Scientific Visualization, 2020, volume 12, number 5, pages 1 - 12, DOI: 10.26583/sv.12.5.01

Solutions to recognize the table structure by an image in the absence of a priori information

N.O. Besshaposhnikov[1,A], A.G. Leonov [2,A,B,C], M.A. Matyushin [3,A]
[A] Scientific Research Institute for System Analysis of the Russian Academy of Sciences
[B] M.V.Lomonosov Moscow State University
[C] Moscow Pedagogical State University
[1] ORCID: 0000-0002-7616-3143, nbesshaposhnikov@vip.niisi.ru
[2] ORCID: 0000-0001-9622-1526, dr.l@vip.niisi.ru
[3] ORCID: 0000-0003-1775-6894, itsaprank@yandex.ru

Abstract

In this paper, we consider the problem of recognizing a table structure through the analysis of the provided picture. The problem statement is the following: we have a photo with an unknown number of particular objects captured, and we know that they are arranged in a flat table structure. It is assumed that the provided picture complies reasonable restrictions concerning perspective distortion and rotation magnitudes. The goal is to recognize the underlying table structure, i.e., to arrange the recognized objects into some table structure that appropriately fits the picture. From now on, we call this procedure the tabulating of the objects. This paper then considers the task of tabulating objects under the conditions of the absence of any antecedent information concerning the table structure, except for the actual picture.

Keywords: tabulating, machine learning, deep learning, neural networks, image recognition, table structure, objects ordering.

1. Introduction

It is known [1], how to tabulate objects with deep neural networks when the size of the table is known a priori. That is, we can quickly restore the object's integer cell coordinates in case we know the total amount of cells in horizontal and vertical directions. Although in this case such information is not provided, the problem entangles as we need to determine the size of the table alongside with the cell coordinates. Therefore we need a model that determines the size of a given table pictured in the image. Let us call such a model the sizer.

In this article, we describe our attempts to build the sizer via deploying the known deep neural networks from [1]. We also propose another approach to determine the size, report some improvements for training methodology, and describe deployed sizer models chronologically along with their results on the test dataset.

2. Formulation of the problem

Lately, the problem of recognition of tables and tabular structures gained much attraction. In general, the model receives the image of a table. Usually, the table in the image is printed or drawn on some plain surface such as paper or a chalk-board. The model then produces some sort of description of table structure, including objects, located in the cells of the table, and the indices of those cells. The image is customary a scan or even an HTML page. Nonetheless, the general setting does not impose such restrictions and also a perspectively distorted image can considered.

Well known approaches include splitting an algorithm into two stages. Namely, table detecting and table recognition. At table detecting, we aim to determine the edges of table elements, and at table recognition, we use those edges to reconstruct table cells indices. In our investigation, we primarily consider the second stage of an algorithm. We also split it into two parts, namely, recognition of relative table coordinates of elements and determining the cell size of the table. As one can easily see, the solutions of these two smaller problems combined yield a full solution of a

table recognition problem, since the relative coordinate multiplied by a cell size gives an absolute cell index.
The problem of deriving relative table coordinates is fully described in [1]. Henceforth in this very article, we aim to provide a solution to the second problem of table recognition, namely, the determination of the cell size of a table. To deliver such a solution we use modern machine learning technics.

3. Related work

In the past few years, the problem has been actively discussed. In particular, deep neural networks and machine learning were shown to be very effective in the context of tabulating (see [2], [3], [4], [5]). Mostly these works consider the narrowed formulation of the problem, considering the image as a scan of a document with a table. The solution is suggested to be a deep neural network, which typically consists of several stacked deep neural networks ([2]), which reduces the problem to the object detection task. Various CNN-based approaches [3] showed remarkable results on standard test datasets. Some approaches demonstrate innovative ideas such as moving documents from black-and-white domain to the traditional for CNNs three-channel domain in order to use known object detection neural networks [4], deploying transfer learning [6] to solve the table detection problem. The latter is also considered to be solved using unsupervised learning [5]. Moreover, in [7] there is a series of classical models to use as a baseline for future investigations as well as a wide dataset of images of tables, which are mostly real documents typed in Word and LaTex.
Nevertheless, in our experiments, we saw very bad convergence of described deep architectures when applied beyond the restricted formulation of the problem. Thus, we decided to split the problem into smaller parts and solve them separately by taking advantage of machine learning and deep neural networks.

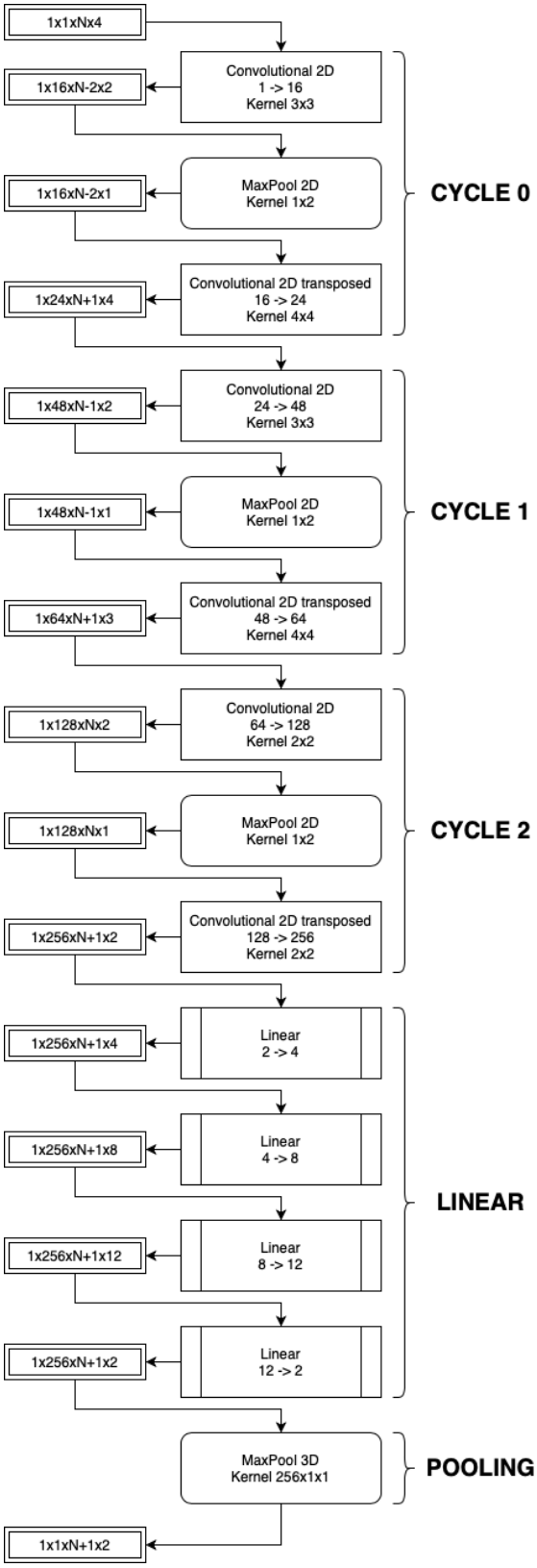
4. Solution methods

4.1. Sizing as a regression problem

There is an effective tabulating neural network model in [1] that obtains relative cell coordinates. However, those results are only useful if we know the cell size of the table. In order to extend the applicability of this solution, we generalized the model to include the unknown parameters, which are the width and height of the table. These two parameters are considered as continuous variables, which allows us to formulate the problem in regression terms.
The natural approach is to build a single model that would solve the whole problem on its own. The result of applying this method is a deep neural network that predicts the size of the table and the relative table coordinates in one run. The architecture of this composite model is represented in the picture 1. From here onwards, we call the layers of a neural network as follows:

MaxPool	Sub-sampling layer with max function [9]
Convolutional transposed	Transposed convolutional layer [8]
Convolutional	Convolutional layer [8]
Linear	Fully connected layer[10]
Softmax	Softmax layer [11]

On the left in the picture 1, one can see output tensors ranks. This neural network receives the tensor of rank $1 \times 1 \times N \times 4$ and returns the tensor of rank $1 \times 1 \times N+1 \times 2$, where N is the number of objects (i.e. cells). The first N rows of output tensor are considered the relative table coordinates, and the last row is the width and the height of the table.



Pic. 1: The architecture of the composite model

It is easy to see that the composite model is a modification of the TabCNNf neural network, which is described in [1]. The difference is the additional CYCLE 0 and a greater number of channels in the composite model. Since this model predicts all necessary data in one run, it is possible to use it with low power devices.

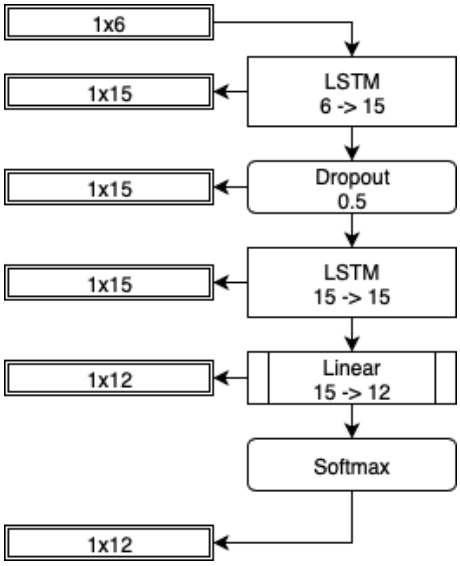
4.2. Reformulation

The above model solves the problem in the regression setting. Taking into the account the natural restrictions on the table size, it is appropriate to reformulate the problem. Namely, let us solve the sizing problem as the classification task. To accomplish this, let us suppose that each size of the table is less or equal than 12 cells, for example. In this case, we have to categorize images into one of 12*12=124 classes. Let us point out that our sizer models receive not the images themselves, but the coordinates of the objects detected in the image which are considered the cells of the table.

Nonetheless, the classification with such a large number of classes is complicated; hence let us split it into two parallel classification problems with 12 classes, namely the prediction of the width and the prediction of the height. As a result of this, the sizer in the classification setting is a model that solves the above two classification tasks.

4.3. Recurrent sizer

In our experiments, we built a recurrent model with two LSTM cells [12], and the regularization dropout layer [13] with 0.5 probability between them followed by a fullyconnected layer. The accurate architecture is represented in picture 2.

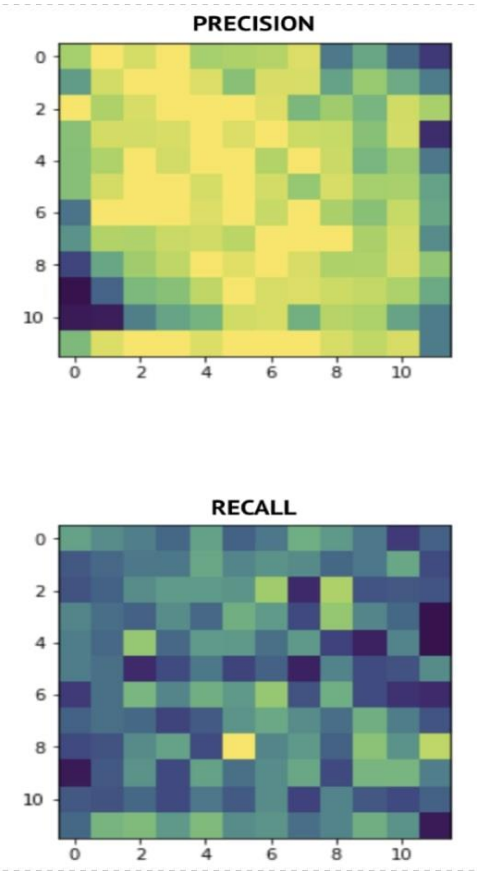


Pic. 2: The architecture of the recurrent model

Let us point out that the recurrent model allows us to neglect the unknown amount of the detected objects which we denote with N in the previous section. In the recurrent model, the input tensor of the model is the tensor of rank 1×6. The difference between two different pictures is only the number of runs of the recurrent model, which equals N. The additional two features (6 instead of 4 in the previous section) are the non-overlapping areas of projection on the bottom and on the left side of the picture.

The hidden state of the model is nullified after each image processed and changes through the predicting on N rows of the tensor of rank $N \times 6$.

For the recurrent sizer, we obtained f1 score = 0.84 on the test dataset. Precision and recall of the model is represented in picture 3.



Pic. 3: Precision and recall of the recurrent model for width and height classes

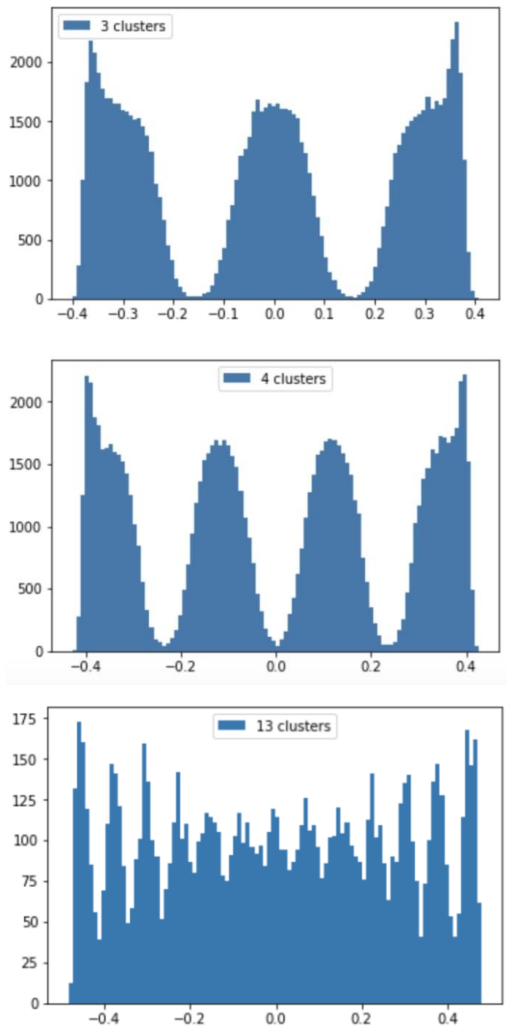
Let us note that we trained two independent models that receive equal inputs. Despite of the outstanding precision of the recurrent sizer, the prediction quality heavily depends on the number of detected objects, which means the decreasing quality when processing smaller tables.

4.4. "Histogram trick"

In order to fix the rank of input tensor, we can also convert the input data with so-called "histogram trick" [14], [15]. Namely, we create a histogram of distribution for each of 6 features according to the detected objects. The obtained histograms are then used as inputs for the model. Histogram trick fixes the rank of input tensor since our features are normalized into [-0.5, 0.5] interval for objects coordinates and into [0, 1] interval for non-overlaping projection areas; hence the intervals and the steps of our histograms are fixed which means that the size of the histogram itself is fixed too. From here onwards, the number of bins of all histograms is assumed to equal 300. Experiments that included the increase of this parameter did not show a significant quality gain.

4.5. Unsupervised sizer

For now, we have input tensor of fixed rank 300 × 6, which allows us to use simple classical approaches. In particular, we built several unsupervised learning methods. The first method we used consists of training of 24 normal distributions mixture models. These models traditionally are trained using the EM algorithm [16]. In our setting, we train 12 models for both the width and height of the table. To compute the size of the table, we can compare our feature histograms with distributions learned by mixture models and pick the model, which delivers the maximum likelihood. The distributions learned by mixture models are represented in the picture 4. X-line here is the relative coordinate of the detected object, and Y-line is the number of objects which have such a center.

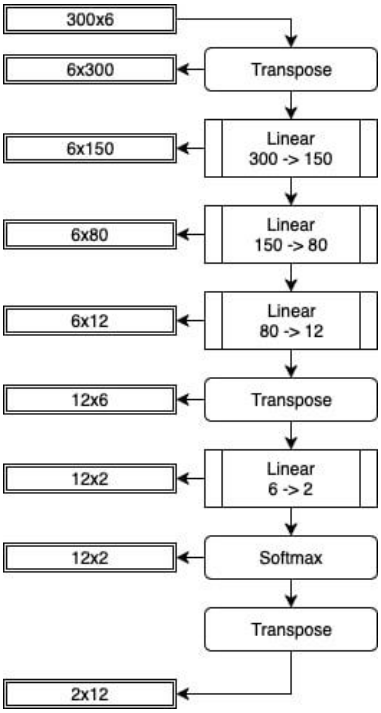


Pic. 4: The distributions of the coordinate of the object's center with respect to the table size

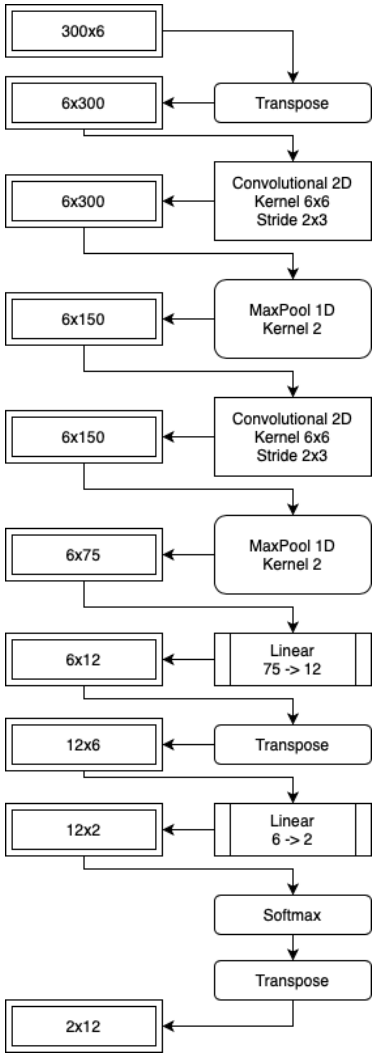
Despite of apparent simplicity of these distributions, the larger value of the number of clusters generates the distorted distribution, which conditions more significant error of this model.
The second unsupervised approach we tested was the silhouette analysis [17], which is the method to choose the number of clusters in the KMeans algorithm, which maximizes the silhouette metric. Both of the clustering approaches showed comparatively low quality.

4.6. Fully connected and convolutional sizers

To provide the solution of the stated classification problem, we also used simple neural networks. The first model is a fully connected neural network, and its architecture is represented in picture 5.
The second model is the convolutional neural network, and its architecture is represented in picture 6.



Pic. 5: The architecture of the fully connected sizer

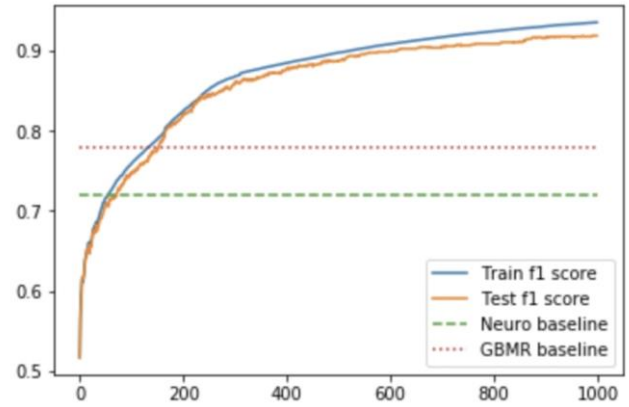


Pic. 6: The architecture of the convolutional sizer

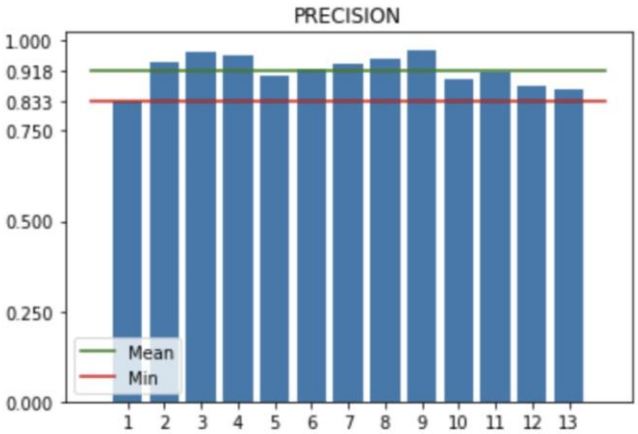
The output of the model is the 12 classes' joint distribution of both width and height.

4.7. Gradient boosting machine, woody sizer

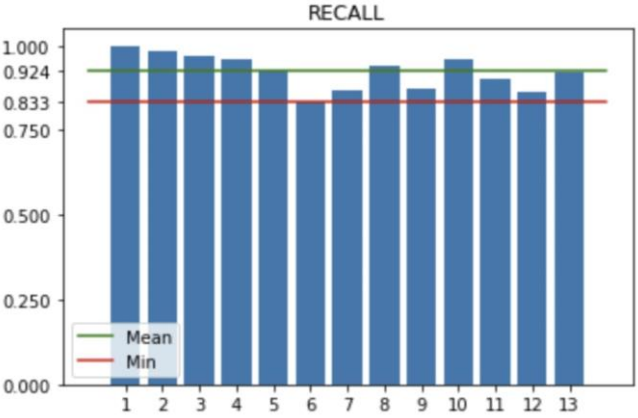
The last approach we have deployed was the gradient boosting machine [18]. We trained GBM with CART trees of depth 2, 3, and 5. The quality of the model was monotonously increasing with the increase of depth. The number of trees was equal to 1000. The f1 score of the model concerning the number of trees is represented in picture 7. Precision and recall metrics are represented in pictures 8 and 9.



Pic. 7: Training the GBM with trees of depth 5



Pic. 8: Precision of the woody sizer



Pic. 9: Recall of the woody sizer

5. Experiments

The input and output data have to be normalized. Let (x_1, y_1), (x_2, y_2) be the coordinates of left top, and right bottom corners of the object in the picture, (x_0, y_0) be the coordinates of left-top of the rectangle area containing all of the detected objects, w, h be the width and the height of this area. Let us denote normalized coordinates with $(\overline{x}_1, \overline{y}_1)$, $(\overline{x}_2, \overline{y}_2)$. For them, we have the formula (1):

$$\begin{aligned} \overline{x}_1 &= \frac{x_1 - x_0}{w} - 0.5 \\ \overline{x}_2 &= \frac{x_2 - x_0}{w} - 0.5 \\ \overline{y}_1 &= \frac{y_1 - y_0}{h} - 0.5 \\ \overline{y}_2 &= \frac{y_2 - y_0}{h} - 0.5 \end{aligned} \tag{1}$$

Considering the object with cell coordinates (i, j), let us denote its normalized cell coordinates as $(\overline{i}, \overline{j})$. For them, we have the formula (2):

$$\begin{aligned} \overline{i} &= \frac{i}{w_c} - 0.5 \\ \overline{j} &= \frac{j}{h_c} - 0.5 \end{aligned} \tag{2}$$

Here w_c, h_c are cell width and cell height of the table, which we aim to predict. Since w_c, h_c are the output of the model, they should be normalized

too. Let us denote normalized cell width and cell height as $\overline{w_c}$, $\overline{h_c}$. Suppose we have only tables with the following restrictions:

$$w_c \text{ x } h_c, \text{ where } 1 \le w \le w_c \le W, 1 \le h \le h_c \le H.$$

Then we have the formula (3):

$$\begin{aligned} \underline{w_c} &= \frac{1}{w_c} \\ \underline{h_c} &= \frac{1}{h_c} \end{aligned} \tag{3}$$

That is according to the formulae above, we normalize cell width and cell height into [-0.5, 0.5] interval.

Besides the bounding box coordinates, we also have non-overlapping projection areas. Let us describe how to calculate these features. Suppose that we have coordinates of points of all objects in the image: $\{(x_1^{(n)}, y_1^{(n)}), (x_2^{(n)}, y_2^{(n)}), \ldots\}$, where $1 \le n \le N$ is the index of the object. Let us denote

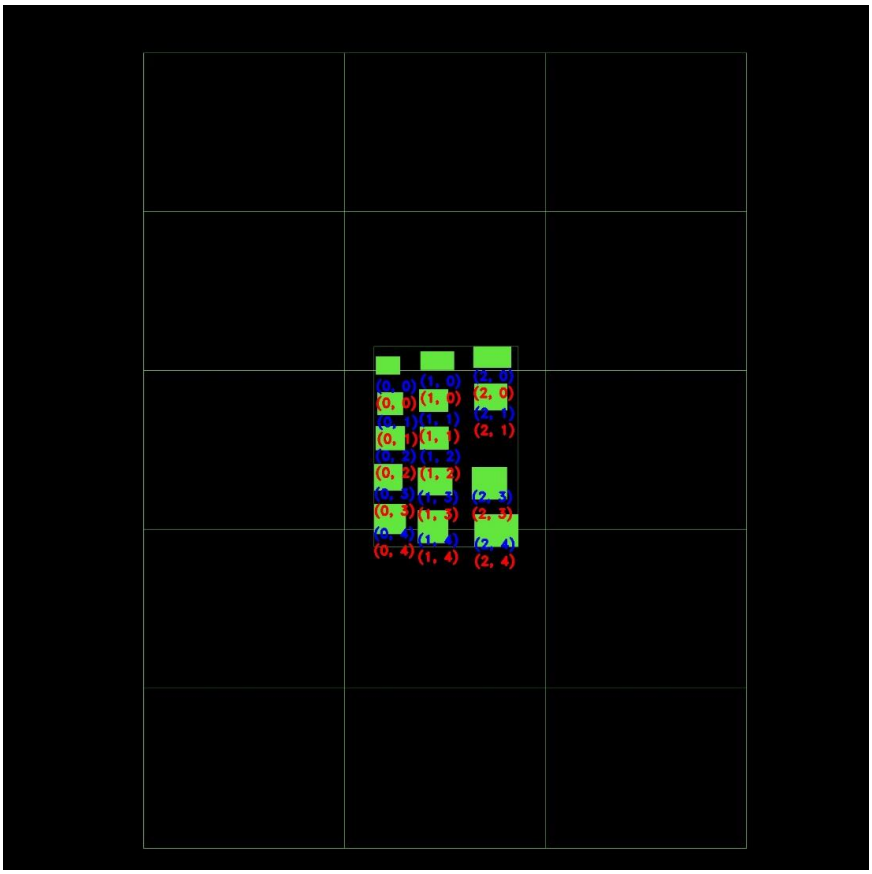
$$\begin{aligned} x^n &= sup(\{x_1^{(n)}, x_2^{(n)}, \ldots\}), \\ x_n &= inf(\{x_1^{(n)}, x_2^{(n)}, \ldots\}), \\ y^n &= sup(\{y_1^{(n)}, y_2^{(n)}, \ldots\}), \\ y_n &= inf(\{y_1^{(n)}, y_2^{(n)}, \ldots\}), \end{aligned}$$

and let μ be the Lebesgue measure on $\mathbb{R}$. Then a non-overlapping projection area of the object with index n is calculated according to formulae (4), (5):

$$P_x(n) = \mu\left([x_n, x^n] \setminus \bigcup_{i:y_i<y_n} [x_i, x^i]\right) \tag{3}$$

$$P_y(n) = \mu\left([y_n, y^n] \setminus \bigcup_{i:x_i<x_n} [y_i, y^i]\right) \tag{4}$$

Taking (1), (2), (3), (4), (5) into account the input has N rows of the form ($\overline{x}_1$, $\overline{y}_1$, $\overline{x}_2$, $\overline{y}_2$,P_x,P_y) and the output has N rows of the form ($\overline{i}$, $\overline{j}$) and the last row of the form ($\overline{w_c}$, $\overline{h_c}$) for the composite model; the output has one row of the form ($\overline{w_c}$, $\overline{h_c}$) for all other models.
During all experiments, the train and the test datasets were fully synthetically generated.
That is, we have a module which generates pictures of tables with rotations and perspective distortions and creates input tensors of rank N × 6, where N is a number of cells in the table, 6 is a number of features, which are normalized coordinates of the cells and the non-overlapping projection areas of the cells. The example of such an image is represented in the picture 10. Blue integers are ground truth cell coordinates, whereas red integers are model predicted cell coordinates.



Pic. 10: The example of the training image

The quality of the described models in terms of F1 score is listed in the table below:

Method	F1 score
Clustering model	0.2
Composite model	0.6
Fully connected and convolutional models	0.71
Recurrent model	0.84
GBM	0.93

6. Conclusion

In this article, we have proposed several approaches to solving the problem of sizing a table, which is applicable in recognition of the table structure in the absence of a priori information.
The acquired results show that classical GBM along with the histogram trick, yields the best results. Nonetheless, potentially GBM lacks generalization abilities while being used in real conditions.
Since the highest obtained score is far less than 1.0, we will continue the research in order to build the solution. In particular, the next experiments include GBM initialized with recurrent sizer, or training recurrent model on features extracted with GBM.
This work was supported by a grant from the Russian Foundation for Basic Research 18-07-00901.

References

1. Besshaposhnikov N.O., Leonov A.G., Matyushin M.A. Voprosy uporyadochivaniya objektov na izobrazhenii s ispol-zovaniem neyrosetevykh i ehvristicheskikh algoritmov [On the arranging detected objects using neural networks and heuristic algorithms]. // Proceedings in cybernetics - 4(32), 2018.
2. Sebastian Schreiber Stefan Agne Ivo Wolf Andreas Dengel Sheraz Ahmed. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017.
3. A Table Detection Method for PDF Documents Based on Convolutional Neural Networks / Leipeng Hao, Liangcai Gao, Xiaohan Yi [et al.]. 12th IAPR Workshop on Document Analysis Systems (DAS), 2016.
4. Table Detection Using Deep Learning / Azka Gilani, Shah Rukh Qasim, Imran Malik [et al.]. 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017.
5. Fan M. Kim D. S. Table Region Detection on Large-scale PDF Files without Labeled Data. CoRR, abs/1506.08891, 2015.
6. West Jeremy; Ventura Dan; Warnick Sean. A Theoretical Foundation for Inductive Transfer. Spring Research, 2005.
7. Minghao Li Lei Cui Shaohan Huang Furu Wei Ming Zhou Zhoujun Li. TableBank: Table Benchmark for Image-based Table Detection and Recognition. arXiv preprint arXiv:1903.01949, 2019.
8. Alex Krizhevsky Ilya Sutskever Geoffrey E. Hinton. ImaeNet Classification with Deep Convolutional Neural Networks. // Advances in Neural Information Processing Systems 25, 2012.
9. Yamaguchi Kouichi; Sakamoto Kenji; Akabane Toshio; Fujimoto Yoshiji. A Neural Network for Speaker-Independent Isolated Word Recognition. // First International Conference on Spoken Language Processing (ICSLP 90), 1990.
10. Rosenblatt F. THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGEAND ORGANIZATION IN THE BRAIN. // Psychological review - No 6(65), 1958.
11. F.R. Christopher M. Bishop. Pattern Recognition and Machine Learning. // Springer, 2006.
12. Sepp Hochreiter Jurgen Schmidhuber. Long Short-Term Memory. // Neural computation - No 9(8), 1997.
13. Nitish Srivastava Geoffrey Hinton Alex Krizhevsky Ilya Sutskever Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. // Journal of Machine Learning Research -15, 2014.
14. Chapelle Olivier, Haffner Patrick, Vapnik Vladimir. SVMs for Histogram-Based Image Classification. // IEEE TRANSACTIONS ON NEURAL NETWORKS - No 5(10), 1999.
15. Norbert Obsuszt AnswerMiner. Histogram 202: Tips and Tricks for Better Data Science. https://www.kdnuggets.com/2018/02/histogram-tips-tricks.html.

16. UniversityStanford.GaussianMixturesandtheEMalgorithm. http://statweb.stanford.edu/tibs/sta306bfiles/mixtures-em.pdf.
17. J.Rousseeuw Peter. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. // Journal of Computational and Applied Mathematics, 1987.
18. Friedman Jerome H. Greedy Function Approximation: A Gradient Boosting Machine. // The Annals of Statistics - No 5(29), 2001.