Multi-Type-TD-TSR - Extracting Tables from Document Images using a Multi-stage Pipeline for Table Detection and Table Structure Recognition: from OCR to Structured Table Representations

Pascal Fischer, Alen Smajic, Alexander Mehler, Giuseppe Abrami

{s4191414@stud,s0689492@stud,amehler@em,abrami@em}.uni-frankfurt.de

ABSTRACT

As global trends are shifting towards data-driven industries, the demand for automated algorithms that can convert digital images of scanned documents into machine readable information is rapidly growing. Besides the opportunity of data digitization for the application of data analytic tools, there is also a massive improvement towards automation of processes, which previously would require manual inspection of the documents. Although the introduction of optical character recognition (OCR) technologies mostly solved the task of converting human-readable characters from images into machine-readable characters, the task of extracting table semantics has been less focused on over the years. The recognition of tables consists of two main tasks, namely table detection and table structure recognition. Most prior work on this problem focuses on either task without offering an end-to-end solution or paying attention to real application conditions like rotated images or noise artefacts inside the document image. Recent work shows a clear trend towards deep learning approaches coupled with the use of transfer learning for the task of table structure recognition due to the lack of sufficiently large datasets. In this paper we present a multistage pipeline named Multi-Type-TD-TSR, which offers an end-to-end solution for the problem of table recognition. It utilizes state-of-the-art deep learning models for table detection and differentiates between 3 different types of tables based on the tables' borders. For the table structure recognition we use a deterministic non-data driven algorithm, which works on all table types. We additionally present two algorithms. One for unbordered tables and one for bordered tables, which are the base of the used table structure recognition algorithm. We evaluate Multi-Type-TD-TSR on the ICDAR 2019 table structure recognition dataset [3] and achieve a new state-of-the-art. The full source code is available on https://github.com/Psarpei/Multi-Type-TD-TSR.

1 INTRODUCTION

OCR based on digitized documents in general and OCR post-correction in particular remains a desideratum, especially in the context of historical documents when they have already been subjected to OCR. In the case of such texts, incorrect or

Figure 1: Types of tables based on how they utilize borders: a) tables without borders, b) tables with partial borders, c) tables with borders.

incomplete recognition results often occur due to the application of mostly purely letter-oriented methods. Considerable methodological progress has been made in the recent past, with the focus of further developments being in the area of neural networks. However, special attention must be paid to the recognition of tables, where performance tends to be poor. In fact, the scores in this area are so poor that downstream NLP approaches are still practically incapable of automatically evaluating the information contained in tables. Better recognition of table structures is precisely the task to which this work relates.

Tables are used to structure information into rows and columns to compactly visualize multidimensional relationships between information units. In order to convert an image of a table, i.e. a scanned document that contains a table into machine readable characters, it is important to structure the information in such a way that the original relationships between the information units and their semantics is preserved. Developing algorithms that can handle such conversion tasks is a major challenge, since the appearance and layout of tables can vary widely and depends very much on the style of the table author. The already mentioned row and column structure of tables goes hand in hand with different sizes and layouts of table elements, changing background colors and fonts per cell, row or column and changing borders of the table as a whole or its individual entries. All these
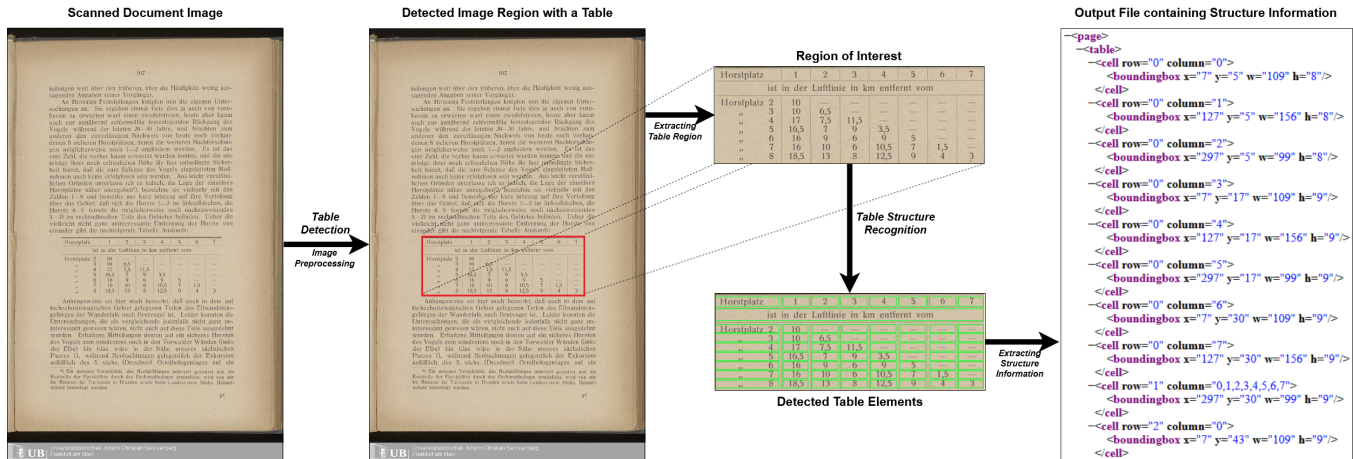
Figure 2: Schematic depiction of the staged process of *Table Detection* (TD) and *Table Structure Recognition* (TSR) starting from table images, i.e. digitized document pages containing tables.

properties must be taken into account in order to achieve sufficient applicability of OCR, especially in the area of historical documents. Otherwise information represented in tables is only insufficiently available or not available at all for downstream tasks of *Natural Language Processing* (NLP) and related approaches.

Consequently, we are faced with a *computer vision task* for mapping table images to structured, semantically interpreted table representations. For this task, table borders are of particular importance because they serve as a direct visualization of the table structure and act as a frame for the elementary cell elements that are ultimately input to NLP. In order to address this scenario, we distinguish between three types of tables. Figure 1a) shows a table without any table borders, Figure 1b) a partially bordered table and Figure 1c) a fully bordered one. We refer to tables without any borders as *unbordered tables*, tables that are completely bordered as *bordered tables* and tables that contain some borders as *partially bordered tables*. It should be mentioned that *partially bordered tables* also include unbordered and bordered tables.

The task of converting an image of a table into machine readable information starts with the digital image of a document, which is created using a scanning device. Obviously, this process is crucial for the later conversion, since small rotations of the document during scanning or noise artifacts generated by the scanning device can have a negative impact on recognition performance. The conversion itself involves two steps, namely *Table Detection* (TD) inside a document image and *Table Structure Recognition* (TSR). TD is performed to identify all regions in images that contain tables, while TSR involves identifying their components, i.e. rows, columns, and cells, to finally identify the entire table structure - see

Figure 2 for a schematic depiction of this two-step recognition process. In general, these two tasks involve detecting some sort of bounding boxes that are correctly aligned with the table elements to be identified. However, without proper alignment of the entire table image, it is not possible to generate accurate bounding boxes, which reduces the overall performance of table representation. Thus, the correct alignment of table images is to be considered as a constitutive step of the computer vision task addressed here.

In this paper, we present a multistage pipeline named *Multi-Type-TD-TSR* which solves the task of extracting tables from table images and representing their structure in an end-to-end fashion. The pipeline consists of four main modules that have been developed independently and can therefore be further developed in a modular fashion in future work. Unlike related approaches, our pipeline starts by addressing issues of rotation and noise that are common when scanning documents. For TD we use a fully data-driven approach based on a *Convolutional Neural Network* (CNN) to localize tables inside images and forward them to TSR. For the latter we use a deterministic algorithm in order to address all three table types of Figure 1. Between TD and TSR, we perform a pre-processing step to create font and background color invariance so that the tables contain only black as the font color and white as the background color. In addition, we present two specialized algorithms for bordered tables and unbordered tables, respectively. These two algorithms form the basis for our table type-independent algorithm for recognizing table structures. Our algorithm finally represents recognized table elements by means of a predefined data structure, so that the table semantics can finally be processed further as a structured document – for example in NLP pipelines.
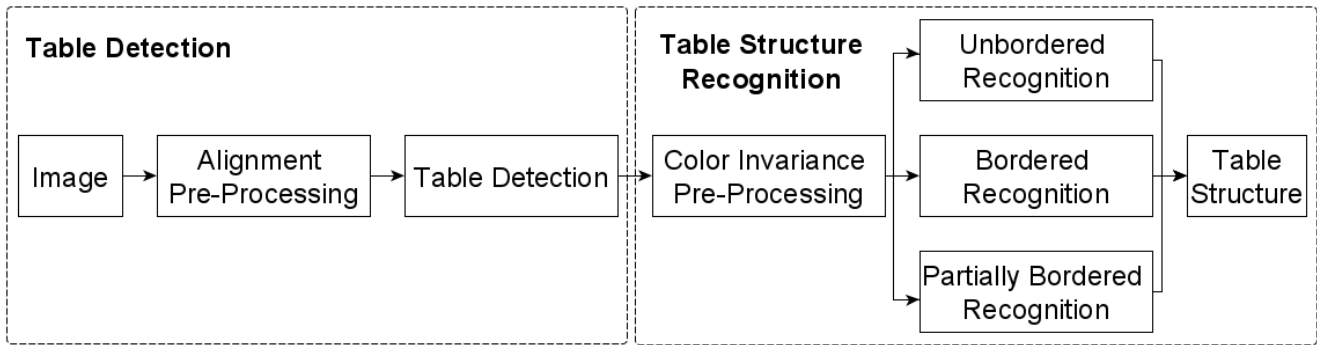
Figure 3: The two-stage process of TD and TSR in Multi-Type-TD-TSR.

For table detection we use the state-of-the-art deep learning approach proposed by Li *et al.* [9], which was evaluated on the TableBank [9] dataset. For the task of table structure recognition we evaluate Multi-type-TD-TSR on the ICDAR 2019 dataset (Track B2) [3].

The paper is structured as follows: In Section 2, we summarize related work. Then the pipeline of Multi-type-TD-TSR is explained in Section 3 and in more detail in Section 4. After that, the evaluation and comparison with state-of-the-art techniques is presented in Section 5. Finally, we draw a conclusion in Section 6 and preview future work in Section 7.

2 RELATED WORK

Several works have been published on the topic of extracting table semantics and there are comprehensive surveys available describing and summarizing the state-of-the-art in the field since 1997 when P. Pyreddy and, W. B. Croft [11] proposed the first approach of detecting tables using heuristics like character alignment and gaps inside table images. To further improve accuracy, Wonkyo Seo *et al.* [16] used an algorithm to detect junctions, that is, intersections of horizontal and vertical lines inside bordered tables. T. Kasar *et al.* [7] also used junction detection, but instead of heuristics he passed the junction information to a SVM [2] to achieve higher detection performance.

With the advent of *Deep Learning* (DL), advanced object recognition algorithms, and the first publicly available datasets, the number of fully data-driven approaches continued to grow. Azka Gilani *et al.* [5] was the first to propose a DL-based approach for table detection by using Faster R-CNN [12]. Currently, there are three primary datasets used for TD and TSR. The first one is provided by the ICDAR 2013 table competition [4], which is a benchmark for TD and TSR. The dataset contains a total of 150 tables: 75 tables in 27 excerpts from the EU and 75 tables in 40 excerpts from the

US Government. Its successor, the ICDAR 2019 competition on Table Detection and Recognition (cTDaR) [3] features two datasets. The first one consists of modern documents with modern tables, while the other consists of archival documents with presence of hand-drawn tables and handwritten text. In general this dataset includes 600 documents for each of the two datasets with annotated bounding boxes for the image regions containing a table. In 2020, Li *et al.* [9] published the TableBank dataset, the latest available dataset. It consists of Word and LaTeX documents and is the first to be generated completely automatically. The TableBank dataset includes 417.234 annotated tables for TD and 145.463 for TSR. Unfortunately, the dataset for TSR contains only information about the number of rows and columns, but no information about the location or size of table elements. Li *et al.* [9] proposed a ResNeXt-152 [17] model trained on TableBank for TD, which represents the current state-of-the-art for type-independent TD.

In 1998, Kieninger and Dengel [8], introduced the first approach to TSR by means of clubbing text into chunks and dividing chunks into cells based on column border. The low number of annotated tables with bounding-boxes for TSR enforces the use of transfer learning and leads to a high risk of overfitting. Schreiber *et al.* [15] addressed TD and TSR in a single approach using a two-fold system based on Faster R-CNN [12] for TD and DL-based semantic segmentation for TSR that utilizes transfer learning. Mohsin *et al.* [13] generalized the model by combining a GAN [6] based architecture for TD with a SegNet [1] based encoder-decoder architecture for TSR. Recently, Prasad *et al.* [10] presented an end-to-end TD and TSR model based on transfer learning. It is the first approach that distinguishes between different types of tables and solves each type with a different algorithm. First, an object detection model is used to identify tables inside a document and to classify them as either bordered or unbordered. In a second step, TSR is addressed with a CNN for non-bordered tables and with a deterministic algorithm

Figure 4: Example of the erosion and dilation operations as performed by Multi-Type-TD-TSR for bordered tables.

based on the vertical and horizontal table borders for bordered tables. For the development of Multi-Type-TD-TSR we utilize the TD approach proposed by Li *et al.* [9], since it offers the state-of-the-art model for type-independent TD, which is crucial for our TSR approach. For the task of TSR we use the architecture proposed by Prasad *et al.* [10], who introduced the erosion and dilation operation for bordered tables, and extend this approach to implement a robust algorithm that can handle all table types.

3 END-TO-END MULTISTAGE PIPELINE

Figure 3 shows our multi-stage, multi-type pipeline. It consists of two main parts, namely *Table Detection* (TD), which processes the full-size image, and *Table Structure Recognition* (TSR), which processes only the recognized sections from TD. In the first step, a pre-processing function is applied to the scanned document image to correct the image alignment before passing the scan to the next step. The aligned image is then fed into into a ResNext152 model of the sort proposed by [9] in order to perform TD. Based on the predicted bounding boxes, the recognized tables are cropped from the image and successively passed to TSR. Here, our algorithm applies a second preprocessing step that converts the foreground (lines and fonts) to black and the background to white to create a color-invariant image. In the next step, 3 branching options are available. The first one uses an algorithm specialized for unbordered tables. The second one utilizes a conventional algorithm based on [10] that is specialized for bordered tables. The third option is a combination of the latter two; it works on partially bordered tables, which includes fully bordered and fully unbordered tables, making the algorithm type-independent. Finally, the recognized table structure is exported per input table using a predefined data structure.

4 METHODS

All three TSR algorithms used by us are based on the following two mathematical operations. The first operation is `dilation` defined as follows:

$$\text{dilation}(x, y) = \max_{(x', y'): K(x', y') \neq 0} I(x + x', y + y') \quad (1)$$

This operation involves filtering an image I with a kernel K, which can be of any shape and size, but in our case is a rectangle. K has a defined anchor point, in our case the center of the kernel. As K slides over the image, the maximum pixel value overlapped by K is determined and the image pixel value at the anchor point position is overwritten by this maximum value. Maximization causes bright areas within an image to be magnified.

The second operation named `erosion` is defined as follows:

$$\text{erosion}(x, y) = \min_{(x', y'): K(x', y') \neq 0} I(x + x', y + y') \quad (2)$$

It works analogously to `dilation`, but determines a local minimum over the area of the kernel. As K slides over I, it determines the minimum pixel value overlapped by K and replaces the pixel value under the anchor point with this minimum value. Conversely to `dilation`, `erosion` causes bright areas of the image to become thinner while the dark areas become larger.

Following the example of Prasad *et al.* [10], we use `erosion` on bordered tables to detect vertical and horizontal borders, which need to be retained, while removing the font and characters from the table cells resulting in a grid-cell image. `Dilation` is applied successively to restore the original table border structure, since `erosion` shortens the borders. Additionally we apply `erosion` on unbordered tables to add the missing borders producing a full grid-cell image.

4.1 Table Alignment Pre-Processing

The first method of our Multi-Type-TD-TSR algorithm includes table alignment pre-processing, which is crucial for TSR. Currently, this pre-processing utilizes the text skew correction algorithm proposed by [14]. To remove all noise artifacts within an image, we apply a median filter of kernel size 5x5 pixels, which showed the best results in our experiments. One by one, the algorithm converts the image to grayscale and flips the pixel values of the foreground (lines and characters) so that they have a white color and the background has a black color. In the next step we compute a single bounding box that includes all pixels that are not of black color and therefore represent the document content. Based on this bounding box we calculate its rotation and apply a deskew function, which rotates the bounding box along with its content to be properly aligned.
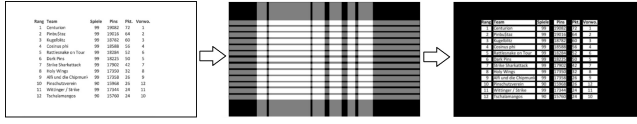
Figure 5: Example of the erosion operation as performed by Multi-Type-TD-TSR for unbordered tables.

Figure 6: Example of the erosion operation as performed by Multi-Type-TD-TSR for partially bordered tables

4.2 Table Detection

In the TD step, we extract the bounding-boxes for each table inside the image by using a *Convolutional Neural Network* (CNN) which does not distinguish between the three table types (see Figure 1). We utilize the approach of Li *et al.* [9] who trained a ResNeXt-152 [17] model on the TableBank dataset [9]. The reason for this selection is that this model reaches the best results by only detecting bounding boxes for each table without classification. The state-of-the-art approach from Prasad *et al.* performs an additional classification of tables by borders. We decided against the approach of Prasad *et al.* [10], since their classification only considers two table types and also includes a slightly different definition of bordered and unbordered tables than ours.

4.3 Bordered TSR

The algorithm for bordered TSR is based on the same named algorithm from Prasad *et al.* [10], which utilizes the erosion and dilation operation for extracting the row-column grid cell image without any text or characters. The first step includes converting the image into a binary representation with pixel values of either zero (black) or one (white) and finally inverting these values to get a table image of white foreground (lines and characters) and black background as shown in the upper part of Figure 4. In the next step, a horizontal and vertical erosion kernel $k_h, k_v \in \mathbb{R}^2$ are applied independently to the inverted image. It is worth mentioning that the kernel shape and size is not fixed and can be set for both erosion kernels. The erosion kernels are generally thin vertical and horizontal strips that are longer than the overall font size but shorter than the size of the smallest grid cell and, in particular, must not be wider than the smallest table border width. Using these kernel size constraints results in the erosion operation removing all fonts and characters from the table while preserving the table borders. Since the erosion operation is keeping the minimum pixel value from the kernel overlay, its application leads to shorter lines compared to the original table borders. In order to restore the original line shape, the algorithm applies the dilation operation using the same kernel size on each of the two eroded images like shown in the middle part of Figure 4, producing an image with vertical and a second with horizontal lines. The dilation operation rebuilds the lines by keeping only the maximum

pixel value from the kernel overlay of the image. Finally, the algorithm combines both images by using a *bit-wise or* operation and re-inverting the pixel values to obtain a raster cell image, as shown in the lower part of Figure 4. We then use the contours function on the grid-cell image to extract the bounding-boxes for every single grid cell.

4.4 Unbordered TSR

The TSR algorithm for unbordered tables works similarly to the one for bordered tables. It also starts with converting the image to a binary representation. However, unlike the first algorithm it does not invert the pixel values straight away and also does not utilize the dilation operation. Furthermore, it uses a different kind of erosion compared to TSR for bordered tables. The erosion kernel is in general a thin strip with the difference that the horizontal size of the horizontal kernel includes the full image width and the vertical size of the vertical kernel the full image height. The algorithm slides both kernels independently over the whole image from left to right for the vertical kernel, and from top to bottom for the horizontal kernel. During this process it is looking for empty rows and columns that do not contain any characters or font. The resulting images are inverted and combined by a *bit-wise and* operation producing the final output as shown in the middle part of Figure 5. This final output is a grid-cell image similar to the one from TSR for bordered tables, where the overlapping areas of the two resulting images represent the bounding-boxes for every single grid cell as shown in the right part of Figure 5 which displays the grid cells produced by our TSR algorithm and the corresponding text.

4.5 Partially Bordered TSR

To handle all types of tables, an algorithm for partially bordered tables is needed. The main goal of our algorithms for bordered and unbordered tables is to create a grid cell image by adding borders in the unbordered case and detecting lines in the bordered case. If a table is only partially bordered, then the unbordered algorithm is prevented to add borders in orthogonal direction to the existing borders, while the bordered algorithm can only find the existing borders.

Figure 7: Color invariance pre-processing example

Both approaches result in incomplete grid cell images. So the question is how to obtain an algorithm that produces a grid cell image for partially bordered tables. The main idea is to detect the existing borders as done by the algorithm for bordered tables, but without using them to create a grid cell, but to delete the borders from the table image to get an unbordered table (see Figure 6 for an example). This allows then for applying the algorithm for unbordered tables to create the grid-cell image and contours by analogy to the variants discussed above. A key feature of this approach is that it works with both bordered and unbordered tables: it is type-independent.

4.6 Color Invariance Pre-Processing

A main goal of this work is to create a multi-level pipeline for TD and TSR that works on all types of documents with tables. To this end, we addressed the problem of image rotation, detected tables in images, and developed an algorithm that can handle all three types of table margins. The quest is then whether this approach can handle different colors. In general, we do not need to treat colors with 3 different channels as in the case of RGB images, for example, because we convert table images to binary images based on the contrast of font and background colors. All algorithms proposed so far require a white background and a black font. But the resulting binary image could have a black background and white font, or cells with a black background and white font, while others have a white background and black font as shown in Figure 7. Therefore, to obtain table images of the desired color type, we insert an additional image processing step between TD and TSR. This step also searches for contours, but now for counting black and white pixels per contour: if there are more black pixels than white pixels, the colors of the contour are inverted. This procedure results in backgrounds being white and fonts being black.

5 EVALUATION

To enable comparability of Multi-Type-TD-TSR with other state-of-the-art approaches [10], we reuse their datasets. This concerns a dataset for the TSR task that was extended by manually annotating selected images from the ICDAR 19

(Track B Modern) training data [3]. Prasad et al. [10] randomly chose 342 images out of 600 of the ICDAR 19 training set to get 534 tables and 24,920 cells, with all these entities annotated accordingly. The reason for using only ICDAR 19 data is that the ground truth information available for the images as provided by the TableBank dataset [9] for TSR contains only table structure labels in the form of HTML tags. It does not provide cell or column coordinates and therefore cannot be used to evaluate object detection performance. IC-DAR 13 [4] is also not usable for evaluating TSR because its evaluation metric uses the textual content of the cell-based mapping of predicted cells to ground truth ones. This requires the extraction of text content with an OCR engine, so that the overall accuracy ultimately depends on the accuracy of the OCR.

Our algorithm recognizes only cells as part of the overlap from recognized rows and columns. To allow a fair comparison, we manually re-annotated the dataset with respect to the cells that our algorithm can recognize at all. An annotation example of both annotation types is shown in Figure 8.

To validate our TSR algorithm, we need to determine the best kernel sizes for the horizontal and vertical kernel. For this purpose, we used a random search to find the best values for the width of the vertical kernel and the height of the horizontal kernel. We determined the best width to be 8 and the best height to be 3 pixel units.

For the final evaluation, we used the type independent algorithm for partially bordered tables, since it is the one we would be deploying in a real world application where we do not have any information about the respective table types. We evaluate using F1-scores by analogy to [10] with IoU (*Intersection over Union*) thresholds of 0.6, 0.7, 0.8, and 0.9, respectively. The results are shown in Table 1. We achieved

Team	IoU 0.6	IoU 0.7	IoU 0.8	IoU 0.9	Weighted Average
CascadeTabNet	0.438	0.354	0.19	0.036	0.232
NLPR-PAL	0.365	0.305	0.195	0.035	0.206
Multi-Type-TD-TSR	0.589	0.404	0.137	0.015	0.253

Table 1: F1-score performances on ICDAR 19 Track B2 (Modern) [3]

the highest F1-score by using a threshold of 0.6 and 0.7. When using higher thresholds (0.8 and 0.9), we encounter a clear performance decrease, which also applies for the other two algorithms we are comparing with. According to the overall result, we conclude that Multi-Type-TD-TSR reaches the highest weighted average F1-score as well as the highest

a)   b)

Figure 8: Annotation example from: a) the original validation dataset of [10], b) the manually labeled validation dataset of Multi-Type-TD-TSR.

overall performance of 0.589, thus representing a new state-of-the-art.

6 CONCLUSION

We presented a multistage pipeline for table detection and table structure recognition with document alignment and color invariance pre-processing. For this purpose, we distinguished three types of tables, depending on whether they are borderless or not. Because of the unavailability of large labeled datasets for table structure recognition we decided to use two conventional algorithms: The first one can handle tables without borders, the second one can handle tables with borders. Further, we combined both algorithms into a third, conventional table structure recognition algorithm that can handle all three types of tables. This algorithm achieves the highest F1-score among the systems compared here for an IoU threshold of 0.6 and 0.7, but does not detect sharp borders, so the F1-score decreases rapidly for higher thresholds 0.8 and 0.9. However, the highest weighted averaged F1-scores obtained by our algorithm show the potential of our multi-type approach, which can handle all three table types considered here: it benefits from using one of the specialized algorithms to transform the input tables so that they can be optimally processed by the other specialized algorithm. This kind of multi-stage division of labor among otherwise conventional algorithms could help to finally bring such a difficult task as table structure recognition into domains that allow downstream NLP procedures to process linguistic table contents properly. This paper made a contribution to this difficult task.

7 FUTURE WORK

The presented table structure recognition algorithms treat an approximation of the table structure recognition problem, because they assume that tables only contain cells as determined by the intersection of rows and columns. In general the problem of table structure recognition is even more difficult, since tables can recursively consist of more complex cells as found in tables with multi-rows or multi-columns. Cells can

again consist of cells or contain entire tables, so that table structure recognition ultimately involves the recognition of recursive structures, which makes this task very difficult to handle for conventional computer vision algorithms.

The recent success of ML is due in part to the availability of large amounts of annotated data. For table structure recognition such a dataset is not yet available so that many data driven algorithms use transfer learning to bypass this problem. Obviously, larger datasets will lead to more general and better algorithms. Future work will therefore probably focus on the development of such datasets for table structure recognition.

REFERENCES

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 39, 12 (2017), 2481–2495.

[2] Corinna Cortes and Vladimir Vapnik. 1995. Support vector machine. *Machine learning* 20, 3 (1995), 273–297.

[3] Liangcai Gao, Yilun Huang, Hervé Déjean, Jean-Luc Meunier, Qinqin Yan, Yu Fang, Florian Kleber, and Eva Lang. 2019. ICDAR 2019 Competition on Table Detection and Recognition (cTDaR). In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 1510–1515. https://doi.org/10.1109/ICDAR.2019.00243

[4] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. 2013. ICDAR 2013 Table Competition. In *2013 12th International Conference on Document Analysis and Recognition*. 1449–1453. https://doi.org/10.1109/ICDAR.2013.292

[5] Azka Gilani, Shah Rukh Qasim, Imran Malik, and Faisal Shafait. 2017. Table detection using deep learning. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, Vol. 1. IEEE, 771–776.

[6] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014).

[7] Thotreingam Kasar, Philippine Barlas, Sebastien Adam, Clément Chatelain, and Thierry Paquet. 2013. Learning to detect tables in scanned document images using line information. In *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 1185–1189.

[8] Thomas Kieninger and Andreas Dengel. 1998. The t-recs table recognition and analysis system. In *International Workshop on Document Analysis Systems*. Springer, 255–270.

[9] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. 2020. TableBank: Table Benchmark for Image-based Table Detection and Recognition. In *Proceedings of the 12th Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, 1918–1925. https://www.aclweb.org/anthology/2020.lrec-1.236

[10] Devashish Prasad, Ayan Gadpal, Kshitij Kapadni, Manish Visave, and Kavita Sultanpure. 2020. CascadeTabNet: An approach for end to end table detection and structure recognition from image-based documents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 572–573.

[11] P Pyreddi and W Bruce Croft. 1997. A system for retrieval in text tables. In *ACM DL*.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497* (2015).

[13] Mohammad Mohsin Reza, Syed Saqib Bukhari, Martin Jenckel, and Andreas Dengel. 2019. Table localization and segmentation using gan and cnn. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, Vol. 5. IEEE, 152–157.

[14] Adrian Rosebrock. 2017. Text skew correction with OpenCV and Python. PyImageSearch, https://www.pyimagesearch.com/2017/02/20/text-skew-correction-opencv-python/, accessed on 17 February 2021.

[15] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. 2017. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, Vol. 1. IEEE, 1162–1167.

[16] Wonkyo Seo, Hyung Il Koo, and Nam Ik Cho. 2015. Junction-based table detection in camera-captured document images. *International Journal on Document Analysis and Recognition (IJDAR)* 18, 1 (2015), 47–57.

[17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 1492–1500.