MT-YOLOv5: Mobile terminal table detection model based on YOLOv5

Zixin Ning, Xinjiao Wu, Jing Yang1 and Yanqin Yang

East China Normal University, Shanghai, China

1Corresponding author’s e-mail: jyang@cs.ecnu.edu.cn

Abstract. Table detection is an important task of optical character recognition (OCR). At present, table detection for desktop applications has basically reached commercial requirements. With the advancement of informatization, personal demand for table detection has gradually increased. There is an urgent need to establish a table detection method that can be deployed on handheld devices. This paper proposes a mobile terminal table detection model based on YOLOv5. First, we used YOLOv5 as the main framework of the model. However, considering the problem of connection redundancy in the backbone of YOLOv5, on the basis of retaining the YOLOv5 multi-scale detection head, we replaced the backbone of YOLOv5 with the same excellent Mobilenetv2. In addition, considering the non-linear defects of the lightweight model, we use deformable convolution to make up for it. This paper has been evaluated on the ICDAR 2019 dataset, and the results show that compared with the baseline model, the model reduces the number of parameters by half and increases the detection speed by 47%. At the same time, the model can reach 35.25 FPS on ordinary Android phones.

1. Introduction

Table Detection is to process the image and locate the regions of the table. With the rapid development of the economy, structured data information also presents explosive growth. As a very important way to organize structured data, tables have been widely used in common scenarios such as corporate bills, financial documents and government documents because of simplicity and visualization [1-3]. Table detection is very popular. And it can be used to facilitate subsequent OCR tasks. Now many table detection and recognition tasks have achieved excellent results. The results are similar to the manual detection results on many datasets. For example, the Decnt model proposed by Siddiqui et al. [4] is 99% accuracy on the ICDAR 2013 competition dataset [5]. And the YOLOV3 proposed by He et al. [6] is 97% accuracy on the ICDAR 2017 competition dataset [7]. But these methods need to be deployed with the help of a server. To reduce the computational complexity and the consumption of computer hardware resources, and to facilitate users, it is of certain significance to study a fast table detection method that works on the handheld mobile terminal.

At present, the table processing methods working on the mobile terminal does not build a table detection model, but only recognize the table data. The process is shown in Figure 1. In these methods, users must manually clip the table, and then upload it to the server to detect the contents of the table. This makes it very inconvenient to use. Therefore, this paper proposes a lightweight table detection method on the mobile terminal to solve this problem.

This paper thinks that the table detection model on the desktop side has reached state-of-the-arts. However, these state-of-the-arts models cannot be directly applied to the mobile terminal due to
insufficient computing power and memory constraints. Therefore, this paper expects to explore an effective model lightweight method with the help of the idea of the lightweight model, so that these state-of-the-arts table detection models can play a greater role in mobile devices.

Figure 1. Mobile terminal table detection process.

As far as known, YOLOv5 [8] is currently the most widely used detection model. YOLOv5 has reached a milestone in detection tasks. It is widely used in target detection. However, "Condensenet" written by Huang et al. [9] demonstrated that dense connectivity introduces redundancies in the DenseNet section, which is the backbone of YOLOv5. This problem is particularly prominent when YOLOv5 is used as a lightweight model. Replacing DenseNet with a more efficient network is a better method. Based on extensive literature review and experiments, we find that Mobilenetv2 [10] is a better alternative. Mobilenetv2 applies many depthwise separable convolutions to avoid invalid links in the network and ensure network performance. Besides, improving the network backbone with Mobilenetv2 can increase its adaptability on the mobile terminal. Therefore, this paper proposes a solution that combines the overall framework of YOLOv5 with the backbone network of Mobilenetv2.

Otherwise, in the lightweight model, if the nonlinear fitting ability of the model is poor, the overall performance of the model will be affected. To make up for this defect and improve the nonlinear fitting ability, we introduce deformable convolution [11] into the multi-scale detection head of the model, hoping to improve the nonlinear fitting ability of the model. To sum up, this paper proposes a lightweight table detection method combining the overall framework of YOLOv5 with the backbone network of Mobilenetv2 and deformable convolution. This paper presents table detection on the mobile terminal base on YOLOv5 for the first time (MT-YOLOv5).

Contributions to this article are as follows:
1. As far as we know, MT-YOLOv5 is the first table detection model for edge devices proposed by us, which has achieved a good balance between response speed and model accuracy.
2. We replaced the backbone of YOLOv5 with the same excellent Mobilenetv2 to reduce the number of parameters and optimize the invalid connections, making the model easier to deploy on the mobile terminal.
3. To improve the nonlinear fitting ability of the lightweight model, we add deformable convolution into the multi-scale detection head of the model.

2. Related Work
The traditional table detection method usually uses the features of the table itself (such as line segment, texture, separator, etc.) to analyze the position of the table.

Thomas et al. [12] proposed a T-RECS system for identifying table structures in electronic or paper documents. First, a random word is selected as a seed and the word's block is recursively expanded to all vertically adjacent neighbour blocks. Some post-processing methods have also been added to handle isolated words or other errors. Cesarini et al. [1] proposed a method based on MXY trees to determine the existence of tables by searching parallel lines. Besides, located tables can be merged based on
proximity and similarity criteria. Yildiz [3] proposed pdf2table to analyze text blocks in PDF documents and identify table regions according to the rules between these text blocks. But this approach is susceptible to header or footer noise and can lead to misjudgments. Hao et al. [13] obtained candidate regions by roughly screening location regions based on rules, and then determined whether these regions were tables by using CNN or CRF. Different from the above methods, Jing Fang et al. [14] used document layout analysis to detect table position. They proposed a method based on the visual separator and geometric content layout information to analyze page whitespace to obtain the number of page columns, and then detect table position through graphic lines and separator, etc. These approaches are usually rules-based and have an advantage in speed, but they all rely heavily on specific datasets and lack generalization performance.

With the development of deep learning, the deep learning model based on the convolutional neural network has gradually returned to people's vision, generating many classic target detection models, such as Faster RCNN [15], YOLO [16], and so on. Some workers combine the characteristics of tables with these excellent target detection models and propose a table detection model based on deep learning [17-19]. These models have the best performance on many classic datasets.

Combined with the Faster RCNN model, Gilani et al. [17] applied deep learning to the field of table detection for the first time. They proposed that the table image should be firstly linked and analyzed through pre-processing, and the processed image should be put into the model for preliminary detection, and then the results should be optimized through a fixed post-processing method. Due to the variety of table styles, the algorithm is not accurate enough. On this basis, Sun et al. [18] proposed the corner point-based Faster RCNN model. They further refined the features of the table and proposed to locate the position of the table. The four corners of the table are located through another branch to further improve the accuracy of positioning. Their method can still achieve high accuracy even when IOU is large. While the corner approach can help with positioning accuracy, it's difficult to visualize the corner in a wide variety of formats. For example, there are no corners on some open tables. Besides, Huang et al. [6] proposed a tabular detection algorithm based on Yolov3. They proposed a new Anchor optimization strategy and three post-processing methods. Using these methods, Huang et al. achieved the best performance to date on the ICDAR 2017 dataset. However, because the specific post-processing method is only effective for a certain data set, their method has a lack of performance on the ICDAR 2013 dataset. Recently, Prasad et al. [19] explored a method to extract required knowledge from a small amount of data based on transfer learning based on Mask RCNN. In addition to directly utilizing the intuitive style features of the table and the deep learning model, some workers also directly study how to utilize the advantages of the deep model itself to optimize the detection ability of various style sheets. Besides, researchers have proposed some novel algorithms. Riba et al. [20], combined with a graph neural network, proposed a model to detect repeated join information in tables. The method does not consider fixed table rows and columns, but captures visual continuity in horizontal or vertical directions and determines the position of the table. But this method can only achieve good results on some specific report data sets.

The above-mentioned methods based on deep learning all need to use a huge amount of parameters, and are lacking in real-time reasoning speed.

3. Methods

3.1. Model framework

The model architecture proposed in this paper can be seen in Figure 2. The left side is the backbone network composed of multiple bottleneck structures, and the right side is the multi-scale detection head.

The input of the model is a three-channel image, ‘conv’ represents the standard convolution operation, and ‘bottleneck’ represents the bottleneck structure in Mobilenetv2, ‘CSP’ represents the CSP structure proposed by Wang et al. [21], ‘upsampling’ represents the up-sampling operation, ‘cat’ in the figure represents splicing by channel. The output bounding box is a multi-group prediction result (x, y, w, h, conf).
This paper presents a new table detection model framework. This framework uses the Mobilebetv2 backbone network, combined with YOLOv5’s multi-scale detection head. Specifically, we first use a convolutional layer with a convolution kernel of 3, then a bottleneck layer for different channels, and finally a detection head with a CSP structure. As shown in Figure 2, ‘(32,3)’ means that the output is a 32-channel bottleneck layer, and the number of repetitions is 3. The number of parameters in this model is only 3.6million. The overall workflow of our work is shown in Figure 3.

3.2. Deformable convolutional
The table usually presents a form with a larger width and a smaller height. Using a neural network to capture such morphological features will be conducive to the sensitivity of the model.

Traditional convolutional networks usually use convolutional kernels of size 3×3 or 5×5, but they are not suitable for objects such as tables, which usually appear as rectangles. The traditional convolution operation is defined as follows:

\[
(F, I)_{(i,j)} = \sum_{m=-k}^{K} \sum_{n=-k}^{K} F(m,n) \times I(i-m,j-n)
\]

\[i = i_1...H, j = j_1...W\]  

(1)
In the convolutional layer of the network, the convolutional kernel shape of the connection between all levels is the same. This is problematic in-depth models, where different tables can appear in arbitrary shapes and sizes. Accurate perception of these tables requires the ability to dynamically adjust the receptive fields of the convolutional layer. Therefore, we use a deformed CNN[11] to replace the traditional CNN in the CSP part of the MT-YOLOv5 model. Figure 3 shows the basic process of deformable convolution.

Its mathematical representation is as follows:

\[
(F, I)_{(i, j)} = \sum_{m=-K}^{K} \sum_{n=-K}^{K} F(m, n) \times I(i - m + \delta_{i, j, m, n}^{\text{vertical}}, j - n + \delta_{i, j, m, n}^{\text{horizontal}})
\]

(2)

\(\delta_{i, j, m, n}^{\text{vertical}}\) represents the vertical offset learned by the model, and \(\delta_{i, j, m, n}^{\text{horizontal}}\) represents the horizontal offset learned by the model. Specifically, in addition to the basic convolution operation, each deformable convolution layer needs to learn an additional offset. In the upper part of Figure 4, it can be seen that the original convolution kernel will be offset in different directions. In actual training, it will automatically fit according to the shape of the table, which greatly improves the perception of the table. In actual use, this article adds deformable convolution to the model's CSP module to replace the original convolution layer.

4. Experiments

4.1. dataset

We evaluated our method on ICDAR 2019 Competition on Table Detection and Recognition (cTDaR) [22]. ICDAR 2019 cTDaR contains 1200 training data and 439 test data. The dataset consists of handwritten documents, scanned documents, and some financial statements.

The ICDAR 2019 cTDaR datasets are in two styles: Modern and Archival. ‘Modern’ is a modern document style, and ‘Archival’ is a handwritten document style. Figure 5 shows both styles of documents. After unifying the pictures in different formats into JPG, we corrected some negative label information and converted it into a VOC competition data format. We used 1200 pieces of data as training samples and tested on 439 pieces of data.
4.2. Evaluation index of experimental details

Our experiments were conducted in a pytorch 1.7.0 environment with GTX TitanXP training equipment. The initial learning rate is $4 \times 10^{-4}$, the learning rate was reduced to $5 \times 10^{-5}$ in the 100th iteration. The model trained a total of 500 arguments.

The classic evaluation index of Intersection Over Union (IOU) was adopted in all the three datasets. We define the loss function as follows:

$$IOU = \frac{S_i \cap S_j}{S_i \cup S_j}$$

(3)

$I_S$ represents the table area predicted by the model, and $J_S$ represents the ground truth. The sample with IOU greater than a certain value is denoted as positive sample true positive (TP) sample. Samples whose IOU is less than a certain value are labeled as negative sample false positive (FP) samples. Follows many evaluation methods, we use precision rate (P), recall rate (R) and F1 value evaluate our models.

$$P = \frac{TP}{TP + FP}$$

(4)

$$R = \frac{TP}{TP + FN}$$

(5)

$$F1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

(6)

In addition, FPS(Frames Per Second) represents the number of responding per second of the model.

4.3. Experimental results and analysis

Table 1 is the evaluation result of our method on the ICDAR2019 competition dataset. The IOU value we adopted was 0.8. In the table, ‘Pre-process’ represents the pre-processing of input images, and ‘Post-process’ represents the post-processing of model output results. ‘Params’ represents the number of model parameters in millions(M). As can be seen from Table 1, compared with the baseline model, the MT-YOLOv5 reduces the number of parameters by nearly half but loses a little precision. Compared with other models with the same number of parameters, MT-YOLOv5 has obvious advantages.

Besides, the model proposed in this paper only uses 3.6M of parameters. This is 1/40 of the number of participants used by the ICDAR2019 competition team. The results show that the method presented in this paper is similar to, or even better than these competition methods.

In conclusion, we have a distinct advantage over lightweight models of the same level, and we have a distinct advantage over traditional methods in terms of speed.

We evaluate the responding speed of our model and the baseline model on different devices. Table 2 shows the experimental results. As can be seen from Table 2, our method improves the reasoning speed.
by 47% compared to the baseline method. Our method is three times faster than the Faster RCNN model commonly used by ICDAR 2019 teams.

**Table 1.** Comparison of ICDAR 2019 Competition dataset. MT-YOLOv5(no-dc) means that MT-YOLOv5 does not use deformable convolution.

| Method          | P  | R  | F1  | Pre-process | Post-process | Params(M) |
|-----------------|----|----|-----|-------------|--------------|-----------|
| TableRadar      | 0.95 | 0.94 | 0.94 | ✓ | - | >240 |
| NLPR-PAL        | 0.93 | 0.93 | 0.93 | ✓ | - | - |
| Lenovo Ocean    | 0.88 | 0.86 | 0.87 | - | ✓ | 220 |
| ABC Fintech     | 0.84 | 0.75 | 0.79 | - | - | - |
| Applica-robots  | 0.82 | 0.82 | 0.82 | ✓ | ✓ | 138 |
| TJNU202-2       | 0.86 | 0.81 | 0.83 | - | ✓ | 138 |
| Table Fan       | 0.81 | 0.79 | 0.80 | ✓ | - | 240 |
| YOLOv5          | 0.84 | 0.92 | 0.88 | - | - | 7 |
| MobileNetV2 + ssd | 0.83 | 0.85 | 0.84 | - | - | 3.2 |
| MT-YOLOv5(no-dc)| 0.79 | 0.88 | 0.83 | - | - | 3.5 |
| MT-YOLOv5       | 0.81 | 0.91 | 0.86 | - | - | 3.6 |

**Table 2.** Responding speed comparison.

| Method          | Platform | Device         | Image-size | FPS |
|-----------------|----------|----------------|------------|-----|
| Faster RCNN     | Desktop  | TITAN XP       | 640        | 20.1|
| YOLOv5          | Desktop  | TITAN XP       | 640        | 56.1|
| MT-YOLOv5       | Desktop  | TITAN XP       | 640        | 82.6|
| YOLOv5          | mobile   | Snapdragon 855 | 128        | 24.2|
| MT-YOLOv5       | mobile   | Snapdragon 855 | 128        | 36.5|

We conduct responding speed experiments on commonly used mobile devices and desktop devices. The experimental results can be seen in Table 3. 439 test images from the ICDAR2019 dataset are used as samples. We resize the images to 320 pixels and 128 pixels. The responding speed of all the images is averaged to get the final FPS.

Table 3 shows that the responding speed of the MT-YOLOv5 model is faster than 24FPS in the mobile terminal.

This also means that real-time detection can be achieved. The model in this paper can omit the manual clipping step. On the mobile terminal, users don’t need to manually clip out the table region of the image. They can extract the table directly with the model and occasionally fine-tune the table. This will speed up detection. Figure 6 shows some table detection examples of our model.

**Table 3.** MT-YOLOv5 responding results on multiple devices.

| Platform | Device | Device type     | Image-size | FPS |
|----------|--------|----------------|------------|-----|
| CPU      | PC     | E5-2686 v4     | 320        | 25.65|
| CPU      | PC     | E5-2628 v4     | 128        | 41.52|
| CPU      | PC     | R5-3600        | 320        | 25.66|
| CPU      | PC     | R5-3600        | 128        | 44.61|
| CPU      | mobile phone | Snapdragon 855 | 320       | 16.8 |
| CPU      | mobile phone | Snapdragon 855 | 128       | 39.9 |
| CPU      | mobile phone | Huawei Kirin 980 | 320  | 18.12|
| CPU      | mobile phone | Huawei Kirin 980 | 128 | 35.25|
| GPU      | PC     | TITAN XP       | 320        | 87.5 |
| GPU      | PC     | TITAN XP       | 128        | 90.4 |


5. Conclusion
In this paper, we propose a mobile terminal table detection model named MT-YOLOv5 based on YOLOv5, which can detect the table position in real-time on the mobile terminal. MT-YOLOv5 replacing the backbone network of YOLOv5 with MobileNetv2. Besides, we have added a deformable convolution module to enhance its nonlinear perception ability. As far as we know, this is the first work to table detect on the mobile terminal. The experimental results show that MT-YOLOV5 can achieve almost the same performance as the original model with less than half of the number of parameters.

References
[1] Cesarini F, Marinai S, Sarti L, et al. 2002 Trainable table location in document images[C]//Object recognition supported by user interaction for service robots. IEEE, 3: 236-240.
[2] Gatos B, Danatsas D, Pratikakis I, et al. 2005 Automatic table detection in document images[C]//International Conference on Pattern Recognition and Image Analysis. Springer, Berlin, Heidelberg, 2005: 609-618.
[3] Yildiz B, Kaiser K, Miksch S. 2005 pdf2table: A method to extract table information from pdf files[C]//IJCAI. 2005: 1773-1785.
[4] Siddiqui S A, Malik M I, Agne S, et al. 2018 Decnt: Deep deformable cnn for table detection[J]. IEEE Access, 6: 74151-74161.
[5] Göbel M, Hassan T, Oro E, et al. 2013 ICDAR 2013 table competition[C]//2013 12th International Conference on Document Analysis and Recognition. IEEE, 2013: 1449-1453.
[6] Huang Y, Yan Q, Li Y, et al. 2019 A YOLO-based table detection method[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 813-818.
[7] Gao L, Xi Y, Jiang Z, et al. 2017 ICDAR2017 competition on page object detection[C]//2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). IEEE, 1: 1417-1422.
[8] Glenn Jocher, Alex Stoken and Jirka Borovec 2020 Yolov5[Z]. Github, 2020-05, https://github.com/ultralytics/yolov5.
[9] Huang G, Liu S, Van der Maaten L, et al. 2018 Condensenet: An efficient densenet using learned group convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 2752-2761.
[10] Sandler M, Howard A, Zhu M, et al. 2018 Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4510-4520.
[11] Dai J, Qi H, Xiong Y, et al. 2017 Deformable convolutional networks[C]//Proceedings of the IEEE international conference on computer vision. 2017: 764-773.
[12] Kieninger T G 1998 Table structure recognition based on robust block
[13] Hao L, Gao L, Yi X, et al. 2016 A table detection method for pdf documents based on convolutional neural networks[C]//2016 12th IAPR Workshop on Document Analysis Systems (DAS). IEEE, 2016: 287-292.

[14] Fang J, Gao L, Bai K, et al. 2011 A table detection method for multipage pdf documents via visual separators and tabular structures[C]//2011 International Conference on Document Analysis and Recognition. IEEE, 2011: 779-783.

[15] Ren S, He K, Girshick R, et al. 2016 Faster R-CNN: towards real-time object detection with region proposal networks[J]. IEEE transactions on pattern analysis and machine intelligence, 39(6): 1137-1149.

[16] Redmon J, Divvala S, Girshick R, et al. 2016 You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

[17] Gilani A, Qasim S R, Malik I, et al. 2017 Table detection using deep learning[C]//2017 14th IAPR international conference on document analysis and recognition (ICDAR). IEEE, 1: 771-776.

[18] Sun N, Zhu Y and Hu X 2019 Faster R-CNN based table detection combining corner locating[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 1314-1319.

[19] Prasad D, Gadpal A, Kapadni K, et al. 2020 CascadeTabNet: An approach for end to end table detection and structure recognition from image-based documents[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020: 572-573.

[20] Riba P, Dutta A, Goldmann L, et al. 2019 Table detection in invoice documents by graph neural networks[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 122-127.

[21] Wang C Y, Liao H Y M, Wu Y H, et al. 2020 CSPNet: A new backbone that can enhance learning capability of CNN[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020: 390-391.

[22] Gao L, Huang Y, Déjean H, et al. 2019 ICDAR 2019 competition on table detection and recognition (cTDaR)[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 1510-1515.