic bar charts generated with the matplotlib library. The PlotQA [32] dataset has 224,377 images (bar, scatter, and line plots) generated using an unspecified tool. DVQA and PlotQA provide millions of question-answer pairs.

Some recent efforts, including this work, have extracted and manually annotated images from scientific literature to create large-scale datasets. These have the advantage of being more reliably available than random images from the web, and many of them are available under licenses that allow their redistribution and even commercial usage. Two examples are the FigureSeer [36] dataset (60K images from 20K papers) and the DocFigure [20] dataset (33K images), but these are mostly designed for figure type classification (including many charts). The CHART-Infographics competitions have led to the creation of datasets with increasing scales. The dataset presented here is an major extension of our previous dataset from ICPR CHART-Info 2022 [11], which is based on real charts extracted from PubMed Central (PMC).

3 The CHART-Info 2024 Dataset

This work extends previous efforts from the CHART-Infographics competitions [10,9,11]. Every edition has provided a novel test dataset based on real charts. While the first edition provided AdobeSynth for training, the second edition [9] expanded the previous test dataset to create the first training dataset based on real charts. The third [11] merged previous datasets to form the new training dataset. The training dataset of CHART-Info 2024 merges previous [11] training and testing datasets, and provides the largest test dataset so far (See Table 1).

The novel test dataset was created using a similar methodology to the previous edition [11]. We selected papers added to the Open Access Section of the PMC between Dec. 2017 and Oct. 2021. We only considered papers released under *CC BY* or *CC-0* licenses containing the keywords "chart" or "plot" in their main text. Out of 241,396 papers matching these filters, we randomly selected 20K papers that were not already included in earlier versions of our dataset. A binary image classifier was used to identify chart candidates from all the figures

Table 1. Distribution of chart types on the training and testing datasets. Values are compared against earlier versions of the dataset.

	ICPR 2020 [9]		**ICPR 2022 [11]**		**ICPR 2024**	
Chart Type	Train	Test	Train	Test	Train	Test
Area	120	52	172	136	308	229
Line	7,401	3,155	10,556	3,400	13,955	5,142
Manhattan	123	53	176	80	256	68
Scatter	875	475	1,350	1,247	2,597	1,311
Scatter-Line	1,260	558	1,818	1,628	3,446	1,684
Pie	170	72	242	191	433	213
Vertical Box	316	447	763	775	1,538	802
Horizontal Bar	429	358	787	634	1,421	636
Vertical Bar	3,818	1,636	5,454	3,745	9,199	3,692
Horizontal Interval	109	47	156	430	586	326
Vertical Interval	342	147	489	182	671	202
Map	373	160	533	373	906	363
Heatmap	138	59	197	180	377	177
Surface	110	45	155	128	283	127
Venn	52	23	75	131	206	121
Total	15,636	7,287	22,923	13,260	36,182	15,093

Table 2. Statistics (min., median, max.) for different image attributes in our dataset.

	Image Width			Image Height			Image DPI			File Size		
Dataset	Min	Med	Max	Min	Med	Max	Min	Med	Max	Min	Med	Max
Training	76	709	10,800	24	486	6,000	28	600	2,400	2,577	49,999	7,986,057
Testing	118	714	6,299	66	490	7016	72	300	2400	3,036	54,506	2,859,605

in these papers. Then, these candidates were manually classified by chart type. Note that all images in our dataset are in JPEG format, just as provided by the PubMed Central. As a result, our dataset has the limitation of not including any images in vector formats. However, we note that all images in vector format can be easily rasterized to be recognized using models trained with raster images.

The original creators of the images in our dataset used a variety of tools to make them leading to diverse quality and sizes as shown in Table 2. Training set images go from 76×75 to $10,800 \times 6,000$ pixels, while testing set images go from 120×66 to $6,299 \times 6,299$ pixels. This is a challenge for vision models expecting fixed resolutions. For task such as image classification, it might be better to downsize large images, but details that help to differentiate between challenging pairs (e.g. line vs. scatter-line) might be lost when the images are shrunk (see Figure 2.b.). For other task such as text recognition, higher resolution images are better because the text is more readable. However, the variability in the relative scales of text and other chart objects can make detection tasks harder.

Data annotation was a collaborative effort between teams at 3 universities: University at Buffalo (USA), IIIT-Hyderabad (India), and UNITEC (Honduras).

Table 3. Available charts for training and testing per task.

Dataset	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
Training	36,182	8,343	8,343	6,965	7,065	5,427	5,427
Testing	15,093	3,280	3,280	3,128	3,128	2,676	2,676

At each location, the annotators collected the ground truth (GT) in 3 sequential stages: *Image class* annotation, *Text* annotation, and *legend, axes and data* annotation. At every stage, annotators were assigned images in batches, and used our publicly available tools [6] to annotate them. The first two stages were semi-automatic, because recognition systems were used to generate initial labels. Then, the annotators had to verify and correct these labels, effectively cutting down the annotation time in half compared to previous years. For quality control, every annotation at every stage had to be approved by one validator who enforced different rules to achieve consistent annotations.

Table 3 shows the number of charts available for training/testing per task. Because of the competition context, previous CHART-Info testing datasets used 5 disjoint splits to evaluate different tasks [9,11]. However, CHART-Info 2024 is an offline evaluation benchmark, eliminating the need for disjoint splits, providing far more data to evaluate each task. Based on the availability of GT per chart, we propose 4 non-disjoint splits for training and testing specific tasks: *DS1*, *DS2*, *DS3* and *DS4*. Split *DS1* is basically the entire dataset (column *Task 1* in Table 3), which can only be used for Task 1. Split *DS2* represents all charts that have GT for task 2 (column *Task 2* in Table 3), used for task 2 only. Tasks 3, 4 and 5 share the same inputs, and Split *DS3* only considers the charts that have GT for the three tasks: 6,957 for training and 3,128 for testing. Finally, Split *DS4* includes all fully annotated charts that can be used to evaluate Tasks 6 and 7 (column *Task 6* in Table 3). All results presented in Section 4 are based on the following protocol: The training portion of each data split is further divided into 80% for training and 20% for validation. Then, the test portion of the same data split is used for evaluation.

4 Tasks and Baselines

In this section, we will describe the inputs, outputs, evaluation metrics and baselines for each task supported by CHART-Info 2024. Based on existing chart recognition systems [8], these tasks are defined in a sequence: chart image classification (Section 4.1), detection and recognition of text (Section 4.2), text role classification (Section 4.3), axes analysis (Section 4.4), legend analysis (Section 4.5), and chart data extraction (Section 4.6). Alternatively, one can design systems that handle the whole process in a end-to-end manner (Section 4.7).

[6] https://github.com/kdavila/ChartInfo_annotation_tools

4.1 Task 1. Chart Image Classification

Task description. The type of the chart in an image determines the rules used to interpret the data encoded in its graphical elements. Therefore, the first task is chart image classification, where every image in the dataset belongs to one of the classes listed in Table 1. We focus on chart types that are well represented in our data source, and some of these might be known under different names.

Inputs, Outputs and Metrics. The input is an image, and the output is a chart class. Evaluation is based on standard classification metrics such as the per-class recall, precision and F1 score. Table 1 shows that this is an unbalanced dataset, therefore we use the macro-average of the F1 scores for all classes as the final score to emphasizes the importance of correctly identifying all classes.

Baselines. We used image classification methods which have achieved good results on chart images in the past. We obtained the following results: ResNet-18 [16] (91.20%), ResNet-32 [16] (91.17%), ResNet-50 [16] (92.37%), Inception V3 [37] (92.07%), Xception [7] (92.98%), MobileNet-V3-Small [18] (89.30%), MobileNet-V3-Large [18] (91.40%), EfficientNet-B0 [38] (91.59%), EfficientNet-B1 [38] (92.14%), Swin-Tiny [29] (91.02%), and Swin-Base [29] (**93.60**%). The highest score is achieved by the more recent model based on transformers [29]. Nevertheless, earlier models such as ResNet-50 [16] still achieve competitive results, with F1 scores only 1.23% lower than the best model. This task can be hard for many reasons [39], and these numbers show that better models are still needed. Some images are hard to classify even for human annotators, like the scatter-line shown in Figure 2.b which can be easily confused with a line chart.

4.2 Task 2. Chart Text Detection and Recognition

Task description. Text is an important component of the semantics of a chart image. The goal of this task is to detect and recognize individual text blocks in the image. Unlike many scene text detection benchmarks, we are not interested in detecting isolated words, but rather we need to identify text blocks meant to be interpreted as a single unit (e.g. an axis title, a legend entry, etc.).

Inputs, Outputs and Metrics. The inputs are the chart image and its classification. The output is a list of text regions (*detection*), represented by quadrilaterals, and their corresponding transcriptions (*recognition*), represented by strings which might include LaTeX notation to handle special symbols and formulas found in charts. For each image, the *predicted texts* are matched against the *GT texts* if their $IoU \geq 0.5$, but a 1-to-1 matching constraint is enforced. The IoU values of matching texts are summed and the total is divided by $max(|predicted\ texts|, |GT\ texts|)$ to produce the *per-image IoU* score. For every GT text, we compute the normalized character error rate (NCER) between its transcriptions and the text of its corresponding prediction. Unmatched GT texts receive a NCER score of 0. All NCER values are summed and divided by $|GT\ texts|$ to produce the *per-image NCER* score. *Detection* and *Recognition* results are evaluated using the average of the *per-image IoU* and *per-image NCER*

scores, respectively. The final metric for Task 2 is the harmonic mean (f-score) of the *detection* and *recognition* scores.

Baselines for Detection. On average, charts have simpler backgrounds than natural scenes, but text regions can be really small, and they can overlap other graphical elements. One missing text region can have a huge impact in the overall accuracy of the whole chart recognition process. While scene text detectors often target isolated words, text in charts is better analyzed using coherent text regions (e.g. a complete data series name). This is hard considering that text regions in charts can go from one symbol to multiple lines of text.

We first considered two out-of-the-box baselines, Tesseract OCR engine v5.3.1 [1] and PaddleOCR engine v3 (PPv3) [24], which achieved average IoU scores of 0.3035 and 0.6022, respectively. These results show that, out of the box, they do not work well on charts probably because our target is text blocks which is not what these methods produce by default. These results already include some post-processing rules that helped but by only so much.

We then considered multiple baselines retrained with our data (*DS2*). Some of these models are based on the PPv3 framework [24], and these are pretrained on the ICDAR 2015 Scene Text Detection dataset [23]. We also considered models from the MMRotate framework [45], which were designed for rotated object detection in aerial images, and are pretrained on ImageNet-1k. We acknowledge that the accuracy of third-party re-implementations might be slightly different to the original models. Table 4 shows results for text detection baselines retrained on our dataset. In the case of PPv3 [24], configurations using the larger ResNet-50 [16] backbone achieved better results than their counterparts using MobileNet-V3 [18], although by a small margin in many cases. Surprisingly, while DB++[27] has produced better results than EAST [44] on natural scenes, EAST achieved better results here. Nevertheless, the best results were obtained using RoI Transformer [12] with Swin-Tiny [29] backbone, from the MMrotate framework [45]. This model might be the best in handling rotated text.

Baselines for Recognition. To make a fair comparison between text recognition baselines, we apply each recognition algorithm over text detection GT. During training and evaluation, the unicodeit Python library is used to convert the special LaTeX annotations into Unicode symbols. This library cannot handle all special symbols and formulas, but is appropriate for the vast majority of alphanumeric strings. We created a custom dictionary based on the 250 most common Unicode symbols in the training set to retrain some of our baselines.

Text recognition models often expect the inputs to contain a single horizontal line of text with at most so many characters. They do not work well with images of texts that are very long, rotated and/or multi-line. Multi-line text blocks are rare in our dataset, but long texts are very common. We use simple rules to identify long and/or multi-line text candidates, and then apply greedy algorithms to cut the image horizontally and/or vertically as required. The recognizer is used over each partition, and the results are concatenated.

Rotated text is common in charts, specially on *axis titles* and *tick labels*, and can be long and/or multi-line as well. All text regions (any rotation) are always

Table 4. Baselines for Chart Text Detection. Backbones include MobileNetv3 [18] (MN3), ResNet-50 [16] (RN50), ReResNet-50 [15] (RRN50) and Swin Tiny [29] (ST)

Method	IoU
DB [26] - MN3	0.7809
DB [26] - RN50	0.7866
PSENet [40] - MN3	0.8242
PSENet [40] - RN50	0.8263
DB++ [27] - RN50	0.7996
EAST [44] - MN3	0.8036
EAST [44] - RN50	0.8396
ReDet [15] - RRN50	0.8875
RoI Trans. [12] - RN50	0.8828
RoI Trans. [12] - ST	**0.8890**

Table 5. Baselines for Chart Text Recognition. These baselines are based on the PPv3 framework, were pretrained on ICDAR 2015 [23] and retrained on our dataset. Results are provided using *Line Splitting* (w LS) and *Without Line Splitting* (w/o LS).

Method	w/o LS NCER	w LS NCER
SAR [25]	0.9064	0.9202
SRN [42]	0.9098	0.9267
RobustScanner [43]	0.9133	0.9312
VisionLAN [41]	0.9132	0.9316
CRNN [35]	**0.9480**	0.9477
StarNet [28]	0.9293	0.9480
RFL [19]	0.9329	0.9506
ABINet [14]	0.9360	0.9528
SVTR [13]	0.9323	**0.9549**

Table 6. Baselines for Chart Text Detection and Recognition. We consider some Out-of-the-box (OOB) configurations, and the best configurations reported earlier.

Detection Method	Recognition Method	OOB	IoU	NCER	H-Mean
Tesseract [1]	Tesseract [1]	Yes	0.3035	0.3663	0.3320
PPv2 [24] DB [26]	PPv2 [24] CRNN [35]	Yes	0.5688	0.7074	0.6305
PPv3 [24] DB [26]	PPv3 [24] SVTR [13]	Yes	0.6022	0.7866	0.6821
RoI Trans. [12] Swin Tiny [29]	PPv3 SVTR [13]	No	**0.8890**	**0.9264**	**0.9073**

projected into axis-aligned rectangular regions, which can only have 0, +90, +180 or −90 degree rotations. The PPv3 framework includes an angle classifier but it did not perform well on chart text. Because of this, we used simple rules based on recognition confidence scores to simultaneously handle rotated text and long and/or multi-line text. When needed, this method tests multiple combinations of rotations with image splitting, and keeps the most confident transcription.

We considered three out-of-the-box baselines, Tesseract OCR [1], PPv2 [24] with CRNN [35], and PPv3 [24] with SVTR [13], which achieved NCER scores of 0.8483, 0.8226 and 0.8921, respectively. These baselines do not use any of our rules to handle special cases. We then experimented using our dataset to retrain multiple models from the PPv3 framework, and applying our rules to handle multi-line and rotated text. Table 5 shows the results for these models, considering whether the horizontal line splitting algorithm (LS) was used to deal with long text candidates or not. We found CRNN [35] to be the most robust in terms of handling long texts on its own. However, by using our horizontal line splitting algorithm, other methods achieved higher recognition results. The best recognition method was SVTR [13].

Complete Baselines. Table 6 shows the results for end-to-end chart text detection and recognition. Here, recognition results are affected by errors made

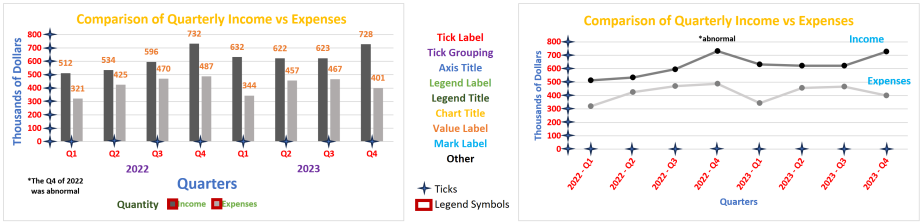


Fig. 1. Targets for multiple tasks. Different text colors are used to illustrate our text roles in two charts (Task 3). We also show the expected ticks using stars (Task 4), and red rectangles define the legend symbols (Task 5). Best seen in digital format.

by the detection model. We only consider out-of-the-box baselines, and the combination of the best models for detection (ROI Trans [12] with Swin-Tiny [29]) and recognition (PPv3 with SVTR [13]). This combined model is also using all of the intermediate rules used for handling rotated, multi-line and/or long texts.

4.3 Task 3. Chart Text Role Classification

Task description. This task aims to determine the role or function that each text region has on the chart. Our dataset considers 9 roles: *chart title*, *axis title*, *tick label*, *tick grouping*, *legend title*, *legend label*, *value label*, *marker label*, and *other*. This covers the categories needed to make sense of a chart image. The *other* category is used to group additional less common roles. These roles are illustrated in Figure 1.

Inputs, Outputs and Metrics. The inputs include the chart image and the GT outputs for Tasks 1 and 2. The expected output is a list of the roles for each GT text region. Like Task 1, evaluation is based on classification metrics.

Baselines. This task is similar to general object classification on images. However, two identical objects (text regions) can have different classes (roles) based on their position within the chart layout. Class imbalance also makes this task challenging, where *tick labels* make about 70.11% of all text regions in the training dataset, while *legend title*, *chart title*, and *tick grouping* are so rare that even combined represent just 1.15% of all text regions. However, all classes have the same impact on the final metrics.

We created our baselines for task 3 by training different object detector models using the role of each text region as their target class. We create variations of these models to simultaneously deal with multiple tasks. The first, *V35*, deals with tasks 3 and 5, and uses the original 9 text roles. The second, *V345*, deals with tasks 3, 4 and 5, and needs 12 roles for text. This is because it replaces the *tick label* class with 4 per-axis classes (more details in Section 4.4).

Task 3 requires assigning classes to text regions in the GT, but the predictions made by the object detectors might not align with the GT. Overlapping pairs of GT text regions and predictions are scored using the harmonic mean of their IoU and prediction confidence. We then greedily pick the highest scoring matches

Table 7. Baselines for Text Role Classification. Columns are F1 scores (%) for *tick label* (TL), *axis title* (AT), *legend label* (LL), *value label* (VL), *legend title* (LT), *mark label* (ML), *tick grouping* (TG), *other* (O), *Chart Title* (CT) and their macro average. For each baseline, we consider the V35 (†) and V345 (‡) variations.

Method	TL	AT	LL	VL	LT	ML	TG	O	CT	AVG
†ReDet [15] (R50 [16])	98.8	97.1	97.9	81.2	79.1	54.4	55.7	71.1	83.2	79.85
†RoI Trans [12] (R50 [16])	98.6	97.3	97.8	81.3	78.0	60.5	54.4	71.0	82.6	80.18
†RoI Trans [12] (Swin Tiny [29])	98.9	98.0	98.1	84.2	79.6	62.1	60.7	74.9	87.9	82.72
†YOLO-V8x [21]	98.8	98.1	97.3	84.7	85.3	63.0	70.7	74.5	89.9	84.71
†Deformable DETR [46]	99.4	98.6	98.6	83.8	87.1	63.8	70.6	75.9	91.3	85.45
†C. Mask R-CNN (Swin Base) [29]	99.2	98.7	98.6	85.4	89.6	68.7	70.9	77.3	92.6	**86.78**
‡ReDet [15] (R50 [16])	98.9	96.7	98.0	81.3	79.6	56.2	57.5	72.2	83.1	80.40
‡RoI Trans [12] (R50 [16])	98.7	97.4	98.1	81.3	75.7	60.2	56.3	71.8	82.6	80.22
‡RoI Trans [12] (Swin Tiny [29])	99.0	98.0	98.1	83.7	81.5	63.4	61.2	74.5	88.5	83.10
‡YOLO-V8x [21]	98.5	96.8	96.8	79.3	76.0	51.9	59.9	67.7	72.7	77.76
‡Deformable DETR [46]	99.4	98.6	98.8	84.2	87.4	66.3	74.9	75.9	92.1	86.40
‡C. Mask R-CNN (Swin Base) [29]	99.2	98.7	99.0	85.2	89.6	69.4	69.8	77.5	90.8	86.57

while enforcing a 1-to-1 matching constraint. The class of the prediction is finally assigned to each of the matched GT text regions. Unmatched predictions are simply ignored, and unmatched GT text regions are omitted.

Table 7 shows the results for Task 3. The *tick label*, *legend label* and *axis title* classes, which represent 87.16% of the training dataset, have very high F1 scores. Meanwhile, the *mark label* and *tick grouping* classes, which represent only 1.98% of the training dataset, have the lowest F1 scores. Except for YOLO-V8 [21], most models have very similar results for both variations. The Cascade Mask R-CNN model with Swin-Base transformer [29] and Deformable DETR [46] are consistently the strongest model from this set.

4.4 Task 4. Chart Axis Analysis

Task description. Axes in charts define the space of the chart data. The goal of this task is to locate the main chart axes (horizontal and vertical), and then link specific points in the axes (ticks) with text (tick labels). This location should be independent of the existence of visual tick marks.

Inputs, Outputs and Metrics. Inputs are the same as Task 3. The output is a dictionary organizing the tick positions by axis. Then, per axis, a set of pairs (text id, point) is expected. Each pair represents a *tick label* by their unique id in the GT, and the point represents the tick position.

Evaluation considers precision and recall of predicted ticks on the main axes (x-axis at the bottom and y-axis at the left). Secondary axes (top or right) are ignored. Predicted and GT ticks are matched by text id, and each match is weighted based on the distance between the GT location and the predicted location. First, the distances are normalized by the length of the image diagonal, and matches with distance ≥ 0.02 receive a weight of 0, and distance ≤ 0.01

Table 8. Baselines for Axes Analysis. All of them are based on Variation 345

Method	Rec. (%)	Prec. (%)	F1 (%)
ReDet [15] (R50 [16])	83.89	85.72	84.79
RoI Trans [12] (R50 [16])	84.09	85.27	84.67
RoI Trans [12] (Swin Tiny [29])	84.56	86.08	**85.31**
YOLO-V8x [21]	54.35	56.64	55.47
Deformable DETR [46]	**85.38**	85.18	85.28
Cascade Mask R-CNN (Swin Base) [29]	77.32	**86.46**	81.63

receive a weight of 1. For $0.01 < \text{distance} < 0.02$, an interpolated weight between 1 and 0 is used. Missing ticks and ticks associated with the wrong axis have weights of 0. The total weight of all matches is divided by the number of GT ticks to compute recall and by the number of predicted ticks to compute precision. Then, the overall recall and precision metrics are the macro averages of the per-axis values. Finally, we compute F1 score as the final per-chart score, and the average of the per-chart scores are computed for the entire evaluation set.

Baselines. While the GT text regions are known for this task, their corresponding roles are unknown. Because of this, our baselines work in combination with role predictions from Task 3, to identify all *tick labels*. The next challenge is to associate these to their corresponding axes. We use Variation V345 (see Task 4.3) which refines the *tick label* class by directly predicting if the region is a *tick label* of: *x-axis* (bottom), *y-axis* (left), *x2-axis* (top) or *y2-axis* (right).

The next step is to associate the *tick labels* to specific image locations. A common idea is to detect *tick marks* and use rules to match these to *tick labels*, but in many cases the *tick marks* are not visible or do not correspond to positions that should be associated with *tick labels*. To solve this problem, we add an *axes corner* object, which is a box of 10-by-10 pixels, centered at the origin of both x and y axes (bottom-left corner). The center of this box, (c_x, c_y), provides the coordinates shared by all ticks in a given axis (c_x for y axis, c_y for x axis). We use rules to determine the other coordinate using the rotated bounding box of the corresponding *tick label*. In most cases we simply use the center of the *tick label* bounding box: vertical center for y axis, horizontal center for x axis. Rotated *tick labels*, commonly found in the x axis, are the exception to this, and we use the x coordinate of the top-most point in their rotated bounding box.

Table 8 shows the results for this task. The performance for most object detectors is reasonably good considering that they do not detect visual tick marks. The baselines will fail when the *axes corner* box is incorrectly detected, or not detected at all. If multiple boxes are detected, the most confident is picked, but that can also produce incorrect outputs. Errors made in Task 3 (e.g., false positives/negatives for *tick labels*) are also propagated here. Also, *tick labels* associated with the wrong axis reduce precision for one axis, and recall for the other. Here, the ROI Trans model [12] achieved the highest score, with more consistent recall and precision levels than other models. The second best is Deformable DETR [46].

4.5 Task 5. Chart Legend Analysis

Task description. A legend is made by a set of *legend entries*, which are (*legend label*, *legend symbol*) pairs. Each *legend label* usually corresponds to one specific data series in the chart. The *legend symbol* exemplifies the appearance of the corresponding *data marks*. An example is shown in the left side of Figure 1. The goal of this task is to identify the *legend entry* pairs in the image.

Inputs, Outputs and Metrics. Inputs are the same as task 3. The output is a list of *legend entry* pairs (text id, bounding box), where the text id represents a *legend label*, and bounding box represents the associated symbol.

The evaluation of this tasks requires correct pairings between *legend labels* and *legend symbols*. For a given chart, predicted *legend entries* are initially matched to GT by the id of the *legend labels*. For each matching pair, the area of intersection between the GT *legend symbol* and the predicted one is computed and used to get two metrics: an IoU-based score (divide by the area of the union) and a recall-based score (divide by the area of the GT bounding box). The sum over all *legend entries* is computed for both scores, and then they are divided by the maximum between the number of GT *legend entries* and predicted *legend entries*. Finally, the average over the evaluation set is computed for both metrics.

Many charts have rather small and thin *legend symbols* (e.g, height of 2 pixels). IoU-score can be over-punishing on bounding boxes that correctly capture the *legend symbol*, but are slightly thicker. The recall-based score is also considered here because of this.

Baselines. Similar to Task 4, this task has access to the GT text regions, but the roles are unknown. All *legend labels* and *legend symbols* candidates need to be identified, and these need to be combined into *legend entry* pairs. As described before, we consider object detection baselines that combine multiple tasks on the same network. For task 5, we simply add the *legend symbol* objects. The same network will produce all *legend label* and *legend symbol* candidates.

The next challenge is to pair the candidates while considering false positives and negatives for both classes. Algorithms typically used for 1-to-1 matching in bipartite graphs will fail due to the noisy predictions. A simple approach is to pair each *legend symbol* with its closest *legend label*, but the way in which the distances are measured determines the quality of results. Based on the observation that for most charts, all *legend entries* have their symbols on the same side, we first estimate if all legend symbols in the image are left, right, above or below their labels. This direction is used to pick the corresponding edges of the bounding boxes of the legend labels, and their middle points are used as reference points for the labels. We then measure the distances between the reference points and the centers of the bounding boxes of the symbols. Matches are then sorted by increasing distance, and they are greedily picked in that order. Only matches between previously unmatched elements are accepted, and the process stops as soon as the first match involving a symbol or label previously matched appears. We do this to prevent spurious matches involving false positives/negatives based on the observation that valid legend entries in the same chart usually have similar edge distances between their symbols and labels.

Table 9. Baselines for Legend Analysis. We consider variations V35 (†) and V345 (‡).

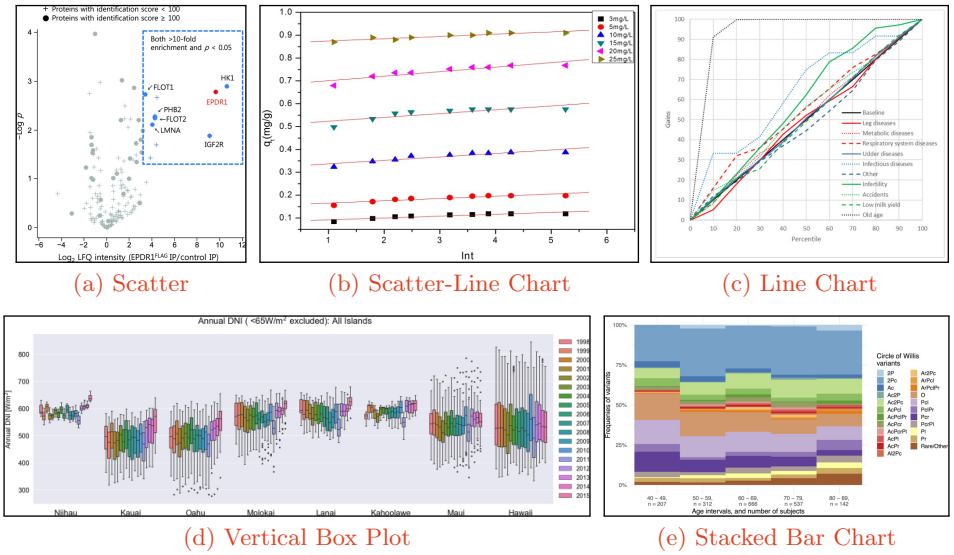
	Average BBox	
Method	IoU (%)	Recall (%)
†ReDet [15] (R50 [16])	83.09	95.33
†RoI Trans [12] (R50 [16])	83.41	95.62
†RoI Trans [12] (Swin Tiny [29])	**84.31**	95.56
†YOLO-V8x [21]	43.86	49.53
†Deformable DETR [46]	80.52	88.53
†Cascade Mask R-CNN (Swin Base) [29]	84.23	93.56
‡ReDet [15] (R50 [16])	83.66	95.57
‡RoI Trans [12] (R50 [16])	83.26	**95.69**
‡RoI Trans [12] (Swin Tiny [29])	83.84	95.42
‡YOLO-V8x [21]	43.42	49.31
‡Deformable DETR [46]	82.86	91.81
‡Cascade Mask R-CNN (Swin Base) [29]	84.12	93.62

Table 9 shows the results for this task. The recall-based scores show that most legend symbols are being detected and matched correctly, and that predicted boxes greatly overlap the symbol regions. It is possible that many of these predictions have the wrong thickness, leading to a much lower IoU-based scores in comparison. Errors can come from false positives/negatives of *legend symbols* and *legend label*. The method with the highest average F1 score for role classification achieves 98.6 F1 score for the *legend label* class (see Table 7). Therefore, it is likely that most errors come from failures to correctly detect the legend symbols. The baselines based on RoI Trans [12] consistently achieve some of the best scores for this task.

4.6 Task 6. Chart Data Extraction

Task description. This task approximates the data table used to create a chart image. It is divided into two sub-tasks: Plot element detection and extraction (Task 6.a), and raw data extraction (Task 6.b). The first sub-task aims at correctly locating the data marks in the chart (e.g., lines in line charts, bars in bar charts, etc.). The second sub-task puts everything together (text, axes, legends, and data marks) to reconstruct the data encoded in the chart image. The chart type might be used to determine the right approach for this task.

Inputs, Outputs and Metrics. The inputs for this task are the outputs from all previous tasks (1-5). The outputs for Task 6.a depend on the chart type. Bar charts require a list of bounding boxes per bar. Line and Scatter plots require a list of points for each data series. Box plots require a tuple with the position of the components of each box (box top, box bottom, box median, top whisker, bottom whisker). For task 6.b, the output is a set of data series with name, and the list of data points (x, y) that make that data series, where x is the independent variable, and y is the dependent variable. Multiple metrics are considered depending on the type of chart (See [9,11] for details).



(a) Scatter (b) Scatter-Line Chart (c) Line Chart

(d) Vertical Box Plot (e) Stacked Bar Chart

Fig. 2. Examples of challenging charts in our dataset. (a) Scatter chart, extracted from [33]. (b) Scatter-line, extracted from [6]. (c) Line chart, extracted from [2]. (d) Vertical Box plot, extracted from [4]. (e) Stacked bar chart, extracted from [17].

Baselines. The baselines for this task are complex, requiring combination of results from previous tasks. Existing works on chart recognition typically focus on a single chart type [8]. Meanwhile, our dataset provides data annotations for: horizontal/vertical bar charts, vertical box plots, scatter plots and line charts. Providing baselines for each of these chart types is out of the scope of this work. However, we briefly discuss the implications and complexities of creating data extraction models for these chart types.

Horizontal/Vertical Bar Charts. This is arguably the simplest chart type for data extraction. Standard object detectors often work with anchors of predefined aspect ratios, and bars in some charts can be extremely narrow or long. Most charts use bars of solid colors, but some charts use complex texture patterns instead. After detecting the bars, the next challenge is to infer the values represented by them. Every bar must be correctly map to a category (x value), a data series (e.g., "legend entry"), and an absolute value (y value). The orientation of the chart defines which axis is the x value (independent variable) and which one is the y value (dependent variable). Many charts use stacked and/or grouped bars, which require associating multiple bars to a single category (e.g., by proximity). Bars are usually associated to particular data series using legend analysis and their appearance. Finally, the y values of the bars are inferred through axes analysis and their spatial location. This process gets slightly more complicated for stacked bars where two extreme points are required to infer the value of each bar. Also, data points with y=0 often lead to *invisible bars*, but

their existence might be inferred by analyzing the chart layout. Figure 2.e shows an example of a complex stacked vertical bar chart included in our dataset.

Vertical Box Plots. The first step is to detect the boxes and their corresponding whiskers and median lines. Object detectors can be used to locate the boxes, but extreme aspect ratios can be a challenge. Whiskers and median lines might not be visible when their values are too close to the values represented by the top and/or bottom of the boxes. Similar to bar charts, we can find grouped box plots, where multiple boxes share a single category and each box needs to be linked to a particular legend entry. There are five points of interest in each box that need to be correctly mapped to their corresponding values in the y-axis. Figure 2.d shows a complex grouped vertical box plot included in our dataset. Note that horizontal box plots can be processed in a similar way, but we excluded them from our dataset because they are quite rare in practice. Also, many box plots include outliers represented by scatter marks, but we decided to currently exclude these from our benchmark because even human annotators often struggle to distinguish and recognize all individual points.

Scatter Plots. First, each data mark representing a data point must be detected. This might be easy when there are only a few large data marks, and extremely difficult when the plot region is cluttered with many small data marks. In the worst case scenario, we might only approximate the distribution of the original data used to create the plot. While our dataset includes all kinds of scatter plots, we only annotated data in cases where human annotators could accurately identify all individual data points. Data marks can be generated using all sorts of shapes, colors and sizes, making their automatic detection very challenging. After detection, each data mark needs to be mapped to a 2D point using their location and axis analysis. Legend analysis and the appearance of the data marks must be used to correctly map them to their corresponding data series. If no legend is present, then the appearance alone must be used to infer the existence of multiple data series. Figure 2.a shows an example of a complex scatter in our dataset where the colors on the legend do not match the colors in the plot region, and matching needs to be done using shapes, but this is also hard due to overlaps between data marks.

Line Plots. This is arguably the most challenging type here. Identifying the pixels of a given line is a segmentation problem which cannot be approached with basic object detectors. Some charts have solid colored lines which can be easily traced by basic segmentation algorithms. However, we find many charts with dashed lines and repeated line colors, where the thickness and/or dash patterns must be considered to differentiate them. Lines can also intersect and significantly occlude each other. Same as other chart types, lines must be associated with data series using legend analysis, but this can be difficult when the associated legend entries are very thin. Figure 2.c shows a line chart in our dataset which displays many of these issues. Some charts do not use legends, and instead directly provide names for each line using colored text within the plot region (e.g. *data mark label*) as illustrated on the right side of Figure 1.

4.7 Task 7. End-to-End Data Extraction

The end-to-end data extraction task has the same goals, outputs and metrics as sub-task 6.b (Section 4.6). However, the only input is the image of a chart. This task was designed for systems that can handle the chart recognition process as a whole, removing the need to produce and evaluate intermediate outputs. It is also possible to create a modular system which handles the recognition process using the individual tasks suggested in this benchmark. Not having any ground truth means that the real outputs from each module in the pipeline must be used. Any errors made in the earlier tasks will propagate to later tasks, affecting the overall recognition results. Similar to Task 6, baselines for this complex task are out of the scope of this work.

5 Conclusion

In this paper, we have introduced the CHART-Info 2024 dataset, a natural extension to the existing ICPR 2022 CHART-Info datasets [11]. Apart from a brand new test dataset, we have also provided a considerable number of baselines for the first 5 tasks defined in our dataset. These baselines provide a starting point for researchers interested in chart recognition. We believe that our dataset will enable the development of better chart recognition systems which will be capable of dealing with complexities found on charts in the wild.

References

1. Tesseract OCR – opensource.google.com, https://opensource.google.com/projects/tesseract
2. Adamczyk, K., Grzesiak, W., Zaborski, D.: The use of artificial neural networks and a general discriminant analysis for predicting culling reasons in holstein-friesian cows based on first-lactation performance records. Animals **11**(3), 721 (2021)
3. Bajić, F., Job, J.: Data extraction of circular-shaped and grid-like chart images. Journal of Imaging **8**(5), 136 (2022)
4. Bryce, R., Carreño, I.L., Kumler, A., Hodge, B.M., Roberts, B., Martinez-Anido, C.B.: Annually and monthly resolved solar irradiance and atmospheric temperature data across the hawaiian archipelago from 1998 to 2015 with interannual summary statistics. Data in Brief **19**, 896–920 (2018)
5. Chagas, P., Freitas, A., Daisuke, R., Miranda, B., De Araújo, T.D.O., Santos, C., Meiguins, B., De Morais, J.M.: Architecture proposal for data extraction of chart images using convolutional neural network. In: 21st IV. pp. 318–323. IEEE (2017)
6. Chen, J., Cai, Y., Clark, M., Yu, Y.: Equilibrium and kinetic studies of phosphate removal from solution onto a hydrothermally modified oyster shell material. PLoS One **8**(4), e60243 (2013)
7. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of CVPR. pp. 1251–1258 (2017)
8. Davila, K., Setlur, S., Doermann, D., Bhargava, U.K., Govindaraju, V.: Chart mining: a survey of methods for automated chart analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(11), 3799–3819 (2020)

9. Davila, K., Tensmeyer, C., Shekhar, S., Singh, H., Setlur, S., Govindaraju, V.: ICPR 2020-Competition on HArvesting Raw Tables from Infographics. In: International Conference on Pattern Recognition. pp. 361–380. Springer (2021)
10. Davila, K., Urala Kota, B., Setlur, S., Govindaraju, V., Tensmeyer, C., Shekhar, S., Chaudhry, R.: ICDAR 2019 Competition on HArvesting Raw Tables from Infographics (CHART-Infographics). In: ICDAR. IEEE (2019)
11. Davila, K., Xu, F., Ahmed, S., Mendoza, D.A., Setlur, S., Govindaraju, V.: ICPR 2022-Challenge on HArvesting Raw Tables from Infographics. In: International Conference on Pattern Recognition. IEEE (2022)
12. Ding, J., Xue, N., Long, Y., Xia, G.S., Lu, Q.: Learning roi transformer for oriented object detection in aerial images. In: CVPR. pp. 2849–2858 (2019)
13. Du, Y., Chen, Z., Jia, C., Yin, X., Zheng, T., Li, C., Du, Y., Jiang, Y.G.: Svtr: Scene text recognition with a single visual model. In: Raedt, L.D. (ed.) International Joint Conference on Artificial Intelligence. pp. 884–890. International Joint Conferences on Artificial Intelligence Organization (7 2022)
14. Fang, S., Xie, H., Wang, Y., Mao, Z., Zhang, Y.: Abinet: Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition pp. 7098–7107 (2021), https://arxiv.org/abs/2103.06495
15. Han, J., Ding, J., Xue, N., Xia, G.S.: Redet: A rotation-equivariant detector for aerial object detection. In: CVPR. pp. 2786–2795 (2021)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
17. Hindenes, L.B., Håberg, A.K., Johnsen, L.H., Mathiesen, E.B., Robben, D., Vangberg, T.R.: Variations in the circle of willis in a large population sample using 3d tof angiography: The tromsø study. PLoS One **15**(11), e0241373 (2020)
18. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: ICCV. pp. 1314–1324 (2019)
19. Jiang, H., Xu, Y., Cheng, Z., Pu, S., Niu, Y., Ren, W., Wu, F., Tan, W.: Reciprocal feature learning via explicit and implicit tasks in scene text recognition (2021), https://arxiv.org/abs/2105.06229
20. Jobin, K., Mondal, A., Jawahar, C.: Docfigure: A dataset for scientific document figure classification. In: ICDARW. vol. 1, pp. 74–79. IEEE (2019)
21. Jocher, G., Chaurasia, A., Qiu, J.: Ultralytics YOLO (Jan 2023), https://github.com/ultralytics/ultralytics
22. Kafle, K., Price, B., Cohen, S., Kanan, C.: Dvqa: Understanding data visualizations via question answering. In: CVPR. pp. 5648–5656 (2018)
23. Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V.R., Lu, S., et al.: Icdar 2015 competition on robust reading. In: ICDAR. pp. 1156–1160. IEEE (2015)
24. Li, C., Liu, W., Guo, R., Yin, X., Jiang, K., Du, Y., Du, Y., Zhu, L., Lai, B., Hu, X., et al.: Pp-ocrv3: More attempts for the improvement of ultra lightweight ocr system. arXiv preprint arXiv:2206.03001 (2022)
25. Li, H., Wang, P., Shen, C., Zhang, G.: Show, attend and read: A simple and strong baseline for irregular text recognition. ArXiv **abs/1811.00751** (2019)
26. Liao, M., Wan, Z., Yao, C., Chen, K., Bai, X.: Real-time scene text detection with differentiable binarization. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11474–11481 (2020)
27. Liao, M., Zou, Z., Wan, Z., Yao, C., Bai, X.: Real-time scene text detection with differentiable binarization and adaptive scale fusion. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)

28. Liu, W., Chen, C., Wong, K.Y.K., Su, Z., Han, J.: Star-net: a spatial attention residue network for scene text recognition. In: BMVC. vol. 2, p. 7 (2016)
29. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021)
30. Luo, J., Li, Z., Wang, J., Lin, C.Y.: Chartocr: Data extraction from charts images via a deep hybrid framework. In: WACV. pp. 1917–1925 (2021)
31. Masry, A., Do, X.L., Tan, J.Q., Joty, S., Hoque, E.: ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In: Findings of the ACL. pp. 2263–2279. Dublin, Ireland (May 2022)
32. Methani, N., Ganguly, P., Khapra, M.M., Kumar, P.: Plotqa: Reasoning over scientific plots. In: WACV. pp. 1527–1536 (2020)
33. Park, J.K., Kim, K.Y., Sim, Y.W., Kim, Y.I., Kim, J.K., Lee, C., Han, J., Kim, C.U., Lee, J.E., Park, S.: Structures of three ependymin-related proteins suggest their function as a hydrophobic molecule binder. IUCrJ **6**(4), 729–739 (2019)
34. Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M., Heer, J.: Revision: Automated classification, analysis and redesign of chart images. In: ACM symposium on User interface software and technology. pp. 393–402 (2011)
35. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(11), 2298–2304 (2017)
36. Siegel, N., Horvitz, Z., Levin, R., Divvala, S., Farhadi, A.: Figureseer: Parsing result-figures in research papers. In: ECCV. pp. 664–680. Springer (2016)
37. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR. pp. 2818–2826 (2016)
38. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: ICML. PMLR, vol. 97, pp. 6105–6114. PMLR (09–15 Jun 2019)
39. Thiyam, J., Singh, S.R., Bora, P.K.: Chart classification: a survey and benchmarking of different state-of-the-art methods. IJDAR pp. 1–26 (2023)
40. Wang, W., Xie, E., Li, X., Hou, W., Lu, T., Yu, G., Shao, S.: Shape robust text detection with progressive scale expansion network. In: CVPR. pp. 9336–9345 (2019)
41. Wang, Y., Xie, H., Fang, S., Wang, J., Zhu, S., Zhang, Y.: From two to one: A new scene text recognizer with visual language modeling network. In: ICCV. pp. 14194–14203 (2021)
42. Yu, D., Li, X., Zhang, C., Han, J., Liu, J., Ding, E.: Towards accurate scene text recognition with semantic reasoning networks. CVPR pp. 12110–12119 (2020)
43. Yue, X., Kuang, Z., Lin, C., Sun, H., Zhang, W.: Robustscanner: Dynamically enhancing positional clues for robust text recognition. ECCV (2020)
44. Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., Liang, J.: East: an efficient and accurate scene text detector. In: CVPR. pp. 5551–5560 (2017)
45. Zhou, Y., Yang, X., Zhang, G., Wang, J., Liu, Y., Hou, L., Jiang, X., Liu, X., Yan, J., Lyu, C., Zhang, W., Chen, K.: Mmrotate: A rotated object detection benchmark using pytorch. In: ACM International Conference on Multimedia (2022)
46. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: Deformable transformers for end-to-end object detection. In: ICLR (2020)

CHART-Info 2024: A dataset for Chart Analysis and Recognition

Kenny Davila[1 [0000−0001−6308−7113]], Rupak Lazarus[2 [0009−0008−9482−3230]], Fei Xu[3 [0000−0002−9353−9528]], Nicole Rodríguez Alcántara[4 [0000−0002−4405−7063]], Srirangaraj Setlur[3 [0000−0002−7118−9280]], Venu Govindaraju[3 [0000−0002−5318−7409]], Ajoy Mondal[2 [0000−0002−4808−8860]], and C. V. Jawahar[2 [0000−0001−6767−7057]]

[1] School of Computing, DePaul University, Chicago IL 60604, USA
kdavila@depaul.edu
[2] International Institute of Information Technology, Hyderabad, India
rupak.lazarus@research.iiit.ac.in, {ajoy.mondal,jawahar}@iiit.ac.in
[3] CSE, University at Buffalo, Buffalo NY 14260, USA
{fxu3,setlur,govind}@buffalo.edu
[4] Facultad de Ingenieria, Universidad Tecnológica Centroamericana, Honduras
nicole.rodriguez@unitec.edu

Abstract. Charts are tools for data communication used in a wide range of documents. Recently, the pattern recognition community has shown interest in developing methods for automatically processing charts found in the wild. Following previous efforts on ICPR's CHART-Infographics competitions, here we propose a newer, larger dataset and benchmark for analyzing and recognizing charts. Inspired by the steps required to make sense of a chart image, the benchmark is divided into 7 different tasks: chart image classification, chart text detection and recognition, text role classification, axis analysis, legend analysis, data extraction, and end-to-end data extraction. We also show the performance of different baselines for the first five tasks. We expect that the increased scale of the proposed dataset will enable the development of better chart recognition systems.

Keywords: Charts · Dataset · Graphic Recognition.

1 Introduction

Charts are tools for data communication used in a wide range of documents. The pattern recognition community has displayed a significant interest in methods for analyzing and recognizing charts in the wild [8]. There are rules and regular patterns that define how data can be converted into charts of different types. Yet, current models still struggle with complex graphics, especially when these do not adhere strictly to these rules and conventions. In contrast, humans can still successfully interpret the data encoded in these images.

Recent years have seen the development of very deep networks which can solve a variety of tasks as long as they are trained with enough data. For chart

recognition, most available large-scale datasets are synthetic, often created using an artificial process to generate charts based on data from real sources [10,32]. However, users can employ many tools to create charts with diverse visual styles. Often, they do not just convert data tables arbitrarily into charts, but instead use domain knowledge to choose appropriate chart types and conventions and carefully communicate their message [8]. Therefore, using a single tool to create large synthetic datasets does little to capture the diversity of charts in the wild.

Large-scale datasets are also needed to evaluate chart recognition systems. The CHART-Infographics competitions [10,9,11] were proposed with the goal of becoming the go-to chart recognition benchmark. Earlier editions included synthetic datasets, but systems trained only on synthetic charts performed very poorly on real charts seen in documents in the wild. Therefore, recent editions of CHART-Info have focused on providing large chart datasets based on real charts. This work presents the next iteration of this effort, the *CHART-Info 2024* dataset, which provides a major increase in the amount of training data facilitating training of larger models. At the same time, we provide a brand new test set with non-disjoint splits to evaluate each task using far more images.

CHART-Info defines six functional tasks critical to the chart recognition process: Chart Image Classification (Task 1), Text Detection and Recognition (Task 2), Text Role Classification (Task 3), Axis Analysis (Task 4), Legend Analysis (Task 5), and Data Extraction (Task 6). To facilitate the development of task-specific models, each task receives as input the ideal outputs from some of the previous tasks. An additional task is formulated for end-to-end data extraction (Task 7), where only the chart image is provided. Tasks 6 and 7 must produce an approximation of the data table used to create the chart. In this work, we provide baselines for all tasks except 6 and 7. These baselines are based on open-source models, but multiple domain adaptations and heuristic rules have been used to make them work well on chart-specific tasks. We plan to release all chart images, annotations, evaluation tools and the custom baseline code. To get a sense of the complexity and diversity of charts in the dataset, reviewers can access a preliminary copy of it.[5].

2 Related Works

Multiple datasets have been created for different chart-related tasks. Earlier datasets were created by collecting chart images using web engines and manual filtering [34,5]. In this category we find the Revision dataset [34] (2,500 images of 10 chart types) and the dataset by Chagas et al. [5] (4,837 images of 10 chart types). However, web images are often available under restrictive licenses.

Other works have collected charts directly from data-oriented web sources. ExcelChart400k [30] was created by collecting Excel spreadsheets from the web. The dataset contains a total of 386,966 charts extracted from these spreadsheets

[5] https://www.dropbox.com/scl/fo/lk9csn1z2hnws8f1nbpjs/ANNPdRx5ChvoxITRHMO1b_o?rlkey=hlr7lxqhrrvy1njbnv0hsv4b5&dl=0

along with the tabular data used to create them (no manual annotation required). Nevertheless, all charts are created using a single tool, thus they lack visual diversity. The chart text was also replaced with random characters, affecting the ability to incorporate multi-modal methods that use text. Another example is the ChartQA [31] dataset which has 21,945 charts (mostly in vector format) of 3 types (bar, line and pie), extracted from 4 websites. Annotations for visual question answering (VQA) tasks were collected via crowdsourcing.

Some synthetic chart datasets have been proposed for tasks such as data extraction [3,10] and VQA [22,32]. Some important advantages of generating synthetic data include: better scalability, cleaner and more detailed annotations, and relatively low cost. The AdobeSynth dataset [10] has 202,550 chart images (10 classes) generated with Matplotlib using data from multiple web sources. Bajić and Job [3] used Plotly to create a dataset with over 120K images from 20 chart classes. The DVQA [22] dataset contains 300K images of synthetic bar charts generated with the matplotlib library. The PlotQA [32] dataset has 224,377 images (bar, scatter, and line plots) generated using an unspecified tool. DVQA and PlotQA provide millions of question-answer pairs.

Some recent efforts, including this work, have extracted and manually annotated images from scientific literature to create large-scale datasets. These have the advantage of being more reliably available than random images from the web, and many of them are available under licenses that allow their redistribution and even commercial usage. Two examples are the FigureSeer [36] dataset (60K images from 20K papers) and the DocFigure [20] dataset (33K images), but these are mostly designed for figure type classification (including many charts). The CHART-Infographics competitions have led to the creation of datasets with increasing scales. The dataset presented here is an major extension of our previous dataset from ICPR CHART-Info 2022 [11], which is based on real charts extracted from PubMed Central (PMC).

3 The CHART-Info 2024 Dataset

This work extends previous efforts from the CHART-Infographics competitions [10,9,11]. Every edition has provided a novel test dataset based on real charts. While the first edition provided AdobeSynth for training, the second edition [9] expanded the previous test dataset to create the first training dataset based on real charts. The third [11] merged previous datasets to form the new training dataset. The training dataset of CHART-Info 2024 merges previous [11] training and testing datasets, and provides the largest test dataset so far (See Table 1).

The novel test dataset was created using a similar methodology to the previous edition [11]. We selected papers added to the Open Access Section of the PMC between Dec. 2017 and Oct. 2021. We only considered papers released under *CC BY* or *CC-0* licenses containing the keywords "chart" or "plot" in their main text. Out of 241,396 papers matching these filters, we randomly selected 20K papers that were not already included in earlier versions of our dataset. A binary image classifier was used to identify chart candidates from all the figures

Table 1. Distribution of chart types on the training and testing datasets. Values are compared against earlier versions of the dataset.

	ICPR 2020 [9]		**ICPR 2022 [11]**		**ICPR 2024**	
Chart Type	Train	Test	Train	Test	Train	Test
Area	120	52	172	136	308	229
Line	7,401	3,155	10,556	3,400	13,955	5,142
Manhattan	123	53	176	80	256	68
Scatter	875	475	1,350	1,247	2,597	1,311
Scatter-Line	1,260	558	1,818	1,628	3,446	1,684
Pie	170	72	242	191	433	213
Vertical Box	316	447	763	775	1,538	802
Horizontal Bar	429	358	787	634	1,421	636
Vertical Bar	3,818	1,636	5,454	3,745	9,199	3,692
Horizontal Interval	109	47	156	430	586	326
Vertical Interval	342	147	489	182	671	202
Map	373	160	533	373	906	363
Heatmap	138	59	197	180	377	177
Surface	110	45	155	128	283	127
Venn	52	23	75	131	206	121
Total	15,636	7,287	22,923	13,260	36,182	15,093

Table 2. Statistics (min., median, max.) for different image attributes in our dataset.

	Image Width			Image Height			Image DPI			File Size		
Dataset	Min	Med	Max	Min	Med	Max	Min	Med	Max	Min	Med	Max
Training	76	709	10,800	24	486	6,000	28	600	2,400	2,577	49,999	7,986,057
Testing	118	714	6,299	66	490	7016	72	300	2400	3,036	54,506	2,859,605

in these papers. Then, these candidates were manually classified by chart type. Note that all images in our dataset are in JPEG format, just as provided by the PubMed Central. As a result, our dataset has the limitation of not including any images in vector formats. However, we note that all images in vector format can be easily rasterized to be recognized using models trained with raster images.

The original creators of the images in our dataset used a variety of tools to make them leading to diverse quality and sizes as shown in Table 2. Training set images go from 76×75 to $10,800 \times 6,000$ pixels, while testing set images go from 120×66 to $6,299 \times 6,299$ pixels. This is a challenge for vision models expecting fixed resolutions. For task such as image classification, it might be better to downsize large images, but details that help to differentiate between challenging pairs (e.g. line vs. scatter-line) might be lost when the images are shrunk (see Figure 2.b.). For other task such as text recognition, higher resolution images are better because the text is more readable. However, the variability in the relative scales of text and other chart objects can make detection tasks harder.

Data annotation was a collaborative effort between teams at 3 universities: University at Buffalo (USA), IIIT-Hyderabad (India), and UNITEC (Honduras).

Table 3. Available charts for training and testing per task.

Dataset	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
Training	36,182	8,343	8,343	6,965	7,065	5,427	5,427
Testing	15,093	3,280	3,280	3,128	3,128	2,676	2,676

At each location, the annotators collected the ground truth (GT) in 3 sequential stages: *Image class* annotation, *Text* annotation, and *legend, axes and data* annotation. At every stage, annotators were assigned images in batches, and used our publicly available tools [6] to annotate them. The first two stages were semi-automatic, because recognition systems were used to generate initial labels. Then, the annotators had to verify and correct these labels, effectively cutting down the annotation time in half compared to previous years. For quality control, every annotation at every stage had to be approved by one validator who enforced different rules to achieve consistent annotations.

Table 3 shows the number of charts available for training/testing per task. Because of the competition context, previous CHART-Info testing datasets used 5 disjoint splits to evaluate different tasks [9,11]. However, CHART-Info 2024 is an offline evaluation benchmark, eliminating the need for disjoint splits, providing far more data to evaluate each task. Based on the availability of GT per chart, we propose 4 non-disjoint splits for training and testing specific tasks: *DS1*, *DS2*, *DS3* and *DS4*. Split *DS1* is basically the entire dataset (column *Task 1* in Table 3), which can only be used for Task 1. Split *DS2* represents all charts that have GT for task 2 (column *Task 2* in Table 3), used for task 2 only. Tasks 3, 4 and 5 share the same inputs, and Split *DS3* only considers the charts that have GT for the three tasks: 6,957 for training and 3,128 for testing. Finally, Split *DS4* includes all fully annotated charts that can be used to evaluate Tasks 6 and 7 (column *Task 6* in Table 3). All results presented in Section 4 are based on the following protocol: The training portion of each data split is further divided into 80% for training and 20% for validation. Then, the test portion of the same data split is used for evaluation.

4 Tasks and Baselines

In this section, we will describe the inputs, outputs, evaluation metrics and baselines for each task supported by CHART-Info 2024. Based on existing chart recognition systems [8], these tasks are defined in a sequence: chart image classification (Section 4.1), detection and recognition of text (Section 4.2), text role classification (Section 4.3), axes analysis (Section 4.4), legend analysis (Section 4.5), and chart data extraction (Section 4.6). Alternatively, one can design systems that handle the whole process in a end-to-end manner (Section 4.7).

[6] https://github.com/kdavila/ChartInfo_annotation_tools

4.1 Task 1. Chart Image Classification

Task description. The type of the chart in an image determines the rules used to interpret the data encoded in its graphical elements. Therefore, the first task is chart image classification, where every image in the dataset belongs to one of the classes listed in Table 1. We focus on chart types that are well represented in our data source, and some of these might be known under different names.

Inputs, Outputs and Metrics. The input is an image, and the output is a chart class. Evaluation is based on standard classification metrics such as the per-class recall, precision and F1 score. Table 1 shows that this is an unbalanced dataset, therefore we use the macro-average of the F1 scores for all classes as the final score to emphasizes the importance of correctly identifying all classes.

Baselines. We used image classification methods which have achieved good results on chart images in the past. We obtained the following results: ResNet-18 [16] (91.20%), ResNet-32 [16] (91.17%), ResNet-50 [16] (92.37%), Inception V3 [37] (92.07%), Xception [7] (92.98%), MobileNet-V3-Small [18] (89.30%), MobileNet-V3-Large [18] (91.40%), EfficientNet-B0 [38] (91.59%), EfficientNet-B1 [38] (92.14%), Swin-Tiny [29] (91.02%), and Swin-Base [29] (**93.60**%). The highest score is achieved by the more recent model based on transformers [29]. Nevertheless, earlier models such as ResNet-50 [16] still achieve competitive results, with F1 scores only 1.23% lower than the best model. This task can be hard for many reasons [39], and these numbers show that better models are still needed. Some images are hard to classify even for human annotators, like the scatter-line shown in Figure 2.b which can be easily confused with a line chart.

4.2 Task 2. Chart Text Detection and Recognition

Task description. Text is an important component of the semantics of a chart image. The goal of this task is to detect and recognize individual text blocks in the image. Unlike many scene text detection benchmarks, we are not interested in detecting isolated words, but rather we need to identify text blocks meant to be interpreted as a single unit (e.g. an axis title, a legend entry, etc.).

Inputs, Outputs and Metrics. The inputs are the chart image and its classification. The output is a list of text regions (*detection*), represented by quadrilaterals, and their corresponding transcriptions (*recognition*), represented by strings which might include LaTeX notation to handle special symbols and formulas found in charts. For each image, the *predicted texts* are matched against the *GT texts* if their $IoU \geq 0.5$, but a 1-to-1 matching constraint is enforced. The IoU values of matching texts are summed and the total is divided by $max(|predicted\ texts|, |GT\ texts|)$ to produce the *per-image IoU* score. For every GT text, we compute the normalized character error rate (NCER) between its transcriptions and the text of its corresponding prediction. Unmatched GT texts receive a NCER score of 0. All NCER values are summed and divided by $|GT\ texts|$ to produce the *per-image NCER* score. *Detection* and *Recognition* results are evaluated using the average of the *per-image IoU* and *per-image NCER*

scores, respectively. The final metric for Task 2 is the harmonic mean (f-score) of the *detection* and *recognition* scores.

Baselines for Detection. On average, charts have simpler backgrounds than natural scenes, but text regions can be really small, and they can overlap other graphical elements. One missing text region can have a huge impact in the overall accuracy of the whole chart recognition process. While scene text detectors often target isolated words, text in charts is better analyzed using coherent text regions (e.g. a complete data series name). This is hard considering that text regions in charts can go from one symbol to multiple lines of text.

We first considered two out-of-the-box baselines, Tesseract OCR engine v5.3.1 [1] and PaddleOCR engine v3 (PPv3) [24], which achieved average IoU scores of 0.3035 and 0.6022, respectively. These results show that, out of the box, they do not work well on charts probably because our target is text blocks which is not what these methods produce by default. These results already include some post-processing rules that helped but by only so much.

We then considered multiple baselines retrained with our data (*DS2*). Some of these models are based on the PPv3 framework [24], and these are pretrained on the ICDAR 2015 Scene Text Detection dataset [23]. We also considered models from the MMRotate framework [45], which were designed for rotated object detection in aerial images, and are pretrained on ImageNet-1k. We acknowledge that the accuracy of third-party re-implementations might be slightly different to the original models. Table 4 shows results for text detection baselines retrained on our dataset. In the case of PPv3 [24], configurations using the larger ResNet-50 [16] backbone achieved better results than their counterparts using MobileNet-V3 [18], although by a small margin in many cases. Surprisingly, while DB++[27] has produced better results than EAST [44] on natural scenes, EAST achieved better results here. Nevertheless, the best results were obtained using RoI Transformer [12] with Swin-Tiny [29] backbone, from the MMrotate framework [45]. This model might be the best in handling rotated text.

Baselines for Recognition. To make a fair comparison between text recognition baselines, we apply each recognition algorithm over text detection GT. During training and evaluation, the unicodeit Python library is used to convert the special LaTeX annotations into Unicode symbols. This library cannot handle all special symbols and formulas, but is appropriate for the vast majority of alphanumeric strings. We created a custom dictionary based on the 250 most common Unicode symbols in the training set to retrain some of our baselines.

Text recognition models often expect the inputs to contain a single horizontal line of text with at most so many characters. They do not work well with images of texts that are very long, rotated and/or multi-line. Multi-line text blocks are rare in our dataset, but long texts are very common. We use simple rules to identify long and/or multi-line text candidates, and then apply greedy algorithms to cut the image horizontally and/or vertically as required. The recognizer is used over each partition, and the results are concatenated.

Rotated text is common in charts, specially on *axis titles* and *tick labels*, and can be long and/or multi-line as well. All text regions (any rotation) are always

Table 4. Baselines for Chart Text Detection. Backbones include MobileNetv3 [18] (MN3), ResNet-50 [16] (RN50), ReResNet-50 [15] (RRN50) and Swin Tiny [29] (ST)

Method	IoU
DB [26] - MN3	0.7809
DB [26] - RN50	0.7866
PSENet [40] - MN3	0.8242
PSENet [40] - RN50	0.8263
DB++ [27] - RN50	0.7996
EAST [44] - MN3	0.8036
EAST [44] - RN50	0.8396
ReDet [15] - RRN50	0.8875
RoI Trans. [12] - RN50	0.8828
RoI Trans. [12] - ST	**0.8890**

Table 5. Baselines for Chart Text Recognition. These baselines are based on the PPv3 framework, were pretrained on ICDAR 2015 [23] and retrained on our dataset. Results are provided using *Line Splitting* (w LS) and *Without Line Splitting* (w/o LS).

Method	w/o LS NCER	w LS NCER
SAR [25]	0.9064	0.9202
SRN [42]	0.9098	0.9267
RobustScanner [43]	0.9133	0.9312
VisionLAN [41]	0.9132	0.9316
CRNN [35]	**0.9480**	0.9477
StarNet [28]	0.9293	0.9480
RFL [19]	0.9329	0.9506
ABINet [14]	0.9360	0.9528
SVTR [13]	0.9323	**0.9549**

Table 6. Baselines for Chart Text Detection and Recognition. We consider some Out-of-the-box (OOB) configurations, and the best configurations reported earlier.

Detection Method	Recognition Method	OOB	IoU	NCER	H-Mean
Tesseract [1]	Tesseract [1]	Yes	0.3035	0.3663	0.3320
PPv2 [24] DB [26]	PPv2 [24] CRNN [35]	Yes	0.5688	0.7074	0.6305
PPv3 [24] DB [26]	PPv3 [24] SVTR [13]	Yes	0.6022	0.7866	0.6821
RoI Trans. [12] Swin Tiny [29]	PPv3 SVTR [13]	No	**0.8890**	**0.9264**	**0.9073**

projected into axis-aligned rectangular regions, which can only have 0, +90, +180 or −90 degree rotations. The PPv3 framework includes an angle classifier but it did not perform well on chart text. Because of this, we used simple rules based on recognition confidence scores to simultaneously handle rotated text and long and/or multi-line text. When needed, this method tests multiple combinations of rotations with image splitting, and keeps the most confident transcription.

We considered three out-of-the-box baselines, Tesseract OCR [1], PPv2 [24] with CRNN [35], and PPv3 [24] with SVTR [13], which achieved NCER scores of 0.8483, 0.8226 and 0.8921, respectively. These baselines do not use any of our rules to handle special cases. We then experimented using our dataset to retrain multiple models from the PPv3 framework, and applying our rules to handle multi-line and rotated text. Table 5 shows the results for these models, considering whether the horizontal line splitting algorithm (LS) was used to deal with long text candidates or not. We found CRNN [35] to be the most robust in terms of handling long texts on its own. However, by using our horizontal line splitting algorithm, other methods achieved higher recognition results. The best recognition method was SVTR [13].

Complete Baselines. Table 6 shows the results for end-to-end chart text detection and recognition. Here, recognition results are affected by errors made

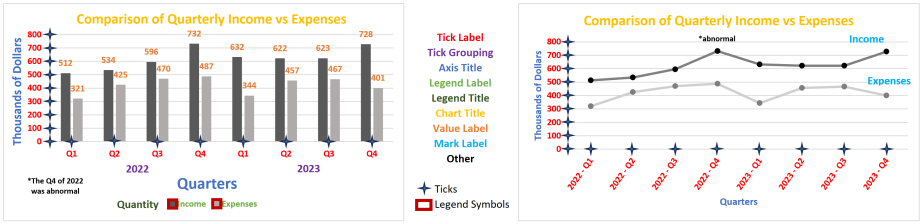


Fig. 1. Targets for multiple tasks. Different text colors are used to illustrate our text roles in two charts (Task 3). We also show the expected ticks using stars (Task 4), and red rectangles define the legend symbols (Task 5). Best seen in digital format.

by the detection model. We only consider out-of-the-box baselines, and the combination of the best models for detection (ROI Trans [12] with Swin-Tiny [29]) and recognition (PPv3 with SVTR [13]). This combined model is also using all of the intermediate rules used for handling rotated, multi-line and/or long texts.

4.3 Task 3. Chart Text Role Classification

Task description. This task aims to determine the role or function that each text region has on the chart. Our dataset considers 9 roles: *chart title*, *axis title*, *tick label*, *tick grouping*, *legend title*, *legend label*, *value label*, *marker label*, and *other*. This covers the categories needed to make sense of a chart image. The *other* category is used to group additional less common roles. These roles are illustrated in Figure 1.

Inputs, Outputs and Metrics. The inputs include the chart image and the GT outputs for Tasks 1 and 2. The expected output is a list of the roles for each GT text region. Like Task 1, evaluation is based on classification metrics.

Baselines. This task is similar to general object classification on images. However, two identical objects (text regions) can have different classes (roles) based on their position within the chart layout. Class imbalance also makes this task challenging, where *tick labels* make about 70.11% of all text regions in the training dataset, while *legend title*, *chart title*, and *tick grouping* are so rare that even combined represent just 1.15% of all text regions. However, all classes have the same impact on the final metrics.

We created our baselines for task 3 by training different object detector models using the role of each text region as their target class. We create variations of these models to simultaneously deal with multiple tasks. The first, *V35*, deals with tasks 3 and 5, and uses the original 9 text roles. The second, *V345*, deals with tasks 3, 4 and 5, and needs 12 roles for text. This is because it replaces the *tick label* class with 4 per-axis classes (more details in Section 4.4).

Task 3 requires assigning classes to text regions in the GT, but the predictions made by the object detectors might not align with the GT. Overlapping pairs of GT text regions and predictions are scored using the harmonic mean of their IoU and prediction confidence. We then greedily pick the highest scoring matches

Table 7. Baselines for Text Role Classification. Columns are F1 scores (%) for *tick label* (TL), *axis title* (AT), *legend label* (LL), *value label* (VL), *legend title* (LT), *mark label* (ML), *tick grouping* (TG), *other* (O), *Chart Title* (CT) and their macro average. For each baseline, we consider the V35 (†) and V345 (‡) variations.

Method	TL	AT	LL	VL	LT	ML	TG	O	CT	AVG
†ReDet [15] (R50 [16])	98.8	97.1	97.9	81.2	79.1	54.4	55.7	71.1	83.2	79.85
†RoI Trans [12] (R50 [16])	98.6	97.3	97.8	81.3	78.0	60.5	54.4	71.0	82.6	80.18
†RoI Trans [12] (Swin Tiny [29])	98.9	98.0	98.1	84.2	79.6	62.1	60.7	74.9	87.9	82.72
†YOLO-V8x [21]	98.8	98.1	97.3	84.7	85.3	63.0	70.7	74.5	89.9	84.71
†Deformable DETR [46]	99.4	98.6	98.6	83.8	87.1	63.8	70.6	75.9	91.3	85.45
†C. Mask R-CNN (Swin Base) [29]	99.2	98.7	98.6	85.4	89.6	68.7	70.9	77.3	92.6	**86.78**
‡ReDet [15] (R50 [16])	98.9	96.7	98.0	81.3	79.6	56.2	57.5	72.2	83.1	80.40
‡RoI Trans [12] (R50 [16])	98.7	97.4	98.1	81.3	75.7	60.2	56.3	71.8	82.6	80.22
‡RoI Trans [12] (Swin Tiny [29])	99.0	98.0	98.1	83.7	81.5	63.4	61.2	74.5	88.5	83.10
‡YOLO-V8x [21]	98.5	96.8	96.8	79.3	76.0	51.9	59.9	67.7	72.7	77.76
‡Deformable DETR [46]	99.4	98.6	98.8	84.2	87.4	66.3	74.9	75.9	92.1	86.40
‡C. Mask R-CNN (Swin Base) [29]	99.2	98.7	99.0	85.2	89.6	69.4	69.8	77.5	90.8	86.57

while enforcing a 1-to-1 matching constraint. The class of the prediction is finally assigned to each of the matched GT text regions. Unmatched predictions are simply ignored, and unmatched GT text regions are omitted.

Table 7 shows the results for Task 3. The *tick label*, *legend label* and *axis title* classes, which represent 87.16% of the training dataset, have very high F1 scores. Meanwhile, the *mark label* and *tick grouping* classes, which represent only 1.98% of the training dataset, have the lowest F1 scores. Except for YOLO-V8 [21], most models have very similar results for both variations. The Cascade Mask R-CNN model with Swin-Base transformer [29] and Deformable DETR [46] are consistently the strongest model from this set.

4.4 Task 4. Chart Axis Analysis

Task description. Axes in charts define the space of the chart data. The goal of this task is to locate the main chart axes (horizontal and vertical), and then link specific points in the axes (ticks) with text (tick labels). This location should be independent of the existence of visual tick marks.

Inputs, Outputs and Metrics. Inputs are the same as Task 3. The output is a dictionary organizing the tick positions by axis. Then, per axis, a set of pairs (text id, point) is expected. Each pair represents a *tick label* by their unique id in the GT, and the point represents the tick position.

Evaluation considers precision and recall of predicted ticks on the main axes (x-axis at the bottom and y-axis at the left). Secondary axes (top or right) are ignored. Predicted and GT ticks are matched by text id, and each match is weighted based on the distance between the GT location and the predicted location. First, the distances are normalized by the length of the image diagonal, and matches with distance ≥ 0.02 receive a weight of 0, and distance ≤ 0.01

Table 8. Baselines for Axes Analysis. All of them are based on Variation 345

Method	Rec. (%)	Prec. (%)	F1 (%)
ReDet [15] (R50 [16])	83.89	85.72	84.79
RoI Trans [12] (R50 [16])	84.09	85.27	84.67
RoI Trans [12] (Swin Tiny [29])	84.56	86.08	**85.31**
YOLO-V8x [21]	54.35	56.64	55.47
Deformable DETR [46]	**85.38**	85.18	85.28
Cascade Mask R-CNN (Swin Base) [29]	77.32	**86.46**	81.63

receive a weight of 1. For $0.01 < \text{distance} < 0.02$, an interpolated weight between 1 and 0 is used. Missing ticks and ticks associated with the wrong axis have weights of 0. The total weight of all matches is divided by the number of GT ticks to compute recall and by the number of predicted ticks to compute precision. Then, the overall recall and precision metrics are the macro averages of the per-axis values. Finally, we compute F1 score as the final per-chart score, and the average of the per-chart scores are computed for the entire evaluation set.

Baselines. While the GT text regions are known for this task, their corresponding roles are unknown. Because of this, our baselines work in combination with role predictions from Task 3, to identify all *tick labels*. The next challenge is to associate these to their corresponding axes. We use Variation V345 (see Task 4.3) which refines the *tick label* class by directly predicting if the region is a *tick label* of: *x-axis* (bottom), *y-axis* (left), *x2-axis* (top) or *y2-axis* (right).

The next step is to associate the *tick labels* to specific image locations. A common idea is to detect *tick marks* and use rules to match these to *tick labels*, but in many cases the *tick marks* are not visible or do not correspond to positions that should be associated with *tick labels*. To solve this problem, we add an *axes corner* object, which is a box of 10-by-10 pixels, centered at the origin of both x and y axes (bottom-left corner). The center of this box, (c_x, c_y), provides the coordinates shared by all ticks in a given axis (c_x for y axis, c_y for x axis). We use rules to determine the other coordinate using the rotated bounding box of the corresponding *tick label*. In most cases we simply use the center of the *tick label* bounding box: vertical center for y axis, horizontal center for x axis. Rotated *tick labels*, commonly found in the x axis, are the exception to this, and we use the x coordinate of the top-most point in their rotated bounding box.

Table 8 shows the results for this task. The performance for most object detectors is reasonably good considering that they do not detect visual tick marks. The baselines will fail when the *axes corner* box is incorrectly detected, or not detected at all. If multiple boxes are detected, the most confident is picked, but that can also produce incorrect outputs. Errors made in Task 3 (e.g., false positives/negatives for *tick labels*) are also propagated here. Also, *tick labels* associated with the wrong axis reduce precision for one axis, and recall for the other. Here, the ROI Trans model [12] achieved the highest score, with more consistent recall and precision levels than other models. The second best is Deformable DETR [46].

4.5 Task 5. Chart Legend Analysis

Task description. A legend is made by a set of *legend entries*, which are (*legend label*, *legend symbol*) pairs. Each *legend label* usually corresponds to one specific data series in the chart. The *legend symbol* exemplifies the appearance of the corresponding *data marks*. An example is shown in the left side of Figure 1. The goal of this task is to identify the *legend entry* pairs in the image.

Inputs, Outputs and Metrics. Inputs are the same as task 3. The output is a list of *legend entry* pairs (text id, bounding box), where the text id represents a *legend label*, and bounding box represents the associated symbol.

The evaluation of this tasks requires correct pairings between *legend labels* and *legend symbols*. For a given chart, predicted *legend entries* are initially matched to GT by the id of the *legend labels*. For each matching pair, the area of intersection between the GT *legend symbol* and the predicted one is computed and used to get two metrics: an IoU-based score (divide by the area of the union) and a recall-based score (divide by the area of the GT bounding box). The sum over all *legend entries* is computed for both scores, and then they are divided by the maximum between the number of GT *legend entries* and predicted *legend entries*. Finally, the average over the evaluation set is computed for both metrics.

Many charts have rather small and thin *legend symbols* (e.g, height of 2 pixels). IoU-score can be over-punishing on bounding boxes that correctly capture the *legend symbol*, but are slightly thicker. The recall-based score is also considered here because of this.

Baselines. Similar to Task 4, this task has access to the GT text regions, but the roles are unknown. All *legend labels* and *legend symbols* candidates need to be identified, and these need to be combined into *legend entry* pairs. As described before, we consider object detection baselines that combine multiple tasks on the same network. For task 5, we simply add the *legend symbol* objects. The same network will produce all *legend label* and *legend symbol* candidates.

The next challenge is to pair the candidates while considering false positives and negatives for both classes. Algorithms typically used for 1-to-1 matching in bipartite graphs will fail due to the noisy predictions. A simple approach is to pair each *legend symbol* with its closest *legend label*, but the way in which the distances are measured determines the quality of results. Based on the observation that for most charts, all *legend entries* have their symbols on the same side, we first estimate if all legend symbols in the image are left, right, above or below their labels. This direction is used to pick the corresponding edges of the bounding boxes of the legend labels, and their middle points are used as reference points for the labels. We then measure the distances between the reference points and the centers of the bounding boxes of the symbols. Matches are then sorted by increasing distance, and they are greedily picked in that order. Only matches between previously unmatched elements are accepted, and the process stops as soon as the first match involving a symbol or label previously matched appears. We do this to prevent spurious matches involving false positives/negatives based on the observation that valid legend entries in the same chart usually have similar edge distances between their symbols and labels.

Table 9. Baselines for Legend Analysis. We consider variations V35 (†) and V345 (‡).

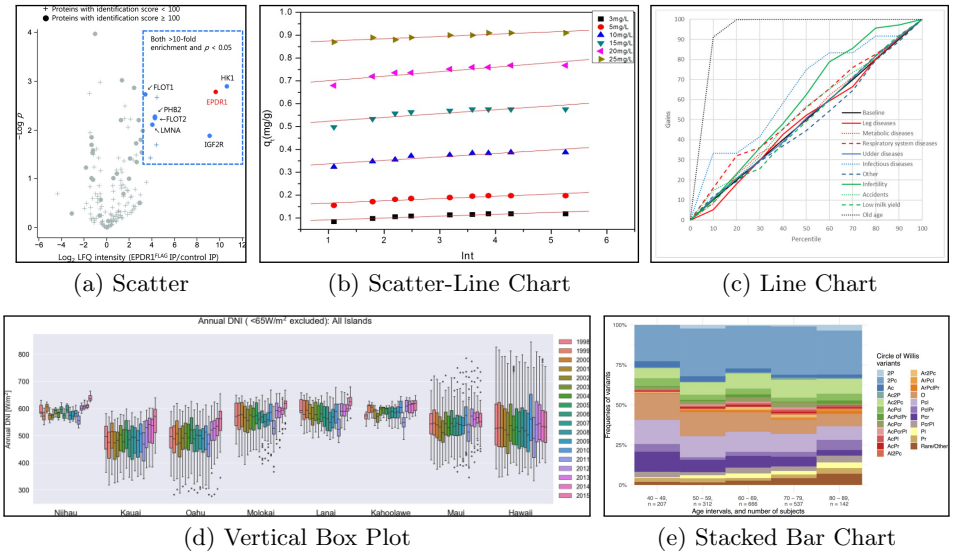
	Average BBox	
Method	IoU (%)	Recall (%)
†ReDet [15] (R50 [16])	83.09	95.33
†RoI Trans [12] (R50 [16])	83.41	95.62
†RoI Trans [12] (Swin Tiny [29])	**84.31**	95.56
†YOLO-V8x [21]	43.86	49.53
†Deformable DETR [46]	80.52	88.53
†Cascade Mask R-CNN (Swin Base) [29]	84.23	93.56
‡ReDet [15] (R50 [16])	83.66	95.57
‡RoI Trans [12] (R50 [16])	83.26	**95.69**
‡RoI Trans [12] (Swin Tiny [29])	83.84	95.42
‡YOLO-V8x [21]	43.42	49.31
‡Deformable DETR [46]	82.86	91.81
‡Cascade Mask R-CNN (Swin Base) [29]	84.12	93.62

Table 9 shows the results for this task. The recall-based scores show that most legend symbols are being detected and matched correctly, and that predicted boxes greatly overlap the symbol regions. It is possible that many of these predictions have the wrong thickness, leading to a much lower IoU-based scores in comparison. Errors can come from false positives/negatives of *legend symbols* and *legend label*. The method with the highest average F1 score for role classification achieves 98.6 F1 score for the *legend label* class (see Table 7). Therefore, it is likely that most errors come from failures to correctly detect the legend symbols. The baselines based on RoI Trans [12] consistently achieve some of the best scores for this task.

4.6 Task 6. Chart Data Extraction

Task description. This task approximates the data table used to create a chart image. It is divided into two sub-tasks: Plot element detection and extraction (Task 6.a), and raw data extraction (Task 6.b). The first sub-task aims at correctly locating the data marks in the chart (e.g., lines in line charts, bars in bar charts, etc.). The second sub-task puts everything together (text, axes, legends, and data marks) to reconstruct the data encoded in the chart image. The chart type might be used to determine the right approach for this task.

Inputs, Outputs and Metrics. The inputs for this task are the outputs from all previous tasks (1-5). The outputs for Task 6.a depend on the chart type. Bar charts require a list of bounding boxes per bar. Line and Scatter plots require a list of points for each data series. Box plots require a tuple with the position of the components of each box (box top, box bottom, box median, top whisker, bottom whisker). For task 6.b, the output is a set of data series with name, and the list of data points (x, y) that make that data series, where x is the independent variable, and y is the dependent variable. Multiple metrics are considered depending on the type of chart (See [9,11] for details).



(a) Scatter (b) Scatter-Line Chart (c) Line Chart

(d) Vertical Box Plot (e) Stacked Bar Chart

Fig. 2. Examples of challenging charts in our dataset. (a) Scatter chart, extracted from [33]. (b) Scatter-line, extracted from [6]. (c) Line chart, extracted from [2]. (d) Vertical Box plot, extracted from [4]. (e) Stacked bar chart, extracted from [17].

Baselines. The baselines for this task are complex, requiring combination of results from previous tasks. Existing works on chart recognition typically focus on a single chart type [8]. Meanwhile, our dataset provides data annotations for: horizontal/vertical bar charts, vertical box plots, scatter plots and line charts. Providing baselines for each of these chart types is out of the scope of this work. However, we briefly discuss the implications and complexities of creating data extraction models for these chart types.

Horizontal/Vertical Bar Charts. This is arguably the simplest chart type for data extraction. Standard object detectors often work with anchors of predefined aspect ratios, and bars in some charts can be extremely narrow or long. Most charts use bars of solid colors, but some charts use complex texture patterns instead. After detecting the bars, the next challenge is to infer the values represented by them. Every bar must be correctly map to a category (x value), a data series (e.g., "legend entry"), and an absolute value (y value). The orientation of the chart defines which axis is the x value (independent variable) and which one is the y value (dependent variable). Many charts use stacked and/or grouped bars, which require associating multiple bars to a single category (e.g., by proximity). Bars are usually associated to particular data series using legend analysis and their appearance. Finally, the y values of the bars are inferred through axes analysis and their spatial location. This process gets slightly more complicated for stacked bars where two extreme points are required to infer the value of each bar. Also, data points with y=0 often lead to *invisible bars*, but

their existence might be inferred by analyzing the chart layout. Figure 2.e shows an example of a complex stacked vertical bar chart included in our dataset.

Vertical Box Plots. The first step is to detect the boxes and their corresponding whiskers and median lines. Object detectors can be used to locate the boxes, but extreme aspect ratios can be a challenge. Whiskers and median lines might not be visible when their values are too close to the values represented by the top and/or bottom of the boxes. Similar to bar charts, we can find grouped box plots, where multiple boxes share a single category and each box needs to be linked to a particular legend entry. There are five points of interest in each box that need to be correctly mapped to their corresponding values in the y-axis. Figure 2.d shows a complex grouped vertical box plot included in our dataset. Note that horizontal box plots can be processed in a similar way, but we excluded them from our dataset because they are quite rare in practice. Also, many box plots include outliers represented by scatter marks, but we decided to currently exclude these from our benchmark because even human annotators often struggle to distinguish and recognize all individual points.

Scatter Plots. First, each data mark representing a data point must be detected. This might be easy when there are only a few large data marks, and extremely difficult when the plot region is cluttered with many small data marks. In the worst case scenario, we might only approximate the distribution of the original data used to create the plot. While our dataset includes all kinds of scatter plots, we only annotated data in cases where human annotators could accurately identify all individual data points. Data marks can be generated using all sorts of shapes, colors and sizes, making their automatic detection very challenging. After detection, each data mark needs to be mapped to a 2D point using their location and axis analysis. Legend analysis and the appearance of the data marks must be used to correctly map them to their corresponding data series. If no legend is present, then the appearance alone must be used to infer the existence of multiple data series. Figure 2.a shows an example of a complex scatter in our dataset where the colors on the legend do not match the colors in the plot region, and matching needs to be done using shapes, but this is also hard due to overlaps between data marks.

Line Plots. This is arguably the most challenging type here. Identifying the pixels of a given line is a segmentation problem which cannot be approached with basic object detectors. Some charts have solid colored lines which can be easily traced by basic segmentation algorithms. However, we find many charts with dashed lines and repeated line colors, where the thickness and/or dash patterns must be considered to differentiate them. Lines can also intersect and significantly occlude each other. Same as other chart types, lines must be associated with data series using legend analysis, but this can be difficult when the associated legend entries are very thin. Figure 2.c shows a line chart in our dataset which displays many of these issues. Some charts do not use legends, and instead directly provide names for each line using colored text within the plot region (e.g. *data mark label*) as illustrated on the right side of Figure 1.

4.7 Task 7. End-to-End Data Extraction

The end-to-end data extraction task has the same goals, outputs and metrics as sub-task 6.b (Section 4.6). However, the only input is the image of a chart. This task was designed for systems that can handle the chart recognition process as a whole, removing the need to produce and evaluate intermediate outputs. It is also possible to create a modular system which handles the recognition process using the individual tasks suggested in this benchmark. Not having any ground truth means that the real outputs from each module in the pipeline must be used. Any errors made in the earlier tasks will propagate to later tasks, affecting the overall recognition results. Similar to Task 6, baselines for this complex task are out of the scope of this work.

5 Conclusion

In this paper, we have introduced the CHART-Info 2024 dataset, a natural extension to the existing ICPR 2022 CHART-Info datasets [11]. Apart from a brand new test dataset, we have also provided a considerable number of baselines for the first 5 tasks defined in our dataset. These baselines provide a starting point for researchers interested in chart recognition. We believe that our dataset will enable the development of better chart recognition systems which will be capable of dealing with complexities found on charts in the wild.

References

1. Tesseract OCR – opensource.google.com, https://opensource.google.com/projects/tesseract
2. Adamczyk, K., Grzesiak, W., Zaborski, D.: The use of artificial neural networks and a general discriminant analysis for predicting culling reasons in holstein-friesian cows based on first-lactation performance records. Animals **11**(3), 721 (2021)
3. Bajić, F., Job, J.: Data extraction of circular-shaped and grid-like chart images. Journal of Imaging **8**(5), 136 (2022)
4. Bryce, R., Carreño, I.L., Kumler, A., Hodge, B.M., Roberts, B., Martinez-Anido, C.B.: Annually and monthly resolved solar irradiance and atmospheric temperature data across the hawaiian archipelago from 1998 to 2015 with interannual summary statistics. Data in Brief **19**, 896–920 (2018)
5. Chagas, P., Freitas, A., Daisuke, R., Miranda, B., De Araújo, T.D.O., Santos, C., Meiguins, B., De Morais, J.M.: Architecture proposal for data extraction of chart images using convolutional neural network. In: 21st IV. pp. 318–323. IEEE (2017)
6. Chen, J., Cai, Y., Clark, M., Yu, Y.: Equilibrium and kinetic studies of phosphate removal from solution onto a hydrothermally modified oyster shell material. PLoS One **8**(4), e60243 (2013)
7. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of CVPR. pp. 1251–1258 (2017)
8. Davila, K., Setlur, S., Doermann, D., Bhargava, U.K., Govindaraju, V.: Chart mining: a survey of methods for automated chart analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(11), 3799–3819 (2020)

9. Davila, K., Tensmeyer, C., Shekhar, S., Singh, H., Setlur, S., Govindaraju, V.: ICPR 2020-Competition on HArvesting Raw Tables from Infographics. In: International Conference on Pattern Recognition. pp. 361–380. Springer (2021)
10. Davila, K., Urala Kota, B., Setlur, S., Govindaraju, V., Tensmeyer, C., Shekhar, S., Chaudhry, R.: ICDAR 2019 Competition on HArvesting Raw Tables from Infographics (CHART-Infographics). In: ICDAR. IEEE (2019)
11. Davila, K., Xu, F., Ahmed, S., Mendoza, D.A., Setlur, S., Govindaraju, V.: ICPR 2022-Challenge on HArvesting Raw Tables from Infographics. In: International Conference on Pattern Recognition. IEEE (2022)
12. Ding, J., Xue, N., Long, Y., Xia, G.S., Lu, Q.: Learning roi transformer for oriented object detection in aerial images. In: CVPR. pp. 2849–2858 (2019)
13. Du, Y., Chen, Z., Jia, C., Yin, X., Zheng, T., Li, C., Du, Y., Jiang, Y.G.: Svtr: Scene text recognition with a single visual model. In: Raedt, L.D. (ed.) International Joint Conference on Artificial Intelligence. pp. 884–890. International Joint Conferences on Artificial Intelligence Organization (7 2022)
14. Fang, S., Xie, H., Wang, Y., Mao, Z., Zhang, Y.: Abinet: Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition pp. 7098–7107 (2021), https://arxiv.org/abs/2103.06495
15. Han, J., Ding, J., Xue, N., Xia, G.S.: Redet: A rotation-equivariant detector for aerial object detection. In: CVPR. pp. 2786–2795 (2021)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
17. Hindenes, L.B., Håberg, A.K., Johnsen, L.H., Mathiesen, E.B., Robben, D., Vangberg, T.R.: Variations in the circle of willis in a large population sample using 3d tof angiography: The tromsø study. PLoS One **15**(11), e0241373 (2020)
18. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: ICCV. pp. 1314–1324 (2019)
19. Jiang, H., Xu, Y., Cheng, Z., Pu, S., Niu, Y., Ren, W., Wu, F., Tan, W.: Reciprocal feature learning via explicit and implicit tasks in scene text recognition (2021), https://arxiv.org/abs/2105.06229
20. Jobin, K., Mondal, A., Jawahar, C.: Docfigure: A dataset for scientific document figure classification. In: ICDARW. vol. 1, pp. 74–79. IEEE (2019)
21. Jocher, G., Chaurasia, A., Qiu, J.: Ultralytics YOLO (Jan 2023), https://github.com/ultralytics/ultralytics
22. Kafle, K., Price, B., Cohen, S., Kanan, C.: Dvqa: Understanding data visualizations via question answering. In: CVPR. pp. 5648–5656 (2018)
23. Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V.R., Lu, S., et al.: Icdar 2015 competition on robust reading. In: ICDAR. pp. 1156–1160. IEEE (2015)
24. Li, C., Liu, W., Guo, R., Yin, X., Jiang, K., Du, Y., Du, Y., Zhu, L., Lai, B., Hu, X., et al.: Pp-ocrv3: More attempts for the improvement of ultra lightweight ocr system. arXiv preprint arXiv:2206.03001 (2022)
25. Li, H., Wang, P., Shen, C., Zhang, G.: Show, attend and read: A simple and strong baseline for irregular text recognition. ArXiv **abs/1811.00751** (2019)
26. Liao, M., Wan, Z., Yao, C., Chen, K., Bai, X.: Real-time scene text detection with differentiable binarization. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11474–11481 (2020)
27. Liao, M., Zou, Z., Wan, Z., Yao, C., Bai, X.: Real-time scene text detection with differentiable binarization and adaptive scale fusion. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)

28. Liu, W., Chen, C., Wong, K.Y.K., Su, Z., Han, J.: Star-net: a spatial attention residue network for scene text recognition. In: BMVC. vol. 2, p. 7 (2016)
29. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021)
30. Luo, J., Li, Z., Wang, J., Lin, C.Y.: Chartocr: Data extraction from charts images via a deep hybrid framework. In: WACV. pp. 1917–1925 (2021)
31. Masry, A., Do, X.L., Tan, J.Q., Joty, S., Hoque, E.: ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In: Findings of the ACL. pp. 2263–2279. Dublin, Ireland (May 2022)
32. Methani, N., Ganguly, P., Khapra, M.M., Kumar, P.: Plotqa: Reasoning over scientific plots. In: WACV. pp. 1527–1536 (2020)
33. Park, J.K., Kim, K.Y., Sim, Y.W., Kim, Y.I., Kim, J.K., Lee, C., Han, J., Kim, C.U., Lee, J.E., Park, S.: Structures of three ependymin-related proteins suggest their function as a hydrophobic molecule binder. IUCrJ **6**(4), 729–739 (2019)
34. Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M., Heer, J.: Revision: Automated classification, analysis and redesign of chart images. In: ACM symposium on User interface software and technology. pp. 393–402 (2011)
35. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(11), 2298–2304 (2017)
36. Siegel, N., Horvitz, Z., Levin, R., Divvala, S., Farhadi, A.: Figureseer: Parsing result-figures in research papers. In: ECCV. pp. 664–680. Springer (2016)
37. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR. pp. 2818–2826 (2016)
38. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: ICML. PMLR, vol. 97, pp. 6105–6114. PMLR (09–15 Jun 2019)
39. Thiyam, J., Singh, S.R., Bora, P.K.: Chart classification: a survey and benchmarking of different state-of-the-art methods. IJDAR pp. 1–26 (2023)
40. Wang, W., Xie, E., Li, X., Hou, W., Lu, T., Yu, G., Shao, S.: Shape robust text detection with progressive scale expansion network. In: CVPR. pp. 9336–9345 (2019)
41. Wang, Y., Xie, H., Fang, S., Wang, J., Zhu, S., Zhang, Y.: From two to one: A new scene text recognizer with visual language modeling network. In: ICCV. pp. 14194–14203 (2021)
42. Yu, D., Li, X., Zhang, C., Han, J., Liu, J., Ding, E.: Towards accurate scene text recognition with semantic reasoning networks. CVPR pp. 12110–12119 (2020)
43. Yue, X., Kuang, Z., Lin, C., Sun, H., Zhang, W.: Robustscanner: Dynamically enhancing positional clues for robust text recognition. ECCV (2020)
44. Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., Liang, J.: East: an efficient and accurate scene text detector. In: CVPR. pp. 5551–5560 (2017)
45. Zhou, Y., Yang, X., Zhang, G., Wang, J., Liu, Y., Hou, L., Jiang, X., Liu, X., Yan, J., Lyu, C., Zhang, W., Chen, K.: Mmrotate: A rotated object detection benchmark using pytorch. In: ACM International Conference on Multimedia (2022)
46. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: Deformable transformers for end-to-end object detection. In: ICLR (2020)