PingAn-VCGroup's Solution for ICDAR 2021 Competition on Scientific Literature Parsing Task B: Table Recognition to HTML

Jiaquan Ye[1], Xianbiao Qi[1], Yelin He[1], Yihao Chen[1], Dengyi Gu[1], Peng Gao[2], and Rong Xiao[1]

[1]Visual Computing Group, Ping An Property & Casualty Insurance Company
[2]Ping An Technology Company

May 6, 2021

Abstract

This paper presents our solution for ICDAR 2021 competition on scientific literature parsing task B: table recognition to HTML. In our method, we divide the table content recognition task into four sub-tasks: table structure recognition, text line detection, text line recognition, and box assignment. Our table structure recognition algorithm is customized based on MASTER [1], a robust image text recognition algorithm. PSENet [2] is used to detect each text line in the table image. For text line recognition, our model is also built on MASTER. Finally, in the box assignment phase, we associated the text boxes detected by PSENet with the structure item reconstructed by table structure prediction, and fill the recognized content of the text line into the corresponding item. Our proposed method achieves a 96.84% TEDS score on 9,115 validation samples in the development phase, and a 96.32% TEDS score on 9,064 samples in the final evaluation phase.

1 Introduction

The ICDAR 2021 competition on scientific literature parsing task B is to reconstruct the table image into an HTML code. In this competition, PubTabNet dataset (v2.0.0) [3] is provided as the official evaluation data, and Tree-Edit-Distance-based similarity (TEDS) metric is used for evaluation. The PubTabNet data set consists of 500,777 training samples, 9,115 validation samples, 9,138 samples for the development stage, and 9,064 samples for the final evaluation stage. For the training and validation data, the ground truth HTML codes and the position of non-empty table cells are provided to the participants. Participants of this competition need to develop a model that can convert images of tabular data into the corresponding HTML code, which should correctly represent the structure of the table and the content of each cell. The labels of samples for the development and the final evaluation stages are preserved by the organizers.

We divide this task into four sub-tasks: table structure recognition, text line detection, text line recognition, and box assignment. And several tricks are tried to improve the model. The details of each sub-task will be discussed in the following section.

2 Method

In this section, we will present these four sub-tasks in order.

*Xianbiao Qi is the corresponding author. If you have any questions or concerns about the implementation details, please do not hesitate to contact jiaquanye@qq.com or qixianbiao@gmail.com.

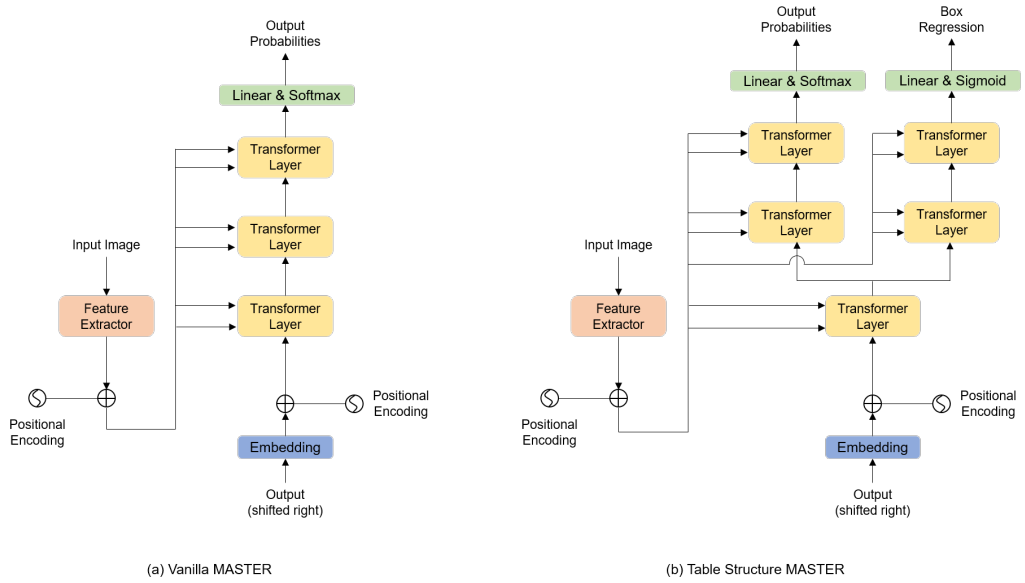


Figure 1: (a) Architecture of vanilla MASTER; (b) Architecture of table structure MASTER

2.1 Table Structure Recognition

The task of table structure recognition is to reconstruct the HTML sequence items and their corresponding locations on the table, but ignore the text content in each item. Our model structure is shown in Figure 1(b). It is customized based on MASTER [1], a powerful image-to-sequence model originally designed for scene text recognition. Different from the vanilla MASTER as shown in Figure 1(a), our model has two branches. One branch is to predict the HTML item sequence, and the other is to conduct the box regression. Instead of splitting the model into two branches in the last layer, we decouple the sequence prediction and the box regression after the first transformer decode layer.

<thead>	rowspan="2"	colspan="2"	<td></td>	<td></td>
</thead>	rowspan="3"	colspan="3"	<td> </td>	<td><sep></sep></td>
<tbody>	rowspan="4"	colspan="4"	<td> </td>	</td>
</tbody>	rowspan="5"	colspan="5"	<td><sep></sep></td>	
<tr>	rowspan="6"	colspan="6"	<td><sup> <sup></td>	
</tr>	rowspan="7"	colspan="7"	<td></td>	
<td></td>	rowspan="8"	colspan="8"	<td> </td>	
<td	rowspan="9"	colspan="9"	<td></td>	
>	rowspan="10"	rowspan="10"	<td> </td>	

Figure 2: 39 classes used in table structure MASTER.

To structure an HTML sequence, we need to define an Alphabet for the sequence. As shown in the left of Figure 2, we define 39 class labels for the sequence prediction. For the pairs *<thead>* and *</thead>*, *<tbody>* and *</tbody>*, and *<tr>* and *</tr>*, some other control characters may appear between these pairs. Thus, we need to define one individual class for each of them. We define the maximum "colspan" and "rowspan" as 10, thus we both use 9 labels for them individually. There are two forms for *<td></td>*, empty content or non-empty content between *<td>* and *</td>*. We use one class to denote the whole of the *<td>[content]</td>*. It should be noted that using one label instead of defining two individual labels for *<td>* and *</td>* can largely reduce the length of the sequence. For the form of *<td></td>* with empty content, we can find 11 special forms. As shown in the right of Figure 2, each form is represented by a

special class label. According to the above description, the sequence lengths of 99.6% HTML in the PubTabNet data set are less than 500.

For the sequence prediction, we use the standard cross-entropy loss. For the box regression, we employ the L1 loss to regress the coordinates of [x,y,w,h]. The coordinates are normalized to [0,1]. For the box regression head, we use an *Sigmoid* activation function before the loss calculation.

Figure 3: Example of table structure prediction. Predicted bounding box are marked with yellow color.

In Figure 3, we show a result example of sequence prediction and box regression. We could see that the structure MASTER can predict out the box coordinates correctly.

2.2 Text Line Detection

PSENet is an efficient text detection algorithm that can be considered as an instance segmentation network. It has two advantages. Firstly, PSENet, as a segmentation-based method, is able to localize texts of arbitrary shape. Secondly, the model proposes a Progressive Scale Expansion Network which can successfully identify adjacent text instances. PSENet not only adapts to text detection at arbitrary angles but also works better for adjacent text segmentation.

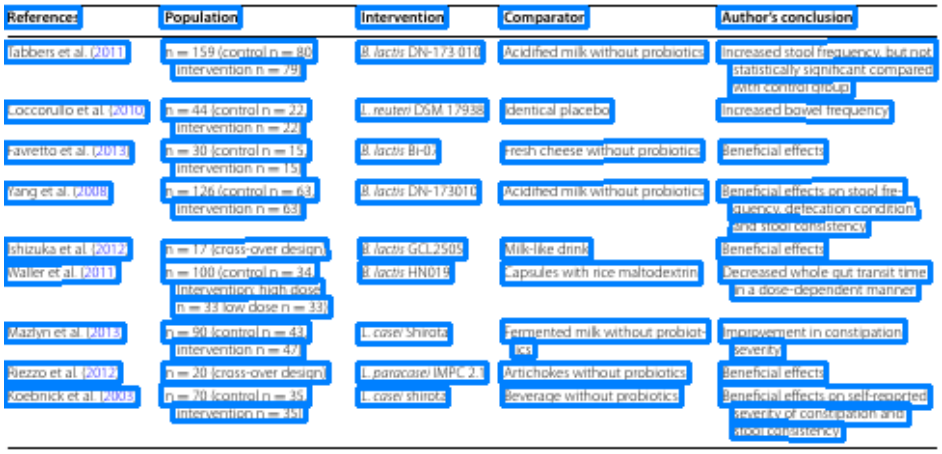
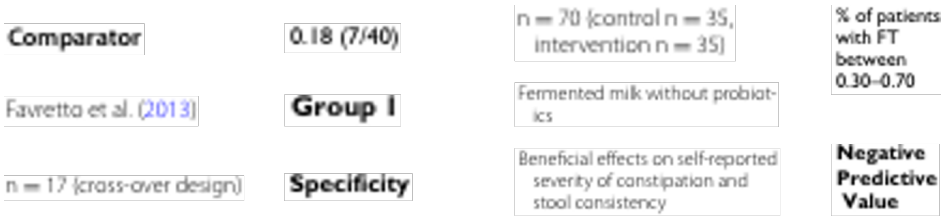


Figure 4: Visualization of text line detection.

Text detection in print documents is an easy task compared to text detection in a natural scene. In training PSENet, there are three key points needing attention, the input image size, the minimum area and the minimum kernel size. To avoid true negative, especially some small region (such as a dash line), the resolution of the input image should be large, and the minimum area size should be set to be small. In Figure 4, we visualize an detection result by PSENet.

2.3 Text Line Recognition

We also use MASTER as our text line recognition algorithm. MASTER is powerful and can be freely adapted to different tasks according to different data forms, e.g. curved text prediction, multi-line text prediction, vertical text prediction, multilingual text prediction.



(a) Single text line training samples (b) Multiply text lines training samples

Figure 5: Example of text line images cropped from training data of PUbTabNet data set; (a) single line text image; (b) multi-lines text image

The position annotations in the PubTabNet dataset (v2.0.0) is cell-level, cropped text images according to the position annotation in the data set contains both single-line and multi-line text images. We construct a text line recognition database according to position information provided in the annotation file. This text line recognition database contains about 32 million samples cropped from 500k training images. We split out 20k text line images as a validation set for checkpoint selection. Some training samples are shown in Figure 5. We can see that some texts are blur, and some are black and some are grey. The maximum sequence length is set to be 100 in our MASTER OCR. Text lines longer than 100 characters will be discarded. Some training samples are shown in Figure 5.

It should be noted that in training stage, our algorithm is trained on a database mixed with single-line text images and multi-line text images, but in the test stage, only single-line text images are inputted. By text line recognition, we can get the corresponding text content of text line images. These text contents will be merged to non-empty *<td></td>* items in the HTML sequence. The details of text content merge will also be discussed in the next subsection.

2.4 Box Assignment

According to the above three subsections, we have obtained the table structure together with the box of each cell, and the box of each text line together with its corresponding text content. To generate the complete HTML sequence, we need to assign each box of text line into its corresponding table structure cell. In this subsection, we will introduce our used match rules in detail. There are three matching rules used in our method, which we call *Center Point Rule*, *IOU Rule* and *Distance Rule*. The details will be discussed below.

2.4.1 Center Point Rule

In this matching rule, we firstly calculate central coordinate of each box obtained by PSENet. If the coordinate is in the rectangular region of the regressed box obtained by structure prediction, we call them a matching pair. The content of the text line will be filled into *<td></td>*. It is important to note that one table structure cell can be associated with several PSENet boxes because of one table structure cell may have multiple text lines.

2.4.2 IOU Rule

If the above rule is not satisfied, we will compute the IOU between the box of the chosen text line and all structure cell boxes. The box cell with the maximum IOU value will be selected. The text content will be filled into the chosen structure cell.

2.4.3 Distance Rule

Finally, if both above rules are unsuccessful. We will calculate the Euclidean distances between between the box of the chosen text line and all structure cell boxes. Similar to the *IOU Rule*, the structure cell with minimum Euclidean distance will be chosen.

2.4.4 Matching Pipeline

All above-mentioned three rules will be applied in order. Firstly, most boxes detected by PSENet will be assigned to their corresponding structure cells by *center point rule*. Owing to prediction deviations of structure prediction, a few central points of PSENet boxes are out of the rectangle region of structure cell boxes obtained by structure prediction. Secondly, some unmatched PSENet boxes under the *center point rule* will get matched under the *IOU Rule*. In the above two steps, we use the PSENet boxes to match their corresponding structure item. If there are some structure items that are not matched. In this way, we use the structure item to find the left PSENet boxes. To do this, the *distance rule* is applied.

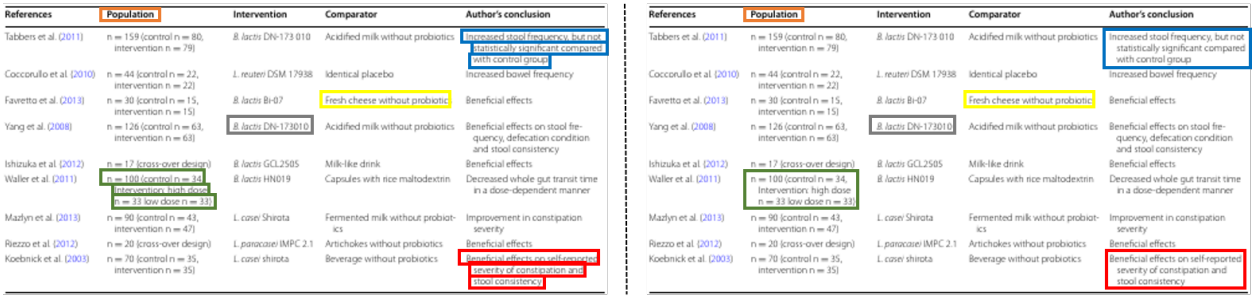


Figure 6: Example of box assignment visualization. On the left side, some detected boxes by PSENet are marked by different colors. On the right side, the boxes generated by structure prediction are marked.

A visualization example of matching results is shown in Figure 6. For aesthetic effect, we only show part of the boxes. On the left side of Figure 6, some detected boxes by PSENet are marked by different colors. On the right side of Figure 6, the boxes generated by structure prediction are marked. The boxes on the left side will be assigned to the box cell with the same color.

3 Experiment

In this section, we will describe the implementation of our table recognition system in detail.

Dataset. Our used data is the PubTabNet dataset (v2.0.0), which contains 500,777 training data and 9,115 validation data in development Phase, 9,138 samples for the development stage, and 9,064 samples for the final evaluation stage. Except for the provide training data, no extra data is used for training. To get text-line level annotation of all text boxes, 2k images of training data are relabeled for PSENet training. Actually, we only need to adjust the annotations of multi-line annotation into single-line box annotation.

Implementation Details. In PSENet training, 8 Tesla V100 GPUs are used with the batch size 10 in each GPU. The input image is resized equally, keeping the long side with resolution 1280. RandomFilp and RandomCrop are used for data augmentation. A 640×640 region is cropped from each image. Adam optimizer is applied, and the initial learning rate is 0.001 with step learning rate decay.

In table structure training, 8 Tesla V100 GPUs are used with the batch size 6 in each GPU. The input image size is 480×480, and the maximum sequence length is 500. Synchronized BN [4] and Ranger optimizer [5] are apply in this experiment, and the initial learning rate of optimizer is 0.001 with step learning rate decay.

In the training of text line recognition, 8 Tesla V100 GPUs are used with the batch size 64 in each GPU. The input size is 256×48, and the maximum length is 100. Synchronized BN and Ranger optimizer are also applied and the hyper-parameter setting is the same as the table structure training.

All models are trained based on our own FastOCR toolbox.

3.1 Ablation Studies

Our table recognition system is described above. We have conducted many attempts in this competition. In this subsection, we will discuss some useful tricks, but ignore some unsuccessful attempts.

Ranger is a synergistic optimizer combining RAdam (Rectified Adam) [6], LookAhead [7], and GC (gradient centralization) [8]. We observe that Ranger optimizer shows a better performance than Adam in this competition, and it is

applied in both table structure prediction and text line recognition. We use default Ranger. Result comparison between Adam and Ranger is shown in Table 1(a).

Synchronized Batch Normalization (SyncBN) is an effective batch normalization approach that is suitable for multi-GPU or distributed training. In standard batch normalization, the data is only normalized within the data on each GPU device. But SyncBN normalizes the input within the whole mini-batch. SyncBN is ideal for situations where the batch size is relatively small on each GPU graphics card. SyncBN is applied in our experiment.

Feature Concatenation of Layers in Transformer Decoder. In structure MASTER and text recognition MASTER, three successive transformer layers [1] is used as decoder. Different from the original MASTER, we concatenate the outputs of each transformer layer [9] and then apply a linear projection on the concatenated feature.

Label Encoding in Structure Prediction After we inspect on the training data of the PubtabNet data set(v2.0.0), we find some ambiguous annotations about empty table cell. Some empty cells of table are labeled as *<td></td>*, whereas the others are labeled as *<td> </td>* in which one space character is inserted. However, these two different table cells look the same visually. According to statistics, the ratio between *<td></td>* and *<td> </td>* is around 4:1. In our experiment, we encode these two different cells into different tokens. Our motivation is to let the model to discover the intrinsic visual features by training.

Optimizer	Structure prediction Acc.
Adam	0.7784
Ranger	**0.7826**

(a) Comparison of optimizer.

Feature Concatenation	Text line recognition Acc.
No	0.9313
Yes	**0.9347**

(b) Comparison of with or without feature concatenation.

SyncBN	FC	Structure prediction Acc.
		0.7734
✓		0.7750
✓	✓	**0.7785**

(c) Evaluation of label encoding, SyncBN and feature concatenation.

Table 1: Evaluation of different tricks on table recognition task. (a). comparison of Ranger and Adam. (b). comparison of with or without feature concatenation. (c). evaluation of label encoding.

In this competition, we have conducted some evaluations and recorded the results. The results are shown in Table 1.

According to Table 1, we have the following observations,

- Ranger optimizer has outperformed Adam optimizer consistently. Similar observation is also found in our another report [10] about ICDAR 2021 Competition on Scientific Table Image Recognition to LaTeX [11]. In our evaluation on standard benchmarks, we also find that Ranger can improve the average accuracy by around 1%.
- SyncBN can improve the performance a little. We also observe that SyncBN also shows better performance than standard BN on ICDAR 2021 competition on Mathematical Formula Detection.
- Feature concatenation can improve the accuracy of the structure prediction on this task. It should be noted that in [10], we do not observe performance improvement.

TLD	TSR			TLR	BA	ME	ForC	TEDS
PSE	ESB	SyncBN	FeaC	FeaC	Extra Insert			
✓								0.9385
✓	✓	✓	✓		✓			0.9621
✓	✓	✓	✓	✓	✓			0.9626
✓	✓	✓	✓	✓	✓	✓		0.9635
✓	✓	✓	✓	✓	✓	✓	✓	**0.9684**

Table 2: End-to-end evaluation on the validation set with TEDS as the indicator. TLD: text line detection; TSR: table structure recognition; TLR: text line recognition; ME: model ensemble. ESB: empty space box encode; SyncBN: synchronized BN; FeaC: feature concatenate output of transformer layers. ForC: format correction.

3.2 End-to-end Evaluation on the Validation Set

We generate the final HTML code by merging structure prediction, text line detection, text line recognition, and box assignment. We evaluate some tricks in these stages. Results are shown in Table 2. TEDS is used as our indicator.

We have some overall conclusions from this competition,

- ESB (empty space box encode) is important for the final TEDS indicator.
- FeaC (feature concatenation) is effective for both table structure recognition and text line recognition.
- ME (model ensemble) improves the performance a little bit. Three model ensembles in the TSR can improve the end-to-end TEDS score for around 0.2%. Three model ensembles in the text line recognition can only improve the TEDS score for around 0.03%. We only use one PSENet model.
- SyncBN is effective for both TSR and TLR.
- ForC (format correction) helps the final indicator. Our format correction is to promise all content between <*thead*> and </*thead*> is black font.

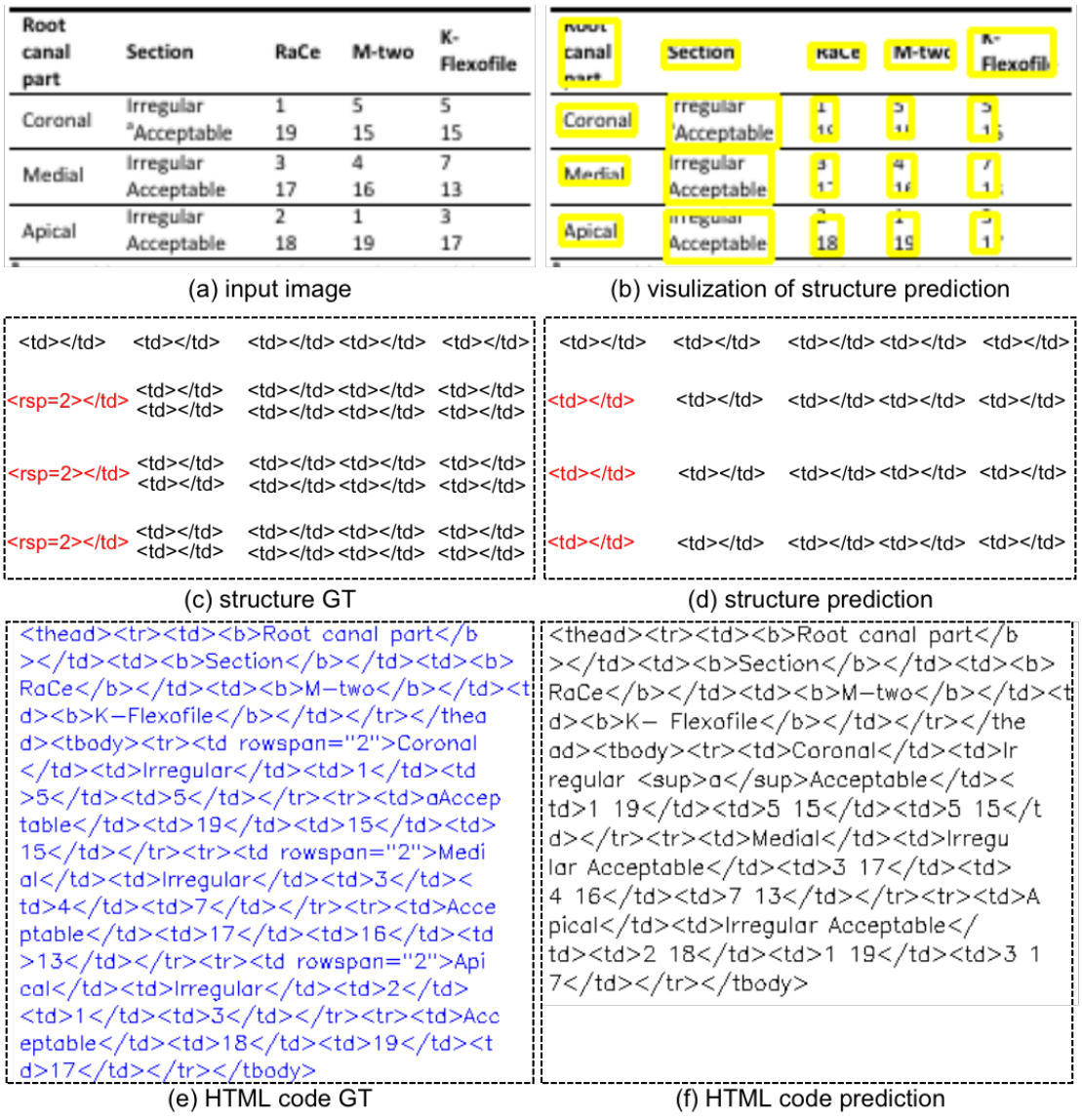


Figure 7: An example of wrong table structure prediction.

Discussion. From this competition, we have some reflections. For the end-to-end table recognition to the HTML code, structure prediction is an extremely important stage, especially for the TEDS indicator. As shown in Figure 7, although all text line information is correctly recognized. Our method obtains very low TEDS (0.423) due to wrong structure prediction. Although the provided data set is large, we still believe larger scale of data that cover more templates may further improve the structure prediction. Secondly, text line detection and text line recognition are easy tasks considering all table images are print. Thirdly, There are some labeling inconsistency issues, such as *<td></td>* and *<td> </td>*. Finally, the box assignment sub-task can be conducted by Graph Neural Network (GNN) [12] instead of hand-crafted rules.

4 Conclusion

In this paper, we present our solution for the ICDAR 2021 competition on Scientific Literature Parsing task B: table recognition to HTML. We divide the table recognition system into four sub-tasks, table structure prediction, text line detection, text line recognition, and box assignment. Our system gets a 96.84 TEDS scores on the validation data set in the development phase, and gets a 96.324 TEDS score in the final evaluation phase.

References

[1] Ning Lu, Wenwen Yu, Xianbiao Qi, Yihao Chen, Ping Gong, Rong Xiao, and Xiang Bai. Master: Multi-aspect non-local network for scene text recognition. *Pattern Recognition*, 2021.

[2] Wenhai Wang, Enze Xie, Xiang Li, Wenbo Hou, Tong Lu, Gang Yu, and Shuai Shao. Shape robust text detection with progressive scale expansion network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9336–9345, 2019.

[3] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-based table recognition: data, model, and evaluation. *arXiv preprint arXiv:1911.10683*, 2019.

[4] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7151–7160, 2018.

[5] Less Wright. Ranger-Deep-Learning-Optimizer, 2019.

[6] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*, April 2020.

[7] Michael R Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. *arXiv preprint arXiv:1907.08610*, 2019.

[8] Hongwei Yong, Jianqiang Huang, Xiansheng Hua, and Lei Zhang. Gradient centralization: A new optimization technique for deep neural networks. In *European Conference on Computer Vision*, pages 635–652. Springer, 2020.

[9] Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. Exploiting deep representations for neural machine translation. *arXiv preprint arXiv:1810.10181*, 2018.

[10] Yelin He, Xianbiao Qi, Jiaquan Ye, Peng Gao, Yihao Chen, Bingcong Li, Xin Tang, and Rong Xiao. Pingan-vcgroup's solution for icdar 2021 competition on scientific table image recognition to latex. *arXiv*, 2021.

[11] Pratik Kayal, Mrinal Anand, Harsh Desai, and Mayank Singh. Icdar 2021 competition on scientific table image recognition to latex. In *2021 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2021.

[12] Yihao Chen, Xin Tang, Xianbiao Qi, Chun-Guang Li, and Rong Xiao. Learning graph normalization for graph neural networks. *arXiv preprint arXiv:2009.11746*, 2020.