Feature Engineering meets Deep Learning: A Case Study on Table Detection in Documents

Muhammad Ali Shahzad*‡, Rabeya Noor*, Sheraz Ahmad‡, Ajmal Mian§, and Faisal Shafait*†
*School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST), Islamabad, Pakistan
†Deep Learning Laboratory, National Center of Artificial Intelligence (NCAI), Islamabad, Pakistan
‡German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
§The University of Western Australia, Perth, Australia
Email: faisal.shafait@seecs.edu.pk

***Abstract*—Traditional computer vision approaches heavily relied on hand-crafted features for tasks such as visual object detection and recognition. The recent success of deep learning in automatically extracting representative and powerful features from images has brought a paradigm shift in this area. As a side effect, decades of research into hand-crafted features is considered outdated. In this paper, we present an approach for table detection in which we leverage a deep learning based table detection model with hand-crafted features from a classical table detection method. We demonstrate that by using a suitable encoding of hand-crafted features, the deep learning model is able to perform better at the detection task. Experiments on publicly available UNLV dataset show that the presented method achieves an accuracy comparable with the state-of-the-art deep learning methods without the need of extensive hyper-parameter tuning.**

***Index Terms*—document image analysis, table detection, machine learning**

I. Introduction

Object detection in images has been an active topic of research for decades. The traditional pipeline for object detection consists of a feature extractor followed by a classifier. Scale Invariant Feature Transform [1] made a break-through in object detection by introducing a carefully designed hand engineered feature extractor. The success of this method has lead to many other feature extraction algorithms such as SURF [2], Histograms of Oriented Gradients [3], ORB [4] and many others. A similar trend was followed in document analysis research, where different tasks such as detection of mathematical symbols [5], graphical content [6], page segments [7], and tables [8], [9] were powered by hand-engineered features.

The success of deep learning in large scale object recognition in recent years has demonstrated the capability of deep networks in automatically learning representative features from a large amount of data [10]. Advances in transfer learning algorithms have made it possible to adapt these learned features to new domains for which only a limited amount of training data is available. For instance, deep networks trained on ImageNet data, which comprises of natural images of various objects, have been shown to perform surprisingly well on various document analysis tasks such as document classification [11] and table detection [12].

In this paper, we argue that even though deep learning algorithms are able to extract representative features from large training sets, their representation capability remains a function of the amount of data used for training. More specifically, in scenarios where a limited amount of training data is available, we can leverage decades of research that went into feature engineering to further increase the performance of deep networks. We take the task of table detection as a case study to show how hand engineered features can be embedded in an image to facilitate subsequent transfer learning for a deep network.

Tables are widely used for displaying essential information in documents in a structured form. They are used in research articles, print media, books and many other types of documents. They show a large amount of information concisely so that it is easy to understand by the readers, highlighting the important information. Therefore, table detection plays an important role in the area of document analysis and recognition. Moreover, tables have a diverse range of layouts and designs without any standardization. Different tables have different features. This is why table detection is regarded as a difficult problem for a general solution to be applicable. Hence we chose table detection in documents as a case study for implementing deep learning algorithm that also incorporates hand engineered features.

For table detection, different heuristic techniques have been proposed in the literature which work on the specific formats of the tables but fail on others. Only recently, this problem is greatly reduced by applying deep learning techniques as they provide much better results than the previous methods.

Faster-RCNN [13], which is a deep learning technique used on natural images for object detection can be used on documents to detect tables. However, this technique requires a large amount of data set to produce good results. This problem is tackled by using transfer learning and domain adaptation [14]. This paper presents the following approach to the table detection problem. First, a document is pre-processed and then fed into a Faster-RCNN model. The processing stage is divided into two parts; color coding and image transformation. In color coding, blocks that are classified as tables are extracted through the T-Recs algorithm (T-Recs) [8], [15],

[16]. These blocks are then color coded (details to follow). Then the document image is transformed into a natural image for Faster-RCNN. Thus, in our approach, we have increased the accuracy of Faster-RCNN by providing it with external features, thereby merging hand engineering and deep learning techniques. This system is evaluated on UNLV dataset [14] which is publicly available.

The rest of this paper is organized as follows. Section II contains literature review and Section III elaborates our approach, explaining both the processing stage as well as the detection stage. Section IV presents the performance evaluation and the experimental results. Finally, Section V gives the conclusive remarks as a guide for future endeavours in this domain.

II. Related Work

The problem of table detection has been addressed by many researchers. Table detection techniques reported in the literature can be divided into two classes; traditional rule-based approaches using feature engineering and the recently proposed machine learning based techniques. First, we will discuss a few traditional approaches and then we will focus on machine learning based solutions.

There have been many prominent works done by researchers in the field of table detection. One of the well-known approaches, proposed by Kieninger et al. [8], [15], [16] is called T-Recs. This algorithm forms word bounding boxes and following a bottom-up approach they are grouped according to their logical units. The issue in this approach is that it does not perform well on multi-column layouts and is dependent on the word bounding boxes.

Hu et al. [17] proposed an algorithm consisting of a high-level framework which takes n number of lines as input and groups a number of lines to form a table by considering a measure of confidence. The measures to check the confidence include merit, scores and correlation of lines. However, the drawback of this approach is that it does not work on multi-column table layouts.

Wang et al. [18] presented a statistical approach for table detection. First, the document is pre-processed to label the column structure. Then, table entity candidates are defined by detecting large blank blocks. Finally, it uses a statistical-based table refinement algorithm to reduce the false positives and verify the results. The limitation of this algorithm is that it only works on a specific document layout. Gatos et al. [19] proposed another approach for table detection. The document images are pre-processed, and then horizontal and vertical lines are estimated after removing text lines. Then, the areas of intersections are calculated and are grouped to form a table. The constraint of this method is that it does not work in the absence of ruling lines.

Mandal et al. [20] proposed an algorithm relying on the observation of the presence of large gaps between the fields of the table as compared to gaps in text lines. Therefore, ruling lines were ignored and as a result removed from tables before detection. This removal led to the poor performance of the algorithm. Costa e Silva [21] used Hidden Markov Models for table detection. Tables from PDFs are used as a dataset. The PDFs are first converted into ASCII by pdftotext Linux utility. This method only works on PDF documents and is unable to correctly identify the first and the last table lines.

Hao et al. [22] presented the first approach of table detection using deep learning. The candidate tables are selected based on ruling line features present in a PDF. After these candidates are passed through a convolutional neural network, the tables are identified. However, this approach only works on PDF documents and is not able to detect multi-column tables.

Rashid et al. [23] used machine learning for table detection. They used a bottom-up approach where features of each word are extracted as feature vectors which are then used for training an AutoMLP classifier. These feature vectors consist of features such as the distance of each word from its neighbour, width and height of word as well as manual features such as white space.

Schreiber et al. [12] presented a data-driven, deep learning approach for table detection. This approach uses Faster RCNN for table structure recognition. Gilani et al. [14] converted document images to natural images by applying distance transform techniques and then fed them to Faster R-CNN. Although this approach out-perfomed the state-of-the-art Tesseract table detection [9] in accuracy, it does not take into account the background and foreground features of the tables. Arif et al. [24] further improved this work by pre-processing the document image in two phases. Observing that the tables contain more numeric data than textual, they color coded both types of data to assist the deep learning algorithm. In the second step, they used image transformation from [14] on document images and fed them to Faster R-CNN. This approach showed even better results on UNLV data-set.

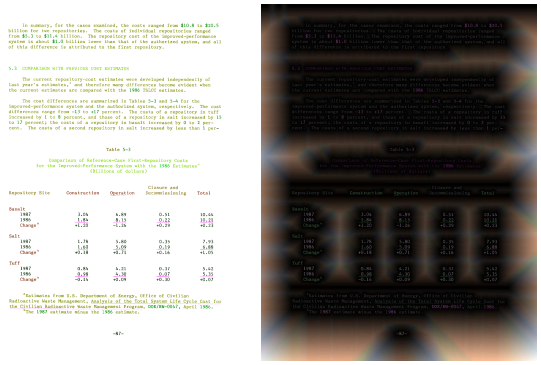
III. METHODOLOGY

The approach presented in this paper makes use of both background and foreground features. Foreground features are encoded using the traditional feature engineering approach and passed to the deep learning module as an input. The background information is encoded and added to all three channels taking inspiration from Gilani et al. [14]. Our approach can be broken down into three core steps:

1) Foreground Feature Engineering
2) Background Feature Engineering
3) Deep Learning on Encoded Features

A. Foreground Feature Engineering

Foreground feature engineering process takes its first inspiration from Kieninger et al. [16] which relies on bottom-up clustering of word segments to extract table structure information. In contrast to other feature engineering approaches that use nearest neighbours [25] or run-length smoothing [26], this approach does not look for separators of any kind. It identifies and clusters the words that belong to the same logical group. This makes it a robust approach when applied to the documents of different layouts and formats. Our approach starts by creating a segmentation graph, which considers horizontally



(a) Feature Encoded Image (b) Transformed Image

Fig. 1: Feature Engineered Images without background (a) with Background Transformation (b)

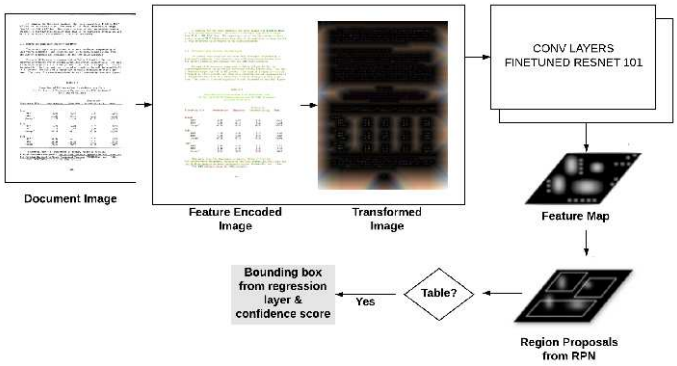


Fig. 2: Proposed approach: Features are engineered and color coded. Distance transform is applied to the original image shown in Figure 1b. Both images are added and the resultant image is be fed to the feature extractor. Feature map generated by the feature extractor is then passed to the region proposal network (RPN) which proposes the regions where tables might be present. The detection network processes the proposed regions as input and classifies them into table or non-table regions.

overlapping word neighbours in the previous and the next line. All the overlapping words are grouped together to form text blocks. The equation from [16] imposes the condition for two words to belong to the same block i.e. if two words to x-axis have a common range and they are located in subsequent lines

$$\begin{aligned} \text{ovl}(w_1, w_2) \iff (x_1[w_1] \geq x_0[w_0]) \wedge (x_0[w_1] \leq x_1[w_2]) \wedge \\ (\text{line}(w_1) = \text{succ}_{\text{doc}}(\text{line}(w_2) \vee \\ \text{line}(w_2) = \text{succ}_{\text{doc}}(\text{line}(w_1)) \end{aligned} \tag{1}$$

Where ovl(w_1,w_2) is a symmetrical binary relation, w_i represents the words, line represents an aggregation of words, "succ" represents the appropriate successor function and x_j represents geometric coordinates [8].

The clusters created by the segmentation graph are then tagged into logical groups. All blocks that consist of a single word per line are classified as *type 1* and everything else is classified into a broader category called *type 2*. Segmentation graph approach creates isolated words, those words are not clustered to any block because they do not have any neighbor in the line above or the line below. These words are the candidates for the table header. A block is marked as a paragraph if the block is of *type 2* and its width exceeds a defined threshold. Compact types are the blocks which have width equivalent of maximum *type 2* cell width multiplied by the document width.

We take our second inspiration from Arif et al. [24] that the tables generally represent more numeric information than the textual information. In their paper, red color is allocated to numeric data, whereas green color is allocated to textual data as part of foreground encoding. Our approach encodes the numeric and textual information in a single red channel with different color codes. The numeric information is given pixel value 255 and text information is given a pixel value of 128. Block *type 1, type 2, type paragraph* and *type compact* encoding are added to the green channel with equal gray value spacing. Blue channel contains an average of the red and green channel.

B. Background Feature Engineering

Distance transformation is an effective approach to separate the background from the foreground text. It calculates the distance between the text region and the background of the image. Tables generally contain sparse text providing reliable indication of where a table might be present. We experimented with the background encoding techniques used by Arif et al. [24] and Gilani et al. [14] and found that the latter technique outperforms the former. Euclidean transformation is applied to the blue image channel, linear distance transformation is applied to the green image channel and max distance transformation to the red image channel. All these distance transformations are added to the respective channels of the feature encoded image.

C. Deep Learning on Encoded Features

Object detection algorithms have been effective on table detection problem as demonstrated by [24] and [14]. We have used Faster R-CNN due to its compelling accuracy on large set object detection problems in different domains. Faster R-CNN has two modules running serially. The first one is the Region Proposal Network (RPN) which generates the candidate regions for the Detection Network. The second one is the Detection Network which classifies the candidates as Tabular regions. The Faster R-CNN works faster than its predecessor Fast R-CNN, as it replaces selective search with a robust network called region proposal network (RPN). Figure 2 shows the proposed approach.

1) Feature Extractor: Feature extractor selection is an important step. The original Faster R-CNN used Simonyan and Zisserman model (VGG-16) [27] and Zeiler and Fergus model (ZF) [28] pre-trained on ImageNet dataset [29] as a feature

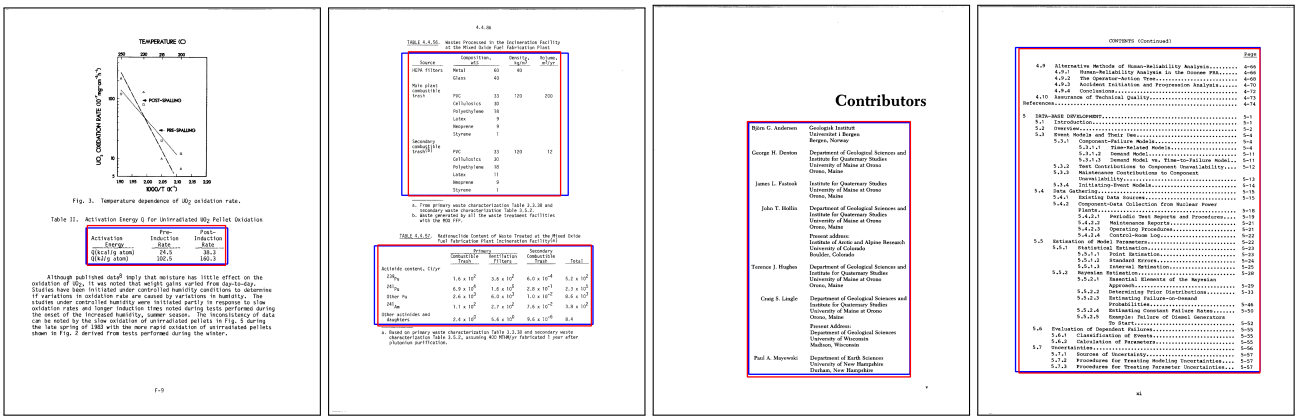


Fig. 3: Selected images from the UNLV dataset demonstrating the results of our proposed approach, red boxes are the detection results, Blue are the ground-truth

extractor. However, we have employed ResNet101 which is trained on the KITTI data-set.

2) Region Proposal Network: Region Proposal Network takes feature map produced by feature extractor and outputs the bounding box along with the objectness score. RPN takes each position in the last convolutional feature map and generates k candidate anchors of different scales and ratios. We have employed default configuration of Faster R-CNN which uses three scales and three aspect ratios and outputs $k = 9$ anchors. RPN generates region proposals by sliding a small network of $n \times n$ over the last feature map. By doing this, it generates a lower dimensional feature map which is then passed to two 1×1 convolutional layers called box classification and regression layers. Box classification layer classifies that the bounding box output by the box regression layer is a table region or a background. Box regression layer produces 4 outputs, each set of four outputs provide the box location. Non-maximum suppression is used to minimize the number of bounding box proposals.

3) Detector Network: Region proposals from RPN module are passed to the detection module which utilizes these proposals to detect tables and return the box coordinates and confidence scores of the candidates that are finally selected as tables.

4) Training: A large amount of diverse data-set is required to train a deep neural network from scratch. In case the data is not available, fine-tuning of existing pre-trained deep models is a viable option. Our training data-set contains 1,274 images collected by [24]. We have not done any data augmentation to keep the dataset comparable with [24] and highlight the accuracy gain achieved with our approach. We have used transfer learning and domain adaptation to fine-tune the weights of ReNet101, pre-trained on KITTI data-set [30], with learning rate of 0.001 for first 50,000 iterations and 0.0001 for next 20,000 iterations. We have trained our model on 70,000 iterations and choose the best set of weights by looking at the training and validation curves. Looking at the Training and Validation accuracy graphs, the model converges at $\tilde{3}5{,}000$ steps. We have split our training data-set of 1,274 into 80:20 for training and validation to detect over-fitting during the training process. The trained network was then evaluated on the publicly available UNLV data-set. Note that no part of the UNLV data-set was used for training.

IV. EXPERIMENTS AND RESULTS

This section provides details about the evaluation protocols, results comparison and analysis of the data used for training and evaluation.

A. Performance Measures

Algorithms for table detection are evaluated through different measures. We have evaluated our approach by precision, recall and F1 score. Several table detection algorithms [7], [18], [20], [31], [32], were evaluated through precision and recall. We used Schreiber et al. [12], Gilani et al. [14] and Arif et al. [24] to compare the accuracy of our method.

Let G_i stand for ground truth bounding boxes and D_j for our detected bounding boxes. The formula through which the region that is overlapped by G_i and D_j is computed as:

$$A(G_i, D_j) = 2 \times \frac{|G_i \cap D_j|}{|G_i| + |D_j|}, \quad A\epsilon[0, 1] \tag{2}$$

The value of $A(G_i \cap D_j)$ will be between zero and one where zero shows that areas of bounding boxes G_i and D_j do not match at all while one shows complete match between G_i and D_j.

1) Precision:: Precision is used to measure the performance of table detection algorithm as a whole by calculating the percentage of detected tables belonging to table areas of ground truth table. Precision is calculated by the following formula:

$$\text{Precision} = \frac{\text{Area of Ground truth regions in Detected regions}}{\text{Area of all Detected table regions}} \tag{3}$$

TABLE 1

PERFORMANCE COMPARISON OF DIFFERENT TABLE DETECTION APPROACHES

Performance Measure	Accuracy (%)			
	Schreiber et al. [12]	Gilani et al. [14]	Arif et al. [24]	Our Approach
Precision	72.67	77.28	86.33	**88.39**
Recall	89.45	90.88	**93.21**	91.04
F1 Score	80.19	83.53	89.64	**89.70**

2) Recall:: Recall is found by calculating the number of table regions checked as detected table areas as a percentage using the following formula:

$$\text{Recall} = \frac{\text{Area of Ground truth regions in Detected regions}}{\text{Area of all Ground truth table regions}} \quad (4)$$

3) F1 Score:: The accuracy calculated through F1 score is composed of both precision and recall as follows:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)$$

B. Experiments and Results:

To test the performance of the proposed technique we have used the publicly available UNLV data-set which covers diverse formats and document layouts such as technical reports, research articles, newspapers etc. A total of 2,889 scanned pages are available in the data-set out of which only 427 document images contain 570 table regions [24]. Our approach uses features from a classical approach and encodes them in a suitable format for the deep learning algorithms to consume. We demonstrate that this approach works reasonably well based on the performance metrics described earlier. The sample results of the proposed approach are shown in Figure 3.

The proposed approach is benchmarked against Schreiber et al. [12], Gilani et al. [14] and Arif et al. [24] where Arif et al. is the state-of-the-art table detection technique. We consider Schreiber et al. [12] as the baseline for our experiment as their approach provides the unprocessed image as an input to the Faster R-CNN for table detection. This gives a key insight into the benefits of using traditional techniques to aid the deep learning algorithm in making an inference. All of these techniques use Faster R-CNN for detection and are evaluated on the UNLV data-set. Hence, the experiment provides a yardstick on the effect of encoding hand-engineered features in the image which is then passed to a deep learning algorithm.

We have trained the approaches by Schreiber et al. [12], Gilani et al. and Arif et al. [24] on the data-set proposed by Arif et al. [24] and compared their generalization on UNLV data-set with 427 table images. UNLV data-set was not used in any part of the training process. Table 1 provides a detailed comparison of our approach with Schreiber et al. [12], Gilani et al. [14] and Arif et al. [24] using precision, recall and F1 score metrics. Our approach achieved a Mean Area Precision of 88.39% outperforming Arif et al. [24] 86.33%, Gilani et al. [14] 77.28% and Schreiber et al. [12] 72.67%. Our approach's Recall rate is 91.04%, which is higher than Gilani et al.'s 90.88% and Schreiber et al.'s 89.45% but lower than Arif et al.'s 93.21%. Finally, our approach's F1 score is 89.70% which our-performs Arif et al.'s 89.64, Gilani et al.'s 83.53% and Schreiber et al.'s 80.19%.

C. Analysis:

In-depth analysis of the results unfolds some interesting facts. Firstly, the ground-truth for UNLV data-set was targeted at a classical feature engineering approach which considered ruling lines as table boundaries from the rest of the document as shown in the Figure 4a and 4b. Our approach does not encode ruling lines as a feature for the algorithm to learn. The boxes created by the algorithm are regressed around the text region rather than the ruling lines. Secondly, the training data prepared in [24] does not contain any images from magazines and newspapers. Newspapers and magazines have document structure that may span up to four columns and the font-size is generally smaller than that used in documents like research papers and other articles. Our approach considers the documents with more than two columns a table in itself considering the absence of representative examples as shown in 4c and 4d. Encoding of ruling lines as a feature into one channel and improving the training data to contain document images from newspapers and magazines have the potential to improve the overall accuracy of the algorithm.

V. CONCLUSION

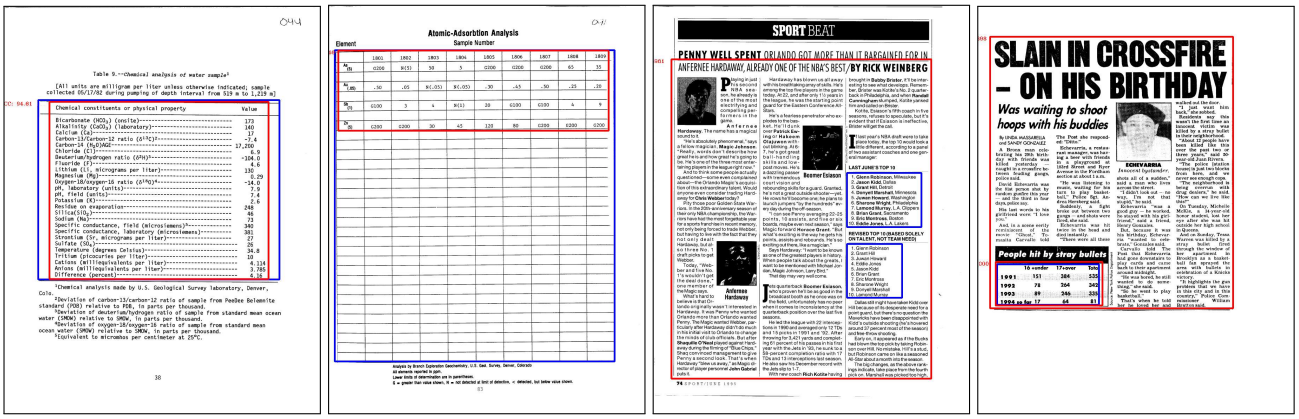
We presented an alternate approach towards solving complex deep learning problems. The primary focus of this method is to make use of hand-crafted features. We demonstrated its effectiveness by encoding traditional hand-crafted block structure features, text and numeric features, as well as background features in the image. These images are then fed to a deep learning network (Faster R-CNN) which generates the detection results. We have used publicly available UNLV data-set for evaluating our approach and achieved results comparable to the best-performing deep learning methods on this dataset.

REFERENCES

[1] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *International Conference on Computer Vision and Pattern Recognition* (C. Schmid, S. Soatto, and C. Tomasi, eds.), vol. 1, (San Diego, United States), pp. 886–893, IEEE Computer Society, 2005.



(a) Horizontal Ruling lines stretching ground-truth box horizontally

(b) Ruling lines in the grid structure extending the ground-truth box vertically

(c) Three column document structure resulting in false detection

(d) Four column document structure resulting in false detection

Fig. 4: Presents the challenges to support our Analysis and proposed direction to improve the detection results.

[4] G. Bradski, K. Konolige, V. Rabaud, and E. Rublee, "Orb: An efficient alternative to sift or surf," in *IEEE International Conference on Computer Vision*, (Barcelona, Spain), pp. 2564–2571, 2011.

[5] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *International Journal on Document Analysis and Recognition*, vol. 15, no. 4, pp. 331–357, 2012.

[6] N. Nayef and T. M. Breuel, "On the use of geometric matching for both: Isolated symbol recognition and symbol spotting," in *Graphics Recognition. New Trends and Challenges*, (Berlin, Heidelberg), pp. 36–48, Springer Berlin Heidelberg, 2013.

[7] F. Shafait, D. Keysers, and T. Breuel, "Performance evaluation and benchmarking of six-page segmentation algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 941–954, 2008.

[8] T. Kieninger and A. Dengel, "Applying the t-recs table recognition system to the business letter domain," in *International Conference on Document Analysis and Recognition*, p. 0518, 2001.

[9] F. Shafait and R. Smith, "Table detection in heterogeneous documents," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, document analysis systems, pp. 65–72, 2010.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[11] M. Z. Afzal, S. Capobianco, M. I. Malik, S. Marinai, T. M. Breuel, A. Dengel, and M. Liwicki, "Deepdocclassifier: Document classification with deep convolutional neural network," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1111–1115, 2015.

[12] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," in *Fourteenth International Conference on Document Analysis and Recognition*, vol. 1, pp. 1162–1167, 2017.

[13] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[14] A. Gilani, S. R. Qasim, I. Malik, and F. Shafait, "Table detection using deep learning," in *14th International Conference on Document Analysis and Recognition*, pp. 771–776, 2017.

[15] T. Kieninger and A. Dengel, "A paper-to-html table converting system," in *Proceedings of document analysis systems*, pp. 356–365, 1998.

[16] T. Kieninger and A. Dengel, "Table recognition and labeling using intrinsic layout features," in *International Conference on Advances in Pattern Recognition*, pp. 307–316, 1999.

[17] J. Hu, R. S. Kashi, D. P. Lopresti, and G. Wilfong, "Medium-independent table detection," in *Document Recognition and Retrieval VII*, vol. 3967, pp. 291–303, 2000.

[18] Y. Wangt, I. Phillipst, and R. Haralick, "Automatic table ground truth generation and a background-analysis-based table structure extraction method," in *Sixth International Conference on Document Analysis and Recognition*, pp. 528–532, 2001.

[19] B. Gatos, D. Danatsas, I. Pratikakis, and S. J. Perantonis, "Automatic table detection in document images," in *International Conference on Pattern Recognition and Image Analysis*, pp. 609–618, 2005.

[20] S. Mandal, S. Chowdhury, A. K. Das, and B. Chanda, "A simple and effective table detection system from document images," *International Journal of Document Analysis and Recognition*, vol. 8, no. 2-3, pp. 172–182, 2006.

[21] A. C. e Silva, "Learning rich hidden markov models in document analysis: Table location," in *Tenth International Conference on Document Analysis and Recognition*, pp. 843–847, 2009.

[22] L. Hao, L. Gao, X. Yi, and Z. Tang, "A table detection method for pdf documents based on convolutional neural networks," in *12th IAPR Workshop on Document Analysis Systems*, pp. 287–292, 2016.

[23] S. F. Rashid, A. Akmal, M. Adnan, A. A. Aslam, and A. Dengel, "Table recognition in heterogeneous documents using machine learning," in *Fourteenth International Conference on Document Analysis and Recognition*, vol. 1, pp. 777–782, 2017.

[24] S. Arif and F. Shafait, "Table detection in document images using foreground and background features," in *Digital Image Computing: Techniques and Applications 2018*, pp. 1–8, 2018.

[25] L. O'Gorman, "The document spectrum for page layout analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 15, pp. 1162–1173, 1993.

[26] K. Y. Wong, R. G. Casey, and F. M. Wahl, "Document analysis system," *IBM Journal of Research and Development*, vol. 26, pp. 647–656, 1982.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[28] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, 2014.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[31] J. Hu, R. S. Kashi, D. P. Lopresti, and G. Wilfong, "Evaluating the performance of table processing algorithms.," in *International Journal on Document Analysis and Recognition*, vol. 4, pp. 140–153, 2002.

[32] A. Shahab, F. Shafait, T. Kieninger, and A. Dengel, "An open approach towards the benchmarking of table structure recognition systems," in *Document Analysis Systems*, pp. 113–120, 2010.