

Data Extraction of Charts with Hybrid Deep Learning Model

Kirill Sviatov[1](✉), Nadezhda Yarushkina[1], and Sergey Sukhov[2]

[1] Ulyanovsk State Technical University, Ulyanovsk, Russia
{k.svyatov,jng,y.lapshov}@ulstu.ru
[2] Ulyanovsk Branch of the Institute of Radio Engineering and Electronics. V. A. Kotelnikov of Russian Academy of Science, Ulyanovsk, Russia

Abstract. This article describes an approach to automatic recognition of charts images using neural networks with hybrid deep learning model, which allows to extract data from an image and use this data to quickly find information, as well as to describe charts for visually impaired people. The key feature of this approach is the model of the recognition process, which includes classical algorithms for image analysis and deep learning models with flexible model tuning to improve the key quality indicators of recognition software.

Currently, the problem of chart recognition is usually solved in an interactive mode, which makes it possible to recognize in a semi-automatic way with a gradual refinement of the recognized data: "end-to-end" models of neural networks or pure computer vision algorithms cannot be used for complete recognition. This article describes an approach and models that use both deep learning models with attention and computer vision algorithms to accurately extract data from charts. This article describes an approach to recognizing only function charts with continuous lines, not pie or histograms. The resulting accuracy of using a deep learning network for localizing parts of charts is 72%, this is enough for recognition since post-processing algorithms significantly improve the final recognition accuracy.

Keywords: Artificial intelligence · Neural networks · Machine learning · Computer vision · Attention networks · Data extraction

1 Introduction

Currently, there are many scientific and technical events with video broadcasting, recordings of which are published in video services, materials in articles, and social networks without source codes of programs and data. When conducting research based on such materials, there is a problem of reproduction in terms of writing program code, as well as finding the data used, which is often presented in the form of function diagrams. Code and data publishing initiatives, such as "Papers with code" [24], are widespread among a narrow circle of researchers and are not widely used.

One of the ways to simplify the solution of the reproducibility problem is automatic graph recognition, which can also be used for visually impaired people when reading

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12957, pp. 382–393, 2021.
https://doi.org/10.1007/978-3-030-87013-3_29

articles. Additional motivation for writing this article is the need to develop technology that will allow visually impaired people to understand the meaning and data presented in the images.

In this paper, we consider the solution of five tasks for data extraction from the chart:

1. Automatic detection of the number of human-readable chart color. This task is important because to highlight the graph lines, it is important to use a separate color for each line. At the same time, upon a detailed examination of the graphs on video or from a camera, it often turns out that the same color can be represented by several colors. In this task, it is necessary to calculate how many human-readable colors are in the image. The following task is a color quantization of the source image.
2. After reducing the number of colors, it is necessary to find charts corresponding to a certain color.
3. Image text recognition. It is used to recognize all chart parts separately: title, labels, legend, axis scales.
4. Detection of the image components location. In this task, it is necessary to detect the position of the parts of the chart images in the coordinate system of the original image: axis labels, title, data labels, and data axes.
5. Chart recognition. It is a way to combine all the previously obtained analysis data in order to obtain a dataset describing the original cha image.

Chart classification, detection of the image, and perspective correction are outside the scope of this work. We only extract data from the image of a chart.

2 Literature Review

Despite the success of using neural networks in image processing, they usually perform only certain processing steps: object detection, localization in the image, and class prediction. It is also possible to determine all available objects in the image for subsequent analysis, for example, a geometric assessment of the relative position of objects, their size, and shape [29]. Despite these possibilities, there is a lot of information in the task of extracting data from charts, which makes the end-to-end approach inapplicable.

Automated data extraction from charts has been studied for many years, and this can be solved with four steps: localization of the chart on the image, chart classification, text detection, and data reconstruction. Our project focuses on the last two steps. In 2007, Prasad et al. [1] have described a classical computer vision method to classify charts in five categories, based on the Histograms of Oriented Gradients and the Scale Invariant Feature Transform descriptors for feature extraction and Support Vector Machine (SVM) for classification. Savva et al. [2] created a system called Revision to recognize charts, which includes chart classification, data extraction, and visualization. Low-level image features are extracted for chart classification by SVMs. There are automatic chart data extraction algorithms based on image processing and machine learning to extract data from chart images [8–11].

ReVision [12] is a tool that makes automatic classification of chart types (ten types) and extracts data from a chart image. iVoLVER [14] is also a tool that allows to extracts

data from an image. It relies on users inputs in both specifying data types (e.g., text, colors, shapes, etc.) and sampling data points in the image. Thus, it requires a relatively large number of interactions to extract data accurately from a chart image. DataThief [15] is a tool that extracts data from line charts. Users have to run it multiple times for a chart with multiple lines because it can extract one line at a time. People also have to specify six points (origin point, end-points of x-y axis, start- and end-point of the target line, and a point on the line).

WebPlotDigitizer [13] is a web-based tool that extracts data from four chart types (bar and line charts, polar and ternary diagrams) in automatic and manual modes. In the manual mode, people have to provide information for data extraction: labels, axis, and other chart components positions. Ycasd [16] is a tool for line chart data extraction with a manual technique similar to WebPlotDigitizer with which people have to specify all the data points in a line chart. In the task of chart recognition, there are many works done with regular computer vision technologies. Zhou et al. [3] use Hough lines transform and boundary detection to recognize bars. Huang et al. [4] use rules with edge maps. Deep learning models allow detect all the texts and chart components automatically with different models for each part of a model: Cliche et al. [5] use three object detection models. Poco et al. [6] use a CNN model to classify each pixel of an image as text or not and then exclude all non-text pixels.

The next step is data reconstruction. It also can be divided into two stages: text recognition and numerical data extraction. All detected texts are extracted by a text recognition model. Tick labels are combined with the coordinate positions to predict the coordinate-value relation. But another problem appears: matching objects between different categories, which has been studied by a few researchers. In the work of Siegel and Farhadi [7], this task is formulated as an optimal-path-finding problem with features combined from CNN model and pixels. The attention mechanism in neural networks [17] is popular now among researchers [18], because it allows us to interpret the results of the models and perform selection according to positions of objects in feature maps. Despite the fact that the technology appeared only several years ago, the range of applications is very wide [19, 28].

In this regard, it is necessary to use a hybrid approach for graph recognition: deep learning models use for chart elements localization, text extraction, and classical image analysis algorithms for data extraction.

In our work, an attention recurrent U-Net deep neural net is built to predict all objects locations. To recognize texts and numbers, Tesseract OCR software library was used. We also introduce brand new concepts and approaches to make the final data extraction.

3 Offered Approach

We offer a hybrid approach that uses different models for different stages of analysis. The common data flow structure is presented in the Fig. 1.

In the first step, the type of image is determined, one of several: bar graph - 1, chart - 2, block diagram - 3, line chart - 4 (the type of charts considered in this article), growth graph - 5, pie chart - 6, table - 7, just images - 0. This problem is solved using a neural network with the Resnet-152 architecture [25], the solution is based on the proposed solution [26].

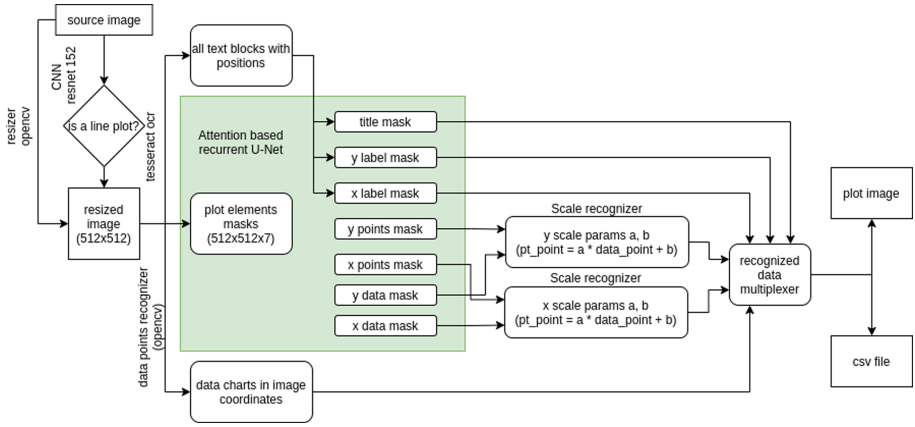

Fig. 1. Data flow of the recognition process

Second, the picture is resized by 512 × 512 pixels. The scale coefficient is calculated by the formula:

$$scale = \frac{512}{\max(width,\ height)}$$

where width and height are the original sizes of the source image. After that scaling, a new empty 512 × 512-pixel image is created and combine with the newly obtained.

The following 3 steps are performed in parallel mode:

- Text recognizing.
- Chart elements position detection.
- Chart data extraction in image coordinates.

4 Text Recognizing

This step is performed with Tesseract OCR software module which allow to extract data in the following format:

$$result = image_to_data(image, config),$$

Where *image* – source image, *result* – resulting dictionary, containing texts and their location on the source image, *config* – configuration parameters of the tesseract software.

After this recognition, we combine closely located words in single phrases. It is important for titles and labels.

5 Chart Components Localization

Chart components localization is very important part of the recognition process. It allows determining a location of all parts of the image: title, data, and labels for both horizontal

and vertical axes, tick points for horizontal and vertical axes, located on the lines. Without this location, important data cannot be extracted.

We use several models to detect the location of the most important parts of a chart: U-Net, Fast R-CNN, Attention Recurrent Residial U-Net (AR2U-Net) [27]. All models don't show meaningful results except Attention AR2U-Net model. It can generalize training set images. An example of such recognition is shown in Fig. 2.

The deep learning model performs the following transformation:

$$mask = recognize(image)$$
$$mask = <Tm, Xd, Yd, Xl, Yl, Xt, Yt>$$

The mask is the result of the recognition process by Attention recurrent AR2U-Net model. It is a set of masks, where Tm – title mask, Xd – horizontal data location, Yd – vertical data location, Xl – labels for horizontal axis, YL – labels for vertical axis, Xt – tick points for horizontal axis, Yt – tick points for vertical axis.

The example of the resulting mask is shown on the Fig. 2.

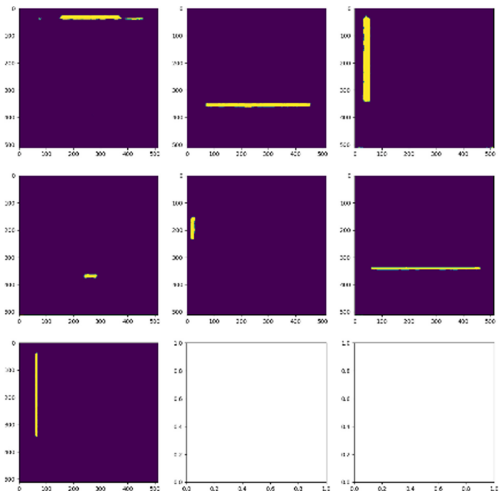

Fig. 2. Result of the inference process for image decomposition

The work [17] describes the problem of recognizing the relationship between objects in the image. The authors note that the variability of relations between objects is too large, therefore it is very difficult to create a one-hot sample to solve a similar problem. They offer an approach based on the search for patterns in textual descriptions of images. Such a network output can be considered weakly formalized, because human speech also has great variability, although textual descriptions of images are much easier to find [23]. The authors use a recursive network that iteratively analyzes objects in the scene and words in the text. However, to solve such a complex problem, the authors use the preliminary extraction of features from the image, namely, they select objects and spatial information between them.

The categorical cross-entropy loss function has a value of 72%, and this is a good result because very precise masking is not required for this task: post-processing algorithms significantly improve the final recognition accuracy.

6 Dataset Collection

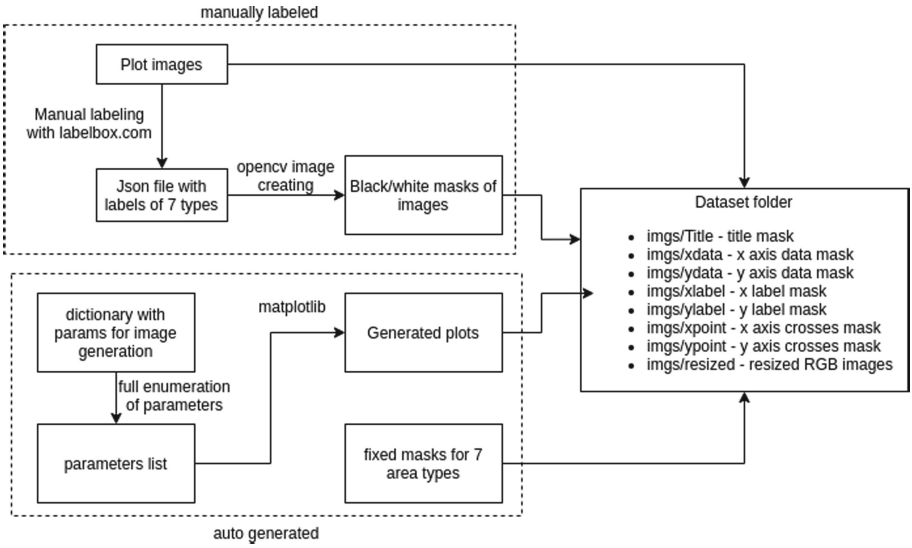

Fig. 3. Dataset generation and labeling process

To prepare the dataset, two sources were used: manually collected data, labeled using the Labelbox tool, and also automatically generated data using the Matplotlib library. The process of data generation and transformation is shown in the Fig. 3. As a result, the final dataset contains 10000 images with 10000 $\times$ 7 mask images, where 2000 images and masks was used for validation, 2000 images and masks for testing and 6000 for training.

7 Chart Scale Recognition

It is very important step to automatically convert the coordinates of the image to the coordinates of the target graph of the function. To do this, it is necessary to recognize ticks on the axis lines, then select the text that refers to the signature of these axes, then filter these values for more accurate recognition, then solve a simple linear equation to further transform the coordinates of the image into the coordinates of the objective function for each of the axes. The resulting process is shown in the Fig. 4.

Tick detection is performed as follows. First, on the area of the image corresponding to the axis line in the mask, the longest line is searched for using the hough lines transform

method from the OpenCV library. This line counts as the axis line. Then, all ticks are localized on this line using the sliding window method. At the next stage, the text is searched for, the position of which corresponds to the axis data mask. The found text is filtered according to two rules. The first is to exclude all texts that do not contain numbers. The second allows to include dot symbol to numbers like "025", and then the number is converted to a real type.

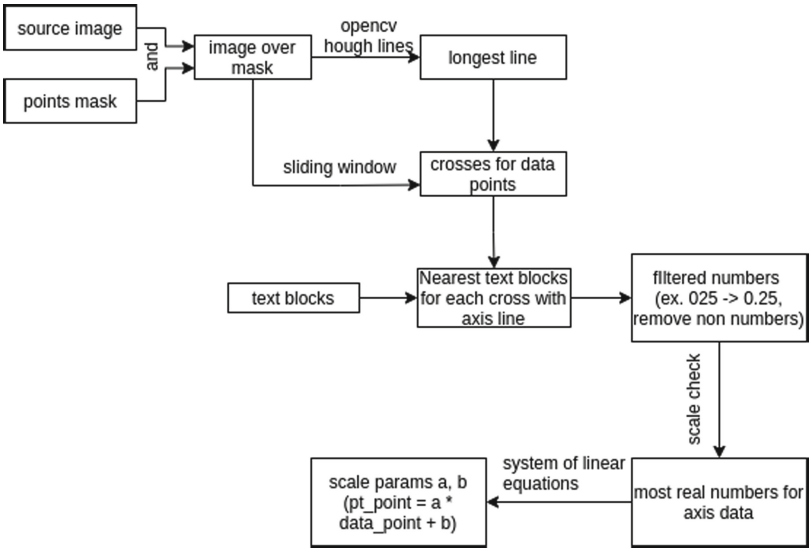

Fig. 4. Scale recognizing data flow

In the next step, the nearest filtered text is selected for each tick. Then, for each pair of ticks and their corresponding numerical values, an extrapolated value is determined for all other ticks. If the number of matches is maximum for the selected pair, then such a pair is used to calculate the coefficients of the linear equation for transferring coordinates from the image coordinate system to the coordinate system of the target function of the chart.

As a result of solving a system of linear equations, coefficients a and b are obtained, which are used to transform the coordinates of points from the image coordinate system to the coordinate system of the target function of the chart by the formulas

$$x_i^{'} = (x_i - b_x)/a_x$$

$$y_i^{'} = \left(y_i - b_y\right)/a_y$$

where a and b are coefficients of the linear functions for horizontal and vertical axes. Example of recognized ticks is presented on the Fig. 5.

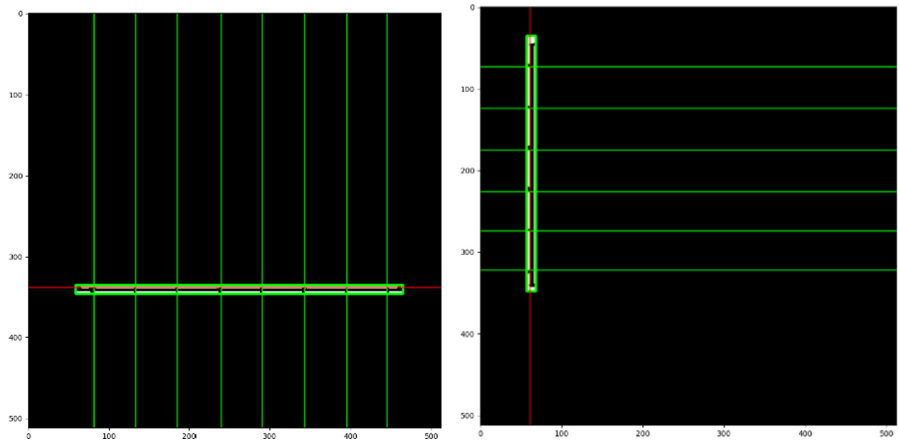

Fig. 5. Example of the axis tick recognition for horizontal axis (left) and vertical axis (right)

8 Data Extraction

The process of extracting data in the image coordinate system is performed by classical algorithms for image analysis without the use of machine learning. The overall process is shown in the Fig. 6.

In the first step, the image is color quantized by k-means clustering to reduce the number of colors. To determine the target value for the number of colors, the following transformations are performed.

First, the image is converted to a HSV palette. Then a HSV (hue, saturation, value) histogram is generated, in which the abscissa axis is the Hue channel (in the interval from 0 to 255), and the ordinate axis is the number of pixels that have a given color on the Hue channel. Then this data is gone through a double filter with a low-pass filter according to the formula

$$O_n = O_{n-1} + \alpha(I_n - O_{n-1})$$

where O_n is a predicted value, O_{n-1} – previous calculated value, I_n – real value, α – regularization term (the best value is selected experimentally).

After double filtering, the number of peaks of the histogram data corresponds to the number of human-readable colors, which is used for color quantization.

An example of such recognition is shown in Fig. 7. For peak detection, we use find_peak method from scipy.signal package.

The task of color quantization is important to the following selection of color lines of the chart.

In the next step, the color masks of the image corresponding to the found number of colors are extracted from the source image and the contours are determined in each part using the canny edge detector. For a found contours, then its area and the area of its bounding rectangle are calculated. If the ratio of these areas is small enough, then we can assume that the contour limits a line, which can be considered a chart of the function.

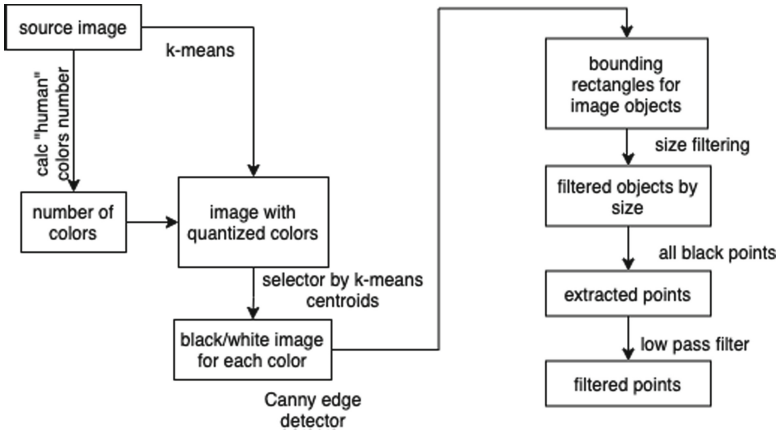

Fig. 6. Data extraction in image coordinates

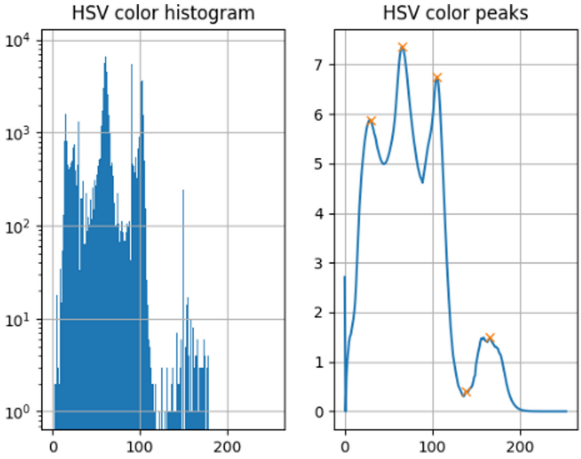

Fig. 7. HSV histogram color peaks

At the final step, the points are extracted from the found contour and the data array is generated in the coordinate system of the image size for each of the color channels. An example of the result of this process is shown in the Fig. 8.

9 Data Complexing

At the final step, the data is converted from the image coordinate system into the coordinate system of the objective function for each graph. A data table is generated with the corresponding signatures of the signs, the data header. The data obtained can be used in the future in the form of tables or text descriptions. We implement the plotting of a function chart for each line. An example is shown in the Fig. 8.

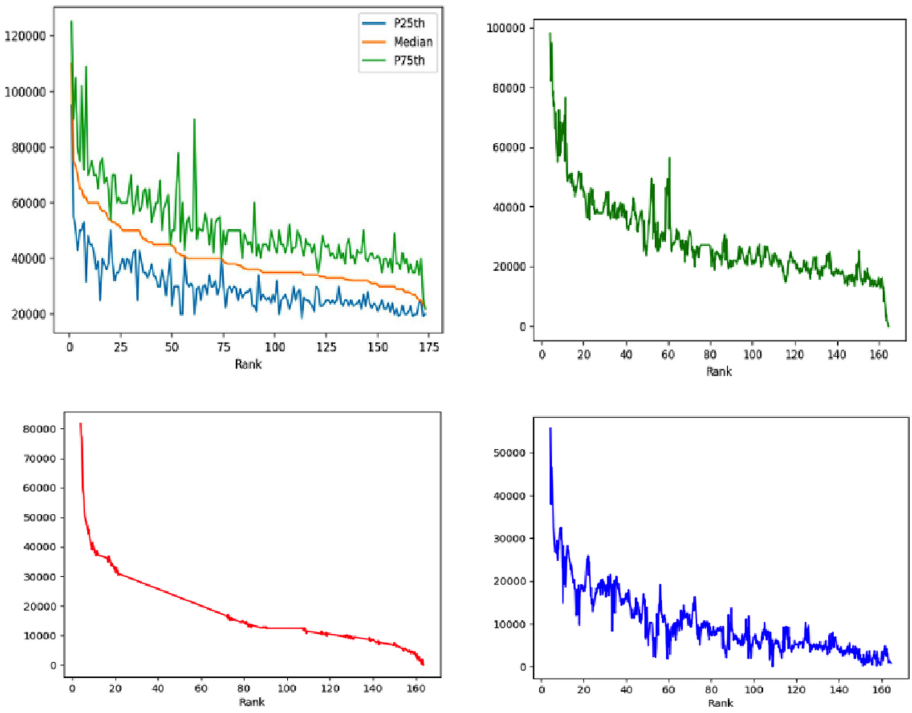

Fig. 8. Original image with chart and extracted charts

10 Conclusion

In some cases, it is not very precise, but it can be improved in the next research. The hybrid architecture of the designed software allows to improve and replace independent modules without changes in other modules:

- Data extraction in the coordinates of the original image can be performed by other methods of filtering.
- Text recognition can use deep learning models taking into account the context of the image.
- Generating data for a dataset can make the data more diverse.
- The mask for detecting image components can be more accurate.
- Applications for recognized data can be broader.

The result of using a deep learning network for localizing parts of charts is 72%, this is enough for recognition since post-processing algorithms significantly improve the final recognition accuracy. The proposed approach and the framework can extract data with good quality and it can be improved with the next steps of work.

Acknowledgments. This study was supported Ministry of Education and Science of Russia in framework of project № 075-00233-20-05 from 03.11.2020 «Research of intelligent predictive

multimodal analysis of big data, and the extraction of knowledge from different sources» and RFBR grant 18-47-732004 p_мк.

References

1. Prasad, V.S.N., Siddiquie, B., Golbeck, J., Davis, L.S.: Classifying computer generated charts. In: Content-Based Multimedia Indexing Workshop, pp. 85–92 (2007)
2. Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M., Heer, J.: Revision: automated classification, analysis and redesign of chart images. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, pp. 393–402 (2011)
3. Zhou, Y., Tan, C.L.: Hough-based model for recognizing bar charts in document images. In: SPIE, pp. 333–341 (2000)
4. Huang, W., Tan, C.L.: A system for understanding imaged infographics and its applications. In: ACM Symposium on Document Engineering, pp. 9–18 (2007)
5. Cliche, M., Rosenberg, D. Madeka, D., Yee, C.: Scatter-act: automated extraction of data from scatter plots. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 135–150 (2017)
6. Poco, J., Heer, J.: Reverse-engineering visualizations: recovering visual encodings from chart images. In: Computer Graphics Forum, pp. 353–363 (2017)
7. Siegel, N., Horvitz, Z., Levin, R., Divvala, S., Farhadi, A.: FigureSeer: parsing result-figures in research papers. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 664–680. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_41
8. Gao, J., Zhou, Y., Barner, K.E.:. View: visual information extraction widget for improving chart images accessibility. In: Proceedings of the 19th IEEE International Conference on Image Processing (ICIP 2012), 2865–2868 (2012)
9. Huang, W., Tan, C.L., Leow, W.K.: Model-based chart image recognition. In: Lladós, J., Kwon, Y.-B. (eds.) GREC 2003. LNCS, vol. 3088, pp. 87–99. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25977-0_8
10. Huang, W., Liu, R., Tan, C.L.: Extraction of vectorized graphical information from scientific chart images. In: Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR 2007), pp. 521–525 (2007)
11. Shao, M., Futrelle, R.P.: Recognition and classification of figures in PDF documents. In: Liu, W., Lladós, J. (eds.) GREC 2005. LNCS, vol. 3926, pp. 231–242. Springer, Heidelberg (2006). https://doi.org/10.1007/11767978_21
12. Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M., Heer, J.: ReVision: automated classification, analysis and redesign of chart images (2011). http://vis.stanford.edu/papers/revision
13. Rohatgi, A.: WebPlotDigitizer, Version 3.8 (2015). http://arohatgi.info/WebPlotDigitizer. Accessed 22 Sept 2015
14. Méndez, G.G., Nacenta, M.A., Vandenheste, S.: iVoLVER: interactive visual language for visualization extraction and reconstruction. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2016), pp. 4073–4085 (2016)
15. Tummers, B.: DataTheif III (2015). http://www.datathief.org/. Accessed 22 Sept 2015
16. Gross, A., Schirm, S., Scholz, M.: Ycasd–a tool for capturing and scaling data from graphical representations. BMC Bioinform. **15**(1), 219 (2014)
17. Liu, X., Klabjan, D., Bless, P.N.: Data extraction from charts via single deep neural network. https://arxiv.org/abs/1906.11906

18. Girshick, R.: Fast R-CNN. https://arxiv.org/pdf/1504.08083.pdf
19. Zhao, Z.-Q., Zheng, P., Xu, S., Wu, X.: Object detection with deep learning: a review. https://arxiv.org/pdf/1807.05511.pdf
20. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. https://arxiv.org/pdf/1502.03044.pdf
21. Wang, W., et al.: Learning unsupervised video object segmentation through visual attention. http://openaccess.thecvf.com/content_CVPR_2019/papers/Wang_Learning_Unsupervised_Video_Object_Segmentation_Through_Visual_Attention_CVPR_2019_paper.pdf
22. Sun, J., Darbehani, F., Zaidi, M., Wang, B.: SAUNet: shape attentive U-net for interpretable medical image segmentation. https://arxiv.org/pdf/2001.07645v3.pdf
23. Sviatov, K., Miheev, A., Kanin, D., Sukhov, S., Tronin, V.: Scenes segmentation in self-driving car navigation system using neural network models with attention. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11623, pp. 278–289. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24308-1_23
24. Papers with code. https://paperswithcode.com/
25. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf
26. Graph-and-Chart-Recognition. https://github.com/Grigorii-24/Graph-and-Chart-Recognition
27. Oktay, O., et al.: Attention U-net: learning where to look for the pancreas. https://arxiv.org/abs/1804.03999
28. Behera, R.K., Shukla, S., Rath, S.K., Misra, S.: Software reliability assessment using machine learning technique. In: Gervasi, O., et al. (eds.) ICCSA 2018. LNCS, vol. 10964, pp. 403–411. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95174-4_32
29. Abayomi-Alli, A., et al.: Facial image quality assessment using an ensemble of pre-trained deep learning models (EFQnet). In: 2020 20th International Conference on Computational Science and Its Applications (ICCSA). IEEE (2020)