

ICPR 2020 - Competition on Harvesting Raw Tables from Infographics

Kenny Davila¹[0000-0001-6308-7113], Chris Tensmeyer²[0000-0002-0761-3243],
Sumit Shekhar²[0000-0002-4794-1962], Hrituraj Singh²[0000-0003-3705-120X],
Srirangaraj Setlur¹[0000-0002-7118-9280], and Venu
Govindaraju¹[0000-0002-5318-7409]

¹ Department of Computer Science and Engineering, University at Buffalo, Buffalo,
New York, USA {kennydav,setlur,govind}@buffalo.edu

² Adobe Research {tensmeyer,sushekha,harsingh}@adobe.com

Abstract. This work summarizes the results of the second Competition on Harvesting Raw Tables from Infographics (ICPR 2020 CHART-Infographics). Chart Recognition is difficult and multifaceted, so for this competition we divide the process into the following tasks: Chart Image Classification (Task 1), Text Detection and Recognition (Task 2), Text Role Classification (Task 3), Axis Analysis (Task 4), Legend Analysis (Task 5), Plot Element Detection and Classification (Task 6.a), Data Extraction (Task 6.b), and End-to-End Data Extraction (Task 7). We provided two sets of datasets for training and evaluation of the participant submissions. The first set is based on synthetic charts (Adobe Synth) generated from real data sources using matplotlib. The second one is based on manually annotated charts extracted from the Open Access section of the PubMed Central (UB PMC). More than 25 teams registered out of which 7 submitted results for different tasks of the competition. While results on synthetic data are near perfect at times, the same models still have room to improve when it comes to data extraction from real charts. The data, annotation tools, and evaluation scripts have been publicly released for academic use.

Keywords: Chart Recognition · Competition · Chart Dataset.

1 Introduction

Visualizations can be helpful tools for the communication of complex ideas. Among these, we find statistical charts which are used to display data in a way that facilitates the observation of patterns that are otherwise much harder to perceive using tables. In many cases, charts are also the main source used to share raw data that is not made publicly available in any other formats. This is just one out of many reasons that have motivated past works in the field of Chart Recognition [19, 1, 7]. Despite significant work in this field, not many have used standardized comparisons against previous methods. This is in part due to the lack of large scale benchmarks which can be used for this purpose.

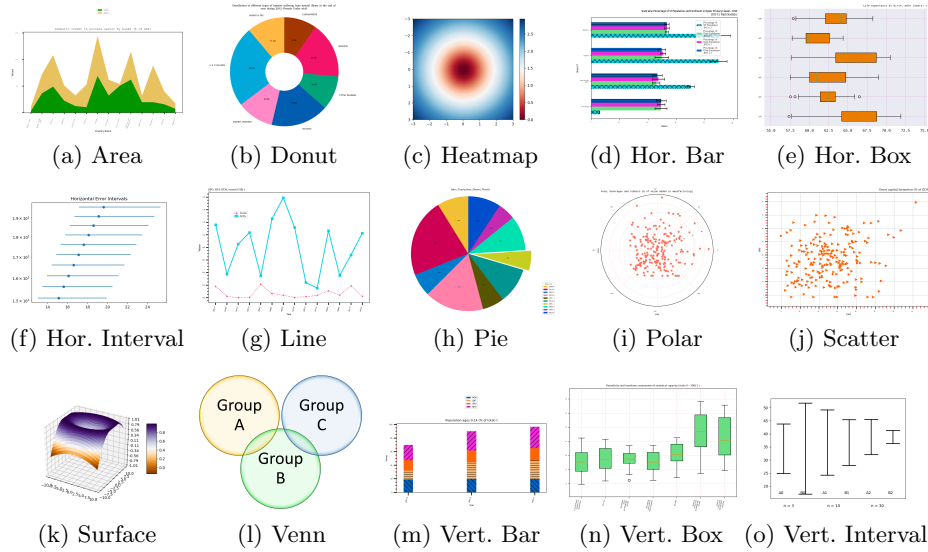


Fig. 1. A sample of 15 types of charts used in the competition.

This began to change with the first edition of the Competition on Harvesting Raw Tables from Infographics (ICDAR 2019 CHART-Infographics) [8], which provided both data and tools for the chart recognition community. This paper describes the continuation of this effort through the second edition of the CHART-Infographics competition at the International Conference on Pattern Recognition (ICPR) 2020.

In the second edition, we provided two pairs of datasets for training and evaluation of chart recognition systems (Section 2). The first pair of datasets, *Adobe Synth*, is based on synthetic charts generated from real data using matplotlib. The second pair of datasets, *UB PMC*, is based on manually annotated charts from PubMed Central³, a free full-text archive of biomedical and life sciences journal literature at the U.S. National Institutes of Health’s National Library of Medicine. We divide the Chart Recognition challenge into a pipeline of 6 tasks, and we also consider a seventh task to evaluate end-to-end systems (Section 3). We evaluate these tasks using revised versions of the metrics that were proposed in the first edition of the competition [8]. Many teams registered for this edition (Section 4), out of which 7 submitted results for different tasks (Section 5). We analyzed these results and the methods that generated them to provide our conclusions (Section 6) based on this competition. The data, annotation tools, and evaluation scripts have been publicly released for academic use⁴.

³ <https://www.ncbi.nlm.nih.gov/pmc/>

⁴ <https://chartinfo.github.io/>

Table 1. Distribution of chart types on the competition training and testing sets.

Chart Type	Adobe Synth.		UB PMC	
	Train	Test	Train	Test
Area	1,200	250	120	52
Line	1,200	250	7,401	3,155
Manhattan	0	0	123	53
Scatter	1,200	250	875	475
Scatter-Line	0	0	1,260	558
Polar	1,200	250	0	0
Donut	1,200	250	0	0
Pie	1,200	250	170	72
Horizontal Box	1,200	249	0	0
Vertical Box	1,200	250	316	447
Horizontal Bar (Grouped)	1,200	250	429	358
Horizontal Bar (Stacked)	1,200	250		
Vertical Bar (Grouped)	1,200	250	3,818	1,636
Vertical Bar (Stacked)	1,200	250		
Horizontal Interval	0	0	109	47
Vertical Interval	0	0	342	147
Map	0	0	373	160
Heatmap	0	0	138	59
Surface	0	0	110	45
Venn	0	0	52	23
Total	14,400	2,999	15,636	7,287

2 Data

We constructed two types of datasets for this competition: synthetic and real. For each type, we have provided both a training set that was released to participants upon registration and a corresponding testing set that was released to all participants at the same time. Participants were given one week to run their systems on the test data and return their results for evaluation by the organizers. There is a large overlap in terms of the classes covered by both types of datasets, but some classes are only included in one type (see Table 1). Both types are described in this section.

2.1 Synthetic Chart Dataset

We constructed the synthetic chart dataset using data from several public sources - (a) World Development Indicators⁵, (b) Gender Statistics (World Bank)⁶, (c)

⁵ www.datacatalog.worldbank.org/dataset/world-development-indicators

⁶ www.datacatalog.worldbank.org/dataset/gender-statistics

Government of India Open Data⁷, (d) Commodity Trade Statistics⁸, (e) US Census Data (for 2000 and 2010)⁹, (f) Price Volume Data for Stocks and ETFs¹⁰.

Chart Generation: Multiple charts of different categories were created using the Matplotlib library¹¹. The tabular data was first cleansed and converted into a common format. From each table, different sets of columns and randomized rows were selected to create the charts. To emulate features of real-world charts, we introduced variations in chart component such as: (1) positioning of titles, legends, and different stackings of legend entries (vertical/horizontal); (2) font families and sizes for different elements; (3) background/foreground text colors, color and/or width of lines, borders, grids, and markers; (4) bar widths and inner/outer radii of pies, exploded pies; and the (5) elements like plot text, minor/major axis ticks, tick direction, axis label variations. The statistics of different types of charts is shown in Table 1.

Chart Annotations: The required annotations for charts were obtained using functions provided by Matplotlib API including tight bounding boxes for text regions, axes, legends, and data elements (e.g. bars, lines, scatter plot markers, box plot elements and pies). For pie charts, tight masks were produced covering each pie region.

2.2 PubMedCentral Chart Dataset

A sample of real charts was extracted from the PubMed Central. The Open Access Section has more than 1.8 million papers, out of which we extracted more than 40,000 papers from different journals in fields such as epidemiology, public health, pathology, and genetics which typically contain many charts. This is the same sample of papers described in the first edition [8].

Data Sampling. We started with the 4242 chart images that were labeled for the first edition of this competition [8]. Using this as a seed, we trained a convolutional neural network using Triplet loss to learn an embedding for different types of charts. This network allowed us to sample images from regions of the embedded space that were not covered by the original sample. The newly sampled images were manually labeled first as either chart or non chart, and single-panel or multi-panel (4 possible combinations). We then chose the single panel chart images and sub-classified them into chart types. Using the newly annotated images and the original sample, we then retrained the embedding network and repeated this sampling process until we collected more than 65,000 images, out of which more than 22,500 were single panel charts. For some under-represented classes (e.g. vertical box charts and horizontal bar charts), we further found multi-panel image charts which contained them and manually labeled the

⁷ www.visualize.data.gov.in

⁸ www.kaggle.com/unitednations/global-commodity-trade-statistics/data

⁹ www.kaggle.com/muonneutrino/us-census-demographic-data/data

¹⁰ www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

¹¹ www.matplotlib.org/

Table 2. Distribution of charts in the testing set. All available charts on each dataset were split into 5 sets used to evaluate different tasks. Note that all pairs of splits need to be disjoint except the first and the last one where no ground truth is provided to the participants.

Split	Adobe Synth	UB PMC	Tasks
Split 1	1,240	5103	Task 1
Split 2	352	732	Task 2
Split 3	528	726	Tasks 3, 4, 5
Split 4	527	726	Tasks 6a, 6b
Split 5	352	726	Task 7
Total	2,999	7,287	

panels to split them into single panel chart images. These charts were further split into training and testing datasets as shown in Table 1. The testing set was further split into 5 disjoint sets used to evaluate different tasks each as shown in Table 2. This is necessary due to the fact that some tasks require inputs from the previous task.

Data Annotation. We further extended and improved our existing tools for the annotation of charts. We started with images which were already divided by chart types. We further split them into small batches (around 60 charts each) for further annotation. A small team of annotators was trained to use the tools that we created to produce a variety of annotations on the chart images.

First, they annotated the text regions on the chart by providing the location, transcription and role of each text region. Unlike the previous competition, this time we allowed text regions to be represented by quadrilaterals to handle rotated text properly. Annotators were only required to provide a rough but valid boundary for the text region and the tool included an option which further refined these text regions by fitting them to the text they contained using connected component analysis. The text itself was annotated in a semi-automatic fashion by first using the Tesseract OCR [23] to transcribe most of the selected text regions, and further manual error corrections when required. For special symbols and formulas, the annotators were required to provide \LaTeX strings.

The next step was legend and axes annotations. For legends, they had to mark tight boxes around the corresponding data marks for each legend entry. For axes, they had to provide a bounding box of the plot region which defines the position of the x and y axes. They also had to annotate the titles of each axes, tick labels, axis types (categorical vs numeric), and axis scales if they were numeric (linear, logarithmic, other). In addition, they had to provide the tick mark locations, types of tick marks (markers or separators), and associations between tick labels and tick marks. Finally, they had to annotate the data on specific types of charts (Line, Scatter, Horizontal/Vertical Box and Horizontal/Vertical Bar). For each type of chart, the tool was modified to provide different options which made the annotation process faster, allowing us to greatly upscale our datasets. The tool also now includes multiple options for quality control which basically attempts

to parse the chart based on the annotations provided and automatically finds and notifies the user of existing inconsistencies in the annotation. After that, all charts were again manually inspected to find and correct additional errors which might not have been captured by the automatic checks.

3 Tasks and Metrics

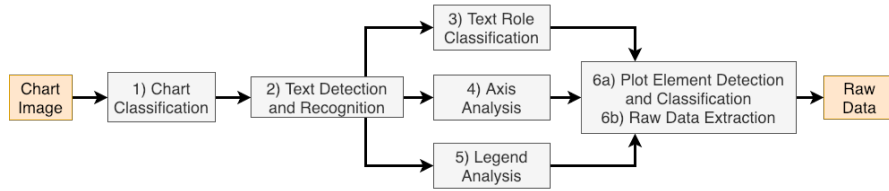


Fig. 2. The process of extracting data from chart images. The output from initial tasks can be used to inform later tasks.

Extracting data from chart images is a complex task. To allow the development of algorithms for specific parts of the process, we have broken it down into smaller tasks (Figure 2): 1) chart classification, 2) text detection and recognition, 3) text role classification, 4) axis analysis, 5) legend analysis, 6) data extraction, and 7) end-to-end data extraction. In this section, we describe each task and the metrics used for evaluation. For each task, we provide the ground truth output from previous tasks in order to analyze the performance of systems independently of errors made in previous tasks. Note that the implementations of all metrics are publicly available¹².

As mentioned in Section 2, we have two datasets with very different properties. We consider individual leaderboards per task per dataset for a total of 14 leaderboards. In order to select the competition winners, we use a point-based system where the method that obtains the top-1 rank on each task on each dataset is awarded 4 points, the top-2 rank gets 3 points, the top-3 rank gets 2 points and the remaining teams with a submission get 1 point for participation, while the rest who did not participate on that task or dataset get 0. We then add up all points across the leaderboards to create the overall ranking of participants.

3.1 Task 1. Chart Classification

In this task, chart images are classified into different types as described in Table 1. Note that some classes are only represented in the synthetic dataset and others are only represented in the UB PMC dataset. Most of these classes are only

¹² <https://chartinfo.github.io/>

considered for this task. For each class, we compute the precision and recall, and then take the harmonic mean (F-measure). We use the average of these values as the final score.

3.2 Task 2. Text Detection and Recognition

The input for the task is the chart image and its correct chart class. Then, text detection and recognition is performed at the logical element level. This means some multi-word and even multi-line elements such as titles and axis tick labels are treated as a single element. Previously, we used bounding boxes for text regions [8]. In this edition, we now use polygons, specifically quadrilaterals, to represent text regions.

During evaluation, the predicted text regions are compared to the ground truth quadrilaterals and they are considered a match if their intersection over union (IOU) exceeded 0.5. For many-to-one and one-to-many matches, the pair with the highest overlap is chosen and the remaining regions are counted as false negatives or false positives. Matched pairs are scored by Normalized Character Recognition Rate (NCRR) for recognition using $\max(1 - \text{NCER}, 0)$ as well as IOU for detection. Normalized Character Error Rate (NCER) is measured as the edit distance between ground truth string and predicted string normalized with respect to ground truth string length. Some regions include special symbols and even mathematical expressions which are all labeled using \LaTeX . At the moment we simply convert these annotations back to unicode strings which might not be ideal for some complex expression as well as symbols having both super and sub indices.

For each image, the maximum between ground truth text regions and predicted text regions is used to normalize the detection scores. For recognition scores, we simply use the number of ground truth boxes. The harmonic mean of detection and recognition scores is used as the final score per image, and the average of this metric is computed for the entire test as the final score for the task.

3.3 Task 3. Text Role Classification

In this task, participants are provided with the GT chart type and the correct text region-transcripts pairs, and their target is to classify each region based on their semantic roles. The classes considered are: *chart title*, *axis title*, *legend title*, *legend label*, *mark label*, *tick label*, *tick grouping*, *value label* and *other*. Submissions were evaluated using the average per-class F-measure (same as Task 1).

3.4 Task 4. Axis Analysis

In Task 4, systems must associate tick labels with pixel coordinates. This is a necessary step in order to convert the coordinates of chart data point geometries

from pixel space to the chart space represented by the axes. Similar to Task 3, the inputs are the chart image, its class and the text regions with transcriptions, but participants do not know which text regions are the tick labels. Then, for each axis the output is a list of text regions (tick labels), each paired with a (x, y) point representing the tick location in the image.

The evaluation is based on a weighted F-measure, where each tick mark can get a score between 0 and 1 based on location accuracy. Extra elements incorrectly predicted as ticks receive a score of 0. There are two thresholds used ($a = 1.0\%$ and $b = 2.0\%$) which are proportional to the chart image diagonal. With these, we score predictions located at a distance d from the GT tick location using:

$$s(d, a, b) = \begin{cases} 1, & \text{for } d \leq a \\ \frac{b-d}{b-a}, & \text{for } a \leq d \leq b \\ 0, & \text{for } d \geq b \end{cases} \quad (1)$$

Recall is computed as the sum of the scores divided by the number of GT ticks, and precision is the sum of scores divided by the number of predicted ticks.

3.5 Task 5. Legend Analysis

The goal of legend analysis is to pair textual legend labels with their corresponding graphical markers. Similar to Task 4, participants must produce a list of text regions, each paired with a bounding box that surrounds its corresponding legend marker. The input is same as Task 4 which means that all text regions are given without knowing which ones belong to the legend. The final score is a normalized IOU score, where true positive predictions must have the text element and a partial score is determined by either the recall or the IOU of the legend marker bounding box. Normalization is done by dividing the sum by the largest between the number of expected legend boxes and the number of predicted boxes, thus punishing for false positives as well. Previously, we only reported the IOU-based metric [8], but this time we are also reporting the Recall-based version.

3.6 Task 6. Data Extraction

Charts are often used to convey numerical data, and the main goal of this competition is to get an approximation of the original data (e.g. data points) used to generate the chart. We further divide this task into two sub-tasks where the inputs are the ideal outputs from all previous tasks.

Task 6.a Plot Element Detection and Classification All visual elements representing data must be localized in the chart. This includes bars, lines, points and boxes (with whiskers). For bar charts, the expected output is a set of bounding boxes representing each bar. For scatter plots, the output is a set of (x, y) points. For line plots, the output is a set of lines, where each line is represented

as an ordered sequence of points. For box plots, the output is a set of boxes, where each box is represented by a tuple with 5 elements corresponding to the min, third quartile, median, first quartile, and max values. We currently ignore outlier marks and assume the ends of the whiskers to be the min and max values.

For all chart types considered here, the metric involves a set of predicted objects that need to be aligned with a set of ground truth objects. We use a difference distance function for each chart type to define a matching score between a predicted object and the ground truth object. For predicted objects $P = \{p_i\}$ and GT objects $G = \{g_i\}$, we construct the $K \times K$ ($K = \max(|P|, |G|)$) cost matrix \mathbf{C} where C_{ij} encodes the cost of matching p_i with g_j . When $i > |P|$ or $j > |G|$, $C_{ij} := 1$ to denote that unmatched objects incur maximum cost. We then use the Hungarian algorithm [16] to solve the assignment problem defined by this cost matrix, which yields the optimal pairing of predicted and GT objects and c , the total cost of this pairing.

The final score (higher is better) for each image is $1 - \frac{c}{K}$. We then average each chart image's score across the entire test set. Further details on the distance functions per chart element are available in our competition website including the scripts used to compute these metrics.

Task 6.b Raw Data Extraction The output of this task is the data that was used to render the chart image. In the case of box plots, we only require the summary statistics (i.e., min, 1st, 2nd, 3rd quartiles, and max) as those are sufficient to re-create the plot. The output of this task is a set of data series objects, where a data series consists of a name (string) and a sequence of (x, y) values (order only matters for line plots). If the x-axis has a discrete domain, then we represent each x value as a string, otherwise it is a real value. All y values are real numbers. The name of a data series corresponds to the textual label found in the legend or mark. If there is no legend in the chart, then we check for mark labels and if these are also absent then the GT name for the data series is ignored for evaluation purposes.

To compare a set of predicted data series with a set of GT data series, we find the optimal pairing using the Hungarian algorithm [16] of data series under a data series distance function that compares both data series names and sets of points. Specifically, the distance D between data series is

$$D(n_1, d_1, n_2, d_2) = (1 - \beta_1)L(n_1, n_2)^\alpha + \beta_1 M(d_1, d_2) \quad (2)$$

where n_1 and n_2 are the names of the data series, L is the normalized Levenshtein Edit Distance, M is a distance function appropriate for the chart type, and d_1 and d_2 are the two sets of points. The hyperparameters $\alpha = 1$ and $\beta_1 = 0.75$ control the relative contribution of the point set comparison and the name comparison.

Each evaluation of M may also contain an assignment problem to find the optimal pairing of (x, y) points between d_1 and d_2 (as in Task 6a). When x is discrete, we use the normalized string edit distance, L . For continuous values, we use the Mahalanobis distance to normalize the dataset scales across images. For

line plots with continuous x values, instead of pairing points, we approximate the integral of the difference between the predicted and GT line, using linear interpolation to account for differences in the predicted and GT x values. For further details, please see the online supplementary materials and scripts to compute this metric.

3.7 Task 7. Raw Data Extraction

The desired output of this challenging task is the same as task 6b, except that the only input is the chart image (i.e., this is the entire pipeline end-to-end). In this case, idealized outputs of tasks 1-5 are not provided, so the method must perform all of these steps in series, which allows for errors to cascade. This task reflects real-world performance of systems and is scored using the same metric as 6b.

4 Participants

A total of 27 teams registered for the competition, out of which 7 finally submitted results for different subsets of tasks each. In this section, we briefly describe each team who made a submission to the competition.

1) *DeepBlueAI*: Zhipeng Luo, Zhiguang Zhang, Ge Li, Lixuan Che, Jianye He, and Zhenyu Xu from DeepBlue Technology and Peking University. *Task 1*: Employed a ResNet50 classifier [11] trained with a cross-entropy loss and label smoothing. *Task 2*: The detection model is a Cascade R-CNN model [2] with a ResNext-101 backbone [24], Deformable Convolution layers [5], and GCBLOCKS [3]. For text recognition, a CRNN model [22] trained with CTC loss [10] and data augmentation is applied to the detected text boxes after rotating them to make the text horizontal. *Task 3*: To classify text role, Random Forest and LightGBM [15] classifiers consume features derived from bounding box geometry, textual content, alignment with other text boxes, and position relative to the detected legend (if present). *Task 4*: A CenterNet [9] with a DLA-34 [26] backbone is used to detect tick locations via heatmap prediction. One branch of the network predicts the offset to the associated text label and matching is performed by L1 distance. *Task 5*: A CenterNet model is used to detect both legend graphics and legend pairs of graphics and legend text blocks. Then the Hungarian algorithm is used to assign graphics to text blocks. *Task 6*: CenterNet is used to detect rectangular elements and point elements are detected by a separate heatmap prediction model. To match chart elements with legend entries, they use similarity between HOG [6] features extracted from the chart elements and the legend entries. For charts without legends, K-means clustering is performed on HOG features to group chart elements.

2) *Magic*: Wang Chen, Cui Kaixu and Zhang Suyu from XinHuaZhiYunInc. and State Key Laboratory of Media Convergence. *Task 1*: A ResNet50 [11] was used for the Synthetic data and an ensemble of 10 ResNet152 models was used

for the PMC data. *Task 2*: They detect blocks of text using a Mask RCNN network [12] with a ResNeXt-152-FPN [17] backbone, deformable convolutions [5], and network cascades [2]. An attentional sequence to sequence model [20] composed of BLSTM layers and trained with numerous public datasets is used for text recognition. *Task 3*: They employed fusion approach of an object detector for 5 text classes and LayoutLM [25] for all text classes. *Task 5*: First the legend area was detected using a model similar to that of Task 2. Then the cropped legend image was fed to a Cascade RCNN [2] to find legend markers.

3) *Lenovo-SCUT-IntSig*: Hui Li, Yuhao Huang, Bangdong Chen, Luyan Wang, Kai Ding, Sihang Wu, Canyu Xie, June Lv, Wei Fei, Yan Li, Qianying Liao, Guozhi Tang, Jiapeng Wang, Xinfeng Chang, and Hongliang Li from SCUT, Lenovo, and IntSig. *Task 1*: An ensemble composed of 5 DenseNet-121 [13] models, a ResNet-152 [11], and a ResNet-152 with pyramid convolutions is trained using class-balancing techniques. *Task 2*: They use Cascade R-CNN for the Synthetic dataset and Cascade Mask R-CNN for PMC text block detection. Text blocks are split into lines before inputting them to a CRNN+CTC [22, 10] and attention model ensemble. *Task 3*: A weighted ensemble of 3 models that use text semantics, visual features, text features, chart type, and location features is trained with data augmentation.

4) *IntSig-SCUT-Lenovo*: Hesuo Zhang, Shuang Yan, Weihong Ma, Guangsun Yao, Adam Wu, Lianwen Jin from SCUT, Lenovo, and IntSig. *Task 5*: A Cascade Mask R-CNN [2] detects both legend marks and mark-text pairs and the pairing is performed using IoU of the detections. *Task 6*: A modified Pyramid Mask Text Detector [18] is used to detect bar boxes in bar charts, and a Gaussian heatmap regression model [4] is used to detect points in other charts. For task6b, they match the detected elements with legend entries and assign values by interpolation from the tick marks detected locations.

5) *SCUT-IntSig-Lenovo*: Weihong Ma, Hesuo Zhang, Guozhi Tang, Jiapeng Wang, Sihang Wu, Yuhao Huang, Hui Li, Canyu Xie, Kai Ding, Adam Wu, Qianying Liao, Ptolemy Zhang, and Yichao Huang from SCUT, Lenovo, and IntSig. *Task 4*: Tick detection is performed by Gaussian heatmap regression using a ResNet18 backbone with a deconvolution prediction layer. Heuristics are used to pair detected tick marks with tick labels. *Task 7*: A DenseNet-121 is first used to classify the chart type, and a Cascade RCNN [2] is used for text detection. For text line recognition, the C-RNN model [22] is trained with CTC loss [10]. Tick detection is performed the same as Task 4. Cascade RCNN is used to detect the legend box, the legend marks, and mark-text pairs. Then they detect and classify individual chart elements in the plot area. To associate chart elements with legend entries, they use K-means and RGB histogram features. Finally, real values are created by interpolating between detected tick marks.

6) *IPSA*: Mandhatya Singh and Puneet Goyal from Indian Institute of Technology Ropar. *Task 1*: DenseNet-121 [13] with an additional set of dense layers, batch normalization [14], and dropout layers have been used. *Task 2*: EAST [27] based text detection (with preprocessing) is used for detecting text boxes.

7) *PY*: Pengyu Yan from University of Buffalo. *Task 2*: Faster-RCNN [21] is used to detect text regions which are then recognized using the open source Tesseract OCR library. *Task 3*: Faster-RCNN is used to detect plot area, x-axis, y-axis, and legend area, and this information is used to classify text role. *Task 4*: Faster-RCNN is used to detect axis areas after which corner detection is used to localize tick marks. Pairing is performed with nearest bipartition rules.

5 Competition Results

5.1 Task 1. Chart Classification Results

Table 3. Task 1 results: Average F-measure across all predicted chart classes.

Team	Adobe Synth	UB PMC
Lenovo-SCUT-Intsig	1.00	0.928
DeepBlueAI	1.00	0.904
IPSA	0.992	0.863
Magic	0.994	0.905

Table 3 shows the average F-measure of each chart type prediction. For Adobe Synth, all participants achieved near perfect accuracy. The only confusions were among sub-categories of bar-charts. We consider DeepBlueAI and Lenovo-SCUT-Intsig tied for 1st with IPSA and Magic tied for 3rd since their performance difference is not statistically significant.

However, this task is not solved for the real world charts with all 4 systems achieving around 90% average F-measure across the classes. Lenovo-SCUT-Intsig is the clear winner for this dataset, while DeepBlueAI and Magic are tied for the second place.

5.2 Task 2. Text Detection and Recognition Results

Table 4. Task 2 results: Intersection over Union (IoU) for Text Detection; Normalized Character Recognition Rate NCRR for Text Recognition; Combined score is the harmonic mean of IoU and NCRR.

Team	Adobe Synth			UB PMC		
	IoU	NCRR	Combined	IoU	NCRR	Combined
Lenovo-SCUT-Intsig	0.943	0.973	0.958	0.741	0.765	0.752
Magic	0.929	0.922	0.926	0.722	0.735	0.729
DeepBlueAI	0.441	0.702	0.542	0.737	0.584	0.652
Py	-	-	-	0.676	0.606	0.639
IPSA	0.135	0.209	0.164	0.275	0.320	0.296

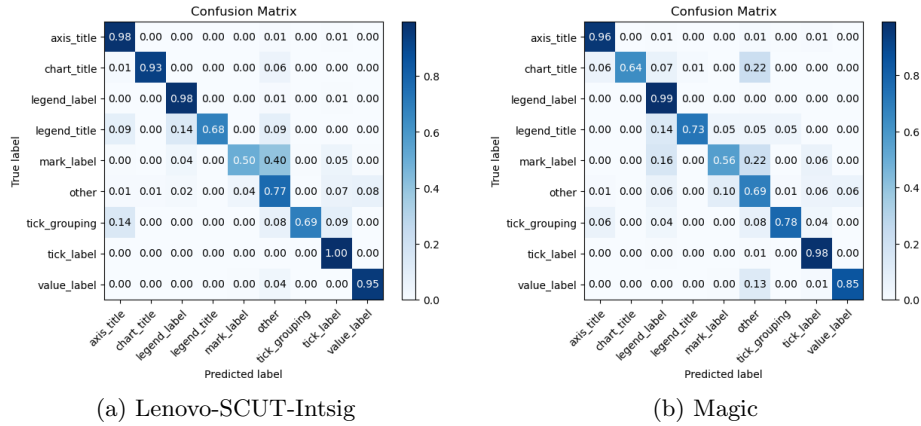


Fig. 3. Confusion matrices for Task 3 for the top-2 participants.

Table 4 shows the detection IoU and the NCCR for each submission for both datasets. The top 2 performers, Lenovo-SCUT-Intsig and Magic, used similar techniques, employing Cascade Mask RCNN models [2, 12], to achieve similar detection scores. DeepBlueAI also used a similar model and is competitive with the top performer on the UB dataset. The Py submission employed a Faster R-CNN [21], which is shown to not be as effective as the Cascade models at detecting small objects like text.

There was a bigger gap between the top-2 in text recognition with Lenovo-SCUT-Intsig using an ensemble of 2 models (CTC [10] and attention-based), and Magic using a seq-2-seq approach. Tesseract, which is not specifically designed for chart OCR, was used by Py and achieved a lower result, which may suggest that tuning models to perform OCR for charts is necessary for optimal performance.

5.3 Task 3. Text Role Classification Results

Table 5. Task 3 results: Average F-measure across all predicted text role classes.

Team	Adobe Synth	UB PMC
Lenovo-SCUT-Intsig	1.00	0.859
Magic	0.999	0.817
DeepBlueAI	0.999	0.772
Py	-	0.654

Table 5 shows the average F-measure for the classification of all text blocks in the charts. There is sufficient regularity in the text placement and style in the synthetic data that 3 participants were able to perfectly or near-perfectly classify all text blocks. In the PMC data, there is much greater variety. Both

Magic and Lenovo-SCUT-Intsig employed ensemble models and used features from pre-trained Language Models. The Lenovo-SCUT-Intsig team used hand-crafted features such as chart type, bounding box geometry, etc. as input to a shallow model while Magic used only deeply learned features from the image and text. The confusion matrices for both Lenovo-SCUT-Intsig and Magic are shown in Figure 3.

5.4 Task 4. Axis Analysis Results

Table 6. Task 4 results: Localization of tick marks for each tick label.

Team	Adobe Synth	UB PMC
DeepBlueAI	0.999	0.813
SCUT-IntSig-Lenovo	0.998	0.801
Py	-	0.588

Table 6 shows the results of Task 4 where tick labels are associated with the pixel location of the corresponding tick mark. We again observe near perfect accuracy on the synthetic data, which is not surprising given that all tick marks in the dataset are rendered using the same pattern of pixels. For the PMC data, DeepBlueAI performed best by a narrow margin over SCUT-IntSig-Lenovo, though both teams used deep models to regress Gaussian heatmaps around tick locations.

Accurate localization of tick marks is critical for performing the later task 6b correctly, since if the ticks are not pixel precise, mapping pixel locations of visual elements onto the axis values will be imprecise. The only exceptions are charts with categorical axes for which the actual tick marks might not be visually aligned with the actual tick values. Charts with this condition are common in the UB PMC dataset and logical tick marks were used to mark the best position within each axis to which the tick values need to be associated. Currently, the best result is 81% and no penalty was given for predictions within 1% (in terms of the image diagonal) of the target location, meaning that at least 19% of the predicted ticks were more than 1% away from the target. Since this task directly impacts the extracted data values, current methods are likely insufficient for real world applications of Chart Recognition and further work is needed on axis analysis.

5.5 Task 5. Legend Analysis Results

Table 7 shows results for legend analysis. We can see that in terms of the recall-based metric both IntSig-SCUT-Lenovo and Magic obtained near perfect scores, and they have a larger gap on the UB PMC dataset. However, in terms of the IOU-based metric, Magic achieved the highest scores on both datasets, including a near perfect score of 99% on the Synthetic dataset. Note that both methods are

Table 7. Task 5 results: Legend Analysis. We report both normalized Recall-based and normalized IOU-based scores.

Team	Adobe Synth		UB PMC	
	Recall	IoU	Recall	IoU
DeepBlueAI	0.928	0.919	0.864	0.818
IntSig-SCUT-Lenovo	0.997	0.950	0.932	0.849
Magic	0.993	0.990	0.920	0.860

based on Cascade RCNNs, but the method from Magic first cropped the legend region thus simplifying the recognition problem given that the legend region is correctly detected.

5.6 Task 6. Data Extraction Results

We break down and analyze the results for Task 6 by Subtasks 6a and 6b.

Table 8. Task 6a results: Detection of visual elements. Note that the PMC dataset did not differentiate between stacked and grouped bar charts, so results are presented only at the horizontal and vertical bar chart level.

Chart Type	Adobe Synth		UB PMC	
	DeepBlueAI	IntSig-SCUT-Lenovo	DeepBlueAI	IntSig-SCUT-Lenovo
Stacked Horz. Bar	0.962	0.957	0.929	0.870
Grouped Horz. Bar	0.963	0.917		
Stacked Vert. Bar	0.963	0.941	0.942	0.932
Grouped Vert. Bar	0.927	0.941		
Horz. Box	0.804	1.000	-	-
Vert. Box	0.832	1.000	0.977	0.989
Scatter	0.799	0.874	0.654	0.793
Line	0.976	0.992	0.842	0.847
All	0.907	0.954	0.870	0.882

Task 6.a Plot Element Detection and Classification Results Table 8 shows the results of Task 6a which received 2 submissions. DeepBlueAI used a CenterNet for detecting all elements, which outperformed the Pyramid Mask Text Detector [18] used by IntSig-SCUT-Lenovo on all but one bar chart category. However, the Gaussian regression model used by IntSig-SCUT-Lenovo on all other chart types was superior on non-bar charts. This was especially true on scatter plots, which had the lowest performance of all chart types.

Task 6.b Raw Data Extraction Results Table 9 shows the results of Task 6b, which with few exceptions are lower than that of Task 6a. This makes sense

Table 9. Task 6b results: Extraction of Data Series. Here we show the combined name and point set scores as described in Section 3.6 for each chart type, and show the average name score and data score for all charts. Note that the PMC dataset did not differentiate between stacked and grouped bar charts, so results are presented only at the horizontal and vertical bar chart level.

Chart Type	Adobe Synth		UB PMC	
	DeepBlueAI	IntSig-SCUT-Lenovo	DeepBlueAI	IntSig-SCUT-Lenovo
Stacked Horz. Bar	0.783	0.987	0.752	0.877
Grouped Horz. Bar	0.905	0.980		
Stacked Vert. Bar	0.261	0.981	0.649	0.913
Grouped Vert. Bar	0.882	0.981		
Horz. Box	0.934	0.992	-	-
Vert. Box	0.977	0.979	0.599	0.863
Scatter	0.419	0.867	0.232	0.710
Line	0.926	0.987	0.679	0.698
All	0.764	0.971	0.610	0.800
All-Name-Score	0.948	0.999	0.785	0.904
All-Data-Score	0.703	0.962	0.552	0.765

since solving task 6b entails taking the detections from Task 6a, associating them with legend entries (or inferring clusters if no legend is present), and then mapping the pixel-based detections to the value-space of the axes. It is interesting that IntSig-SCUT-Lenovo scores for bar charts actually increased w.r.t. Task 6a, which may be explained by the fact that Task 6a is scored based on bar bounding box IoU, but perhaps this can be explained because the width of vertical bars (and height of horizontal bars) is irrelevant for data extraction. This suggests that a better metric for Task 6a might only score bar bounding boxes based on the longer dimension rather than requiring precise localization of the shorter dimension.

For this task, the systems were furnished with the GT axes and legend information, which alleviates some of the difficulty, but still leaves some room for error. We see that systems degraded most on scatter plots, which is reasonable, given that overlapping scatter points are difficult to segment and properly associate with legend symbols.

It is indeed impressive that IntSig-SCUT-Lenovo was able to achieve a data score of 96.2% and a near perfect name score on Adobe Synth. However, the same system applied to real world data only achieved a data score of 76.5%, suggesting that a large difficulty gap still exists between the synthetic and real data and that training a chart recognition system on purely homogeneous synthetic data may not work well on real chart images. Interesting future work would examine the learning curve of the systems to gain a better understanding of how much further improvement could be made if more real data were to be annotated.

Table 10. Task 7 results for SCUT-IntSig-Lenovo, which submitted the only entry for this task. Scores are decomposed by chart type and by name-score and data-score.

Chart Type	Adobe Synth			UB PMC		
	Combined	Name	Data	Combined	Name	Data
Stacked Horz. Bar	0.960	0.978	0.954	0.742	0.796	0.724
Grouped Horz. Bar	0.912	0.965	0.894			
Stacked Vert. Bar	0.975	0.995	0.968	0.881	0.951	0.857
Grouped Vert. Bar	0.952	0.974	0.945			
Horz. Box	0.949	0.926	0.957	-	-	-
Vert. Box	0.969	0.948	0.976	0.660	0.662	0.660
Scatter	0.855	1.00	0.806	0.558	0.814	0.472
Line	0.960	0.975	0.955	0.660	0.762	0.626
Average	0.941	0.970	0.932	0.723	0.821	0.691

5.7 Task 7. Raw Data Extraction Results

Table 10 shows the results for the only submission, by SCUT-IntSig-Lenovo, for the end-to-end Task 7. Overall, the scores for bar charts are much higher than those of box, scatter, and line plots. For both datasets, scatter plots had the lowest data-scores, but were among the highest name-scores.

Surprisingly, the overall scores are not much lower than the best scores of Task 6b systems on UB PMC, which systems have the advantage of GT information for the output of Tasks 1-5. Given that other systems were able to get perfect results on Tasks 1 and 3-5 for Adobe Synth, it is reasonable for an end-to-end system to not degrade too much (e.g. 2%) w.r.t. Task 6b results. However, for UB PMC, the Task 1-5 results for other systems were much lower than for Adobe Synth, and generally in pipeline architectures such as the one designed for this competition (see Fig. 2), errors made on previous tasks tend to cascade and cause even more errors on downstream modules that rely on accurate inputs. We observe only an 8% decrease overall. Further analysis of the performance of the SCUT-IntSig-Lenovo system on Tasks 1-6 could help explain this finding; however, we only have their results for Task 4.

5.8 Final Ranking

The overall ranking for our competition is presented in Table 11. This table applies the scoring scheme described in Section 3, where winners of each task per dataset get 4 points, other teams can get at least 1 point for participation or 0 if they did not submit an entry for the task and dataset. Using this method, the overall winner of this competition is DeepBlueAI with an overall score of 35, Magic is second with 27 points, and third is Lenovo-SCUT-Intsig with 24 points. The ties over near perfect scores on Adobe Synth allowed multiple teams to get the 4 points on a few selected tasks. Meanwhile, the differences on the UB PMC were much larger allowing only one team to get the 4 points on each task.

Table 11. Overall Ranking: Total points scored per team across all tasks.

Team	UB PMC							Adobe Synth							Score
	1	2	3	4	5	6	7	1	2	3	4	5	6	7	
DeepBlueAI	3	2	2	4	2	3	0	4	2	4	4	2	3	0	35
Magic	3	3	3	0	4	0	0	3	3	4	0	4	0	0	27
Lenovo-SCUT-Intsig	4	4	4	0	0	0	0	4	4	4	0	0	0	0	24
SCUT-IntSig-Lenovo	0	0	0	3	0	0	4	0	0	0	4	0	0	4	15
IntSig-SCUT-Lenovo	0	0	0	0	3	4	0	0	0	0	0	3	4	0	14
IPSA	2	1	0	0	0	0	0	3	1	0	0	0	0	0	7
PY	0	1	1	2	0	0	0	0	0	0	0	0	0	0	4

6 Conclusion

In this paper, we have presented a summary of all activities of the second Competition on HARvesting Raw Tables from Infographics (CHART-Infographics). Compared to the previous edition [8], this year we provided upscaled datasets including a brand new training set based on real charts from the PubMed Central, and we also had participants for all tasks including task 6 and 7 which did not get any submissions in the previous edition. Consistent with the first version of the competition, we observed higher scores for the synthetic dataset including many near perfect ones, while the new testing dataset based on real charts remained the most challenging. The one submission we received for Task 7 scored fairly well in comparison with submissions for Task 6 which had access to additional ground truth data. We chose a brand new scoring scheme to rank participants across all tasks, and the overall winner this year is Team DeepBlueAI.

This year we also observed multiple methods for different tasks which were on average, far more complex than the ones we evaluated in the previous edition. However, despite the increased complexity of these methods, there are tasks where results are still far from ideal for large-scale applications. It might be possible that some of these methods will perform better in the future as more data becomes available. We hope that all data and tools and data that were produced during this competition will be valuable assets for future research in the field of chart recognition.

Acknowledgment: This material is based upon work partially supported by the National Science Foundation under Grant No. OAC/DMR 1640867.

References

1. Böschen, F., Beck, T., Scherp, A.: Survey and empirical comparison of different approaches for text extraction from scholarly figures. *Multimedia Tools and Applications* **77**(22), 29475–29505 (2018)
2. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 6154–6162 (2018)

3. Cao, Y., Xu, J., Lin, S., Wei, F., Hu, H.: Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 0–0 (2019)
4. Cheng, B., Xiao, B., Wang, J., Shi, H., Huang, T.S., Zhang, L.: Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. arXiv preprint arXiv:1908.10357 (2019)
5. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 764–773 (2017)
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05). vol. 1, pp. 886–893. IEEE (2005)
7. 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 pp. 1–1 (2020). <https://doi.org/10.1109/TPAMI.2020.2992028>
8. Davila, K., Kota, B.U., Setlur, S., Govindaraju, V., Tensmeyer, C., Shekhar, S., Chaudhry, R.: Icdar 2019 competition on harvesting raw tables from infographics (chart-infographics). In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 1594–1599. IEEE (2019)
9. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: Centernet: Keypoint triplets for object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6569–6578 (2019)
10. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on Machine learning. pp. 369–376 (2006)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
12. He, T., Tian, Z., Huang, W., Shen, C., Qiao, Y., Sun, C.: An end-to-end textspotter with explicit alignment and attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5020–5029 (2018)
13. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
14. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
15. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. In: Advances in neural information processing systems. pp. 3146–3154 (2017)
16. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)
17. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
18. Liu, J., Liu, X., Sheng, J., Liang, D., Li, X., Liu, Q.: Pyramid mask text detector. arXiv preprint arXiv:1903.11800 (2019)
19. Liu, Y., Lu, X., Qin, Y., Tang, Z., Xu, J.: Review of chart recognition in document images. In: Visualization and Data Analysis. p. 865410 (2013)

20. Mnih, V., Heess, N., Graves, A., et al.: Recurrent models of visual attention. In: Advances in neural information processing systems. pp. 2204–2212 (2014)
21. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
22. 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 (2016)
23. Smith, R.: An overview of the Tesseract OCR engine. In: International Conference on Document Analysis and Recognition. vol. 2, pp. 629–633. IEEE (2007)
24. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1492–1500 (2017)
25. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M.: Layoutlm: Pre-training of text and layout for document image understanding. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1192–1200 (2020)
26. Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2403–2412 (2018)
27. Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., Liang, J.: East: an efficient and accurate scene text detector. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 5551–5560 (2017)