IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 19, 2021, accepted February 4, 2021, date of publication March 2, 2021, date of current version March 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3063184

A Post-Processing Method for Text Detection Based on Geometric Features

XIAOGANG QIU, SHANXIONG CHEN, RANKANG LI, DINGWANG WANG, AND XIAOYU LIN
School of Computer and Information Science, Southwest University, Chongqing 400715, China

Corresponding author: Shanxiong Chen (csxpml@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 61603310.

ABSTRACT Deep learning text detection is generally divided into two steps: prediction candidate box of depth model and post-processing, and post-processing usually uses NMS or prediction box to merge and connect. At present, many methods of text detection can detect the character position of the document image, but they are often not accurate enough, which affects the effect of subsequent recognition. To solve the problem of inaccurate detection, this article proposes a post-processing method of document detection based on geometric features from the perspective of post-processing. The post-processing method is mainly divided into four modules. Firstly, the background removal (BR) module separates the background information and character information through pixel threshold. Secondly, the candidate box expansion (CBE) module expands the prediction box in all directions by judging whether the boundary of the prediction box is in the character pixel. Then is the non-standard box removal (NBR) module, using the consistency principle of characters and surrounding characters to filter out the error detection of some prediction boxes. Finally, the module of repeating box removal (RBR) is used to remove the repeated prediction box. In order to verify the effectiveness of this method, a large number of experiments have been conducted on Standard yi, Chinese2k, English2k, ICDAR 2015 and ICDAR 2017(CTW-12k) datasets. The experimental results show that the method proposed in this article can improve the effect of text detection.

INDEX TERMS Deep learning, text detection, target box correction, post-processing.

I. INTRODUCTION

The image of document is an important carrier of information and plays an important role in daily life [1]. With the widespread application of digitization in various fields, human beings hope that machines can also imitate human's ability to read books, hence optical character recognition (OCR) technology was proposed [2]. And the text detection is an indispensable part of OCR, which is very important for subsequent text recognition. Efficient and accurate text detection has important applications in the field of document image, including the character Recognition system, multilingual translation of images, human-computer interaction, etc. The text detection is mainly to predict the character area information for the image, that is, the prediction box coordinates, which is convenient for subsequent extraction of character information for processing.

The associate editor coordinating the review of this manuscript and approving it for publication was Shiqi Wang.

According to the development of text detection technology and the classification criteria of text area description features, the text detection method can be roughly divided into two categories, traditional text detection method and text detection method based on deep learning. The traditional text detection method mainly divided into the method based on connected domain analysis [3]–[10] and the method based on sliding detection window [11]–[18]. They use hand-designed features to verify the obtained candidate regions, and finally obtain text region information, that is, the prediction box. In recent years, due to the emergence of deep learning, the performance of document detection has been significantly improved. At present, most text detection methods based on the deep learning by CNN (Convolutional Neural Networks). These methods obtain text features through deep learning and detect text based on the above features. Compared with the traditional hand-designed features used before, this type of method has achieved more excellent detection results. Among the text detection methods based on deep learning, the method based on the text region proposal [19]–[26] is used

the most extensive, followed by the method based on image segmentation [27]–[32].

Related literature research shows that the methods based on deep learning can achieve good results in text detection, but whether it is a method based on text area suggestions or a method based on image segmentation, the detection results are often insufficient in detail. These shortcomings are mainly manifested as: 1) character misrecognition (two characters are mistaken for one character, punctuation marks or other areas that are not characters are wrongly considered as characters), 2) detection deviation (one character is only half detected), 3) overlap detection (the same character is detected by multiple detection boxes). The above shortcomings seriously affect the performance of subsequent recognition. However, these details are often not solved by deep model training. It is closely related to the post-processing of text detection and requires us to optimize through post-processing methods to get better detection results. Therefore, this article proposes a text detection post-processing method for the above problems. This method can effectively solve the problems of character misrecognition, detection deviation, overlap detection, etc., and make the detection results more excellent. The method in this article is mainly divided into four modules, background removal module, candidate box expansion module, non-standard box removal module and repeating box removal module. Fig. 1 shows the detection results before and after processing using the method in this article on different datasets. The first row in the figure uses the detection results of the mainstream deep learning text detection framework, and the second row uses the post-processing method of this article. Fig. 1(a), Fig. 1(b), Fig. 1(c), Fig. 1(d), Fig. 1(e) are the text detection results on the datasets of standard Yi, Chinese2k, English2k, ICDAR 2015, ICDAR 2017(CTW-12k).

II. RELATED WORK

At present, in the text detection method based on deep learning, whether it is a method based on text area suggestions or a method based on image segmentation, we have to post-process the obtained candidate boxes after making predictions through the deep learning model. According to different methods, post-processing is generally divided into three categories: non-maximum suppression (NMS), clustering and candidate box merging. After obtaining the candidate boxes through the deep learning model, a text area often has multiple candidate boxes. NMS is to process these candidate boxes by setting a threshold, only the optimal box is retained. Merging is to process the obtained candidate boxes according to certain rules to obtain the final prediction box, which is generally suitable for text line detection. Clustering is a special method that processes the prediction results of special models. According to the author's research, such post-processing is currently only used in the LSAE.

Post-processing uses the text detection method of NMS. Liao and Shi *et al.* proposed a text detection method A Fast Text Detector with a Single Deep Neural Network (TextBoxes) at AAAI 2017. This method is improved on the basis of SSD. Taking SSD as the basic framework, an end-to-end training text detector is proposed [26]. The post-processing uses NMS, but because the training uses multi-scale training, so the NMS has been improved and is called an effective cascade NMS. Effective cascading NMS is divided into two steps: the first step is to perform NMS on the smallest bounding box, this step is fast and eliminates a large number of boxes; the second step is to perform quadrangle NMS, and use the obtained output as the prediction result. Zhou and Yao *et al.* proposed a text detection method An Efficient and Accurate Scene Text Detector (EAST) at CVPR 2017 [29]. The post-processing of this method is also NMS. EAST believes that splitting a text detection algorithm into multiple stages does not have much benefit. It is the correct move to implement a true end-to-end text detection network. Therefore, a two-stage text detection method EAST is proposed, which is composed of the FCN stage and the NMS stage. The post-processing step only includes NMS processing of the geometric prediction box.

Post-processing text detection method using clustering. Tian and Shu *et al.* proposed a text detection method Learning Shape-Aware Embedding for Scene Text Detection (LSAE) at CVPR 2019 [33]. LSAE introduces the concept of embedding distance, and the author refers to the overlapping distance between candidate boxes as embedding distance. The post-processing uses a combination of NMS and clustering. First process through NMS to obtain the processed candidate box, and then use DBSCAN to obtain the clustering results of the global feature map and the central feature map. Judge the pixels that belong to the global feature map and the outside of the central feature map. When the embedding distance of the central feature cluster is less than the threshold, the pixels are assigned to the nearest central feature cluster, and the other central clusters are continuously looped until all the central clusters are processed. Finally, the corresponding minimum bounding box is generated for each central cluster as the output to obtain the prediction box.

Post-processing uses the text detection method of merging. Detecting Text in Natural Image with Connectionist Text Proposal Network (CTPN), a text detection method proposed by Tian and Huang *et al.* at ECCV 2016 [23]. CTPN combined with CNN and LSTM deep network can effectively detect horizontally distributed text, which is currently a better text detection method. The post-processing of this method is mainly merging, which splits the task of text detection. And we only need to judge whether the obtained candidate box is part of a text. When all small text boxes in a picture are detected, the belong to the small text boxes of the same text are merged. At this time, a complete and large text box can be obtained, and the text detection task is completed. Shi and Bai published a spotlight article Detecting Oriented Text in Natural Images by Linking Segments (SegLink) at CVPR 2017. The article introduces a detection method that can detect text at any angle. We generally call this method SegLink [22]. It combines both the idea of CTPN small-scale candidate box

FIGURE 1. The result of detection. (a)Standardize Yi; (b)Chinese2K; (c)English2K; (d)ICDAR 2015; (e)ICDAR 2017(CTW-12k).

and the idea of SSD algorithm. The post-processing of this method is mainly connection, and it can also be understood as a special merge. By predicting, the word was cut into small text blocks that were easier to detect, and then the text blocks were connected into words. At that time, the effect of text detection state-of-art was achieved. Baek and Lee proposed a text detection method Character Region Awareness for Text Detection (CRAFT) [37] in CVPR 2019. CRAFT this article mainly solves the problem of text detection. The idea is to use the method of segmentation. Unlike image segmentation, CRAFT does not perform pixel-level classification of the entire image, but does regression. It has two branches, one is the probability that the target is the center of the character, and the other is the connection relationship between the characters, and then after a step of merging and processing, the bounding box of the text is obtained.

III. OUR APPROACH

At present, the deep learning text detection method is mainly to achieve accurate positioning through an excellent deep learning model. It is generally divided into two steps. The first step is to predict the candidate box according to the trained model. At this time, the candidate boxes often have many overlapping areas. Therefore, in order to find the best candidate box in the obtained candidate box, it is often necessary to perform the second step, that is, to obtain the final prediction box through post-processing to achieve accurate positioning. However, in actual applications, the final prediction box is often not very accurate, so many algorithms also perform other post-processing, such as merging adjacent candidate boxes. The current deep learning text detection methos are less for post-processing, and the obtained prediction box can generally obtain better detection results, but there are many details that are not processed in place. For example: character misrecognition, detection deviation, overlap detection, etc. In order to solve the above problems, this article proposes a text detection post-processing method.

The framework diagram of the model structure proposed in this article is shown in Fig. 2. The input of the model in this article is the original document image and label of the training set during the training process, the image to be predicted during the test process, and the output is the location information of the image to be predicted. The information here mainly includes two coordinates ((X1, Y1), (X2, Y2)), (X1, Y1) are the coordinates of the upper left corner of the prediction box, and (X2, Y2) are the coordinates of the lower right corner of the prediction box. Experiments in this article use mainstream text detection models, such as: LSAE [33], CTPN [23], SegLink [22], TextBoxes [26], EAST [29], TextBoxes++ [36], CRAFT [37], etc. We use the deep learning model to obtain the prediction boxes, and then post-process the prediction boxes separately. The post-processing is mainly divided into 4 modules. First, the surrounding background image is removed by the algorithm. Second, the algorithm adjusts the coordinates of the prediction boxes to obtain more accurate coordinates. Then we remove the non-standard prediction box. There are two main types of non-standard prediction boxes. One is the prediction box that detects punctuation marks as characters, and the other is that two characters are incorrectly detected as one character. Finally, the obtained repeated prediction box is removed to obtain the final prediction boxes.

The overall network framework can be divided into two sub-processes, one for obtaining the original prediction box, and one for post-processing the original prediction box, as is shown in Fig. 2. Among them, the original prediction box is obtained by deep learning text detection method; post-processing is divided into 4 modules, each module is set up an algorithm for processing. The orange, light blue, green, and blue detection boxes in Fig. 2 are the detection results after processing by the BR, CBE, NBR, and RBR algorithms. The post-processing methods in this article can achieve good results in most deep learning text detection methods after being individually trained on specific datasets.

A. BACKGROUND REMOVAL MODULE

The prediction box obtained by the deep learning text detection method often contains some background information. If too much background information may cause a decrease in detection precision and recall, this article proposes a BR

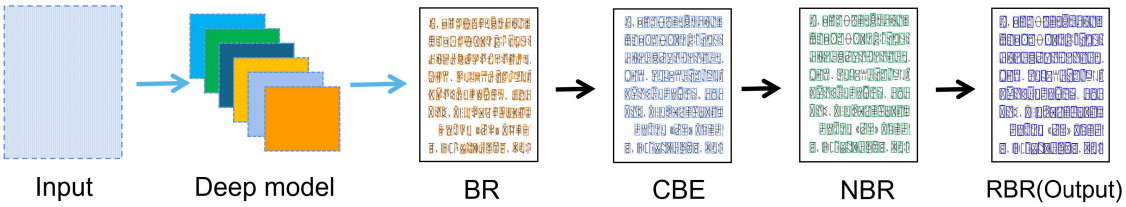


FIGURE 2. The frame diagram of model structure. The BR, CBE, NBR and RBR represent the four modules proposed in this article.

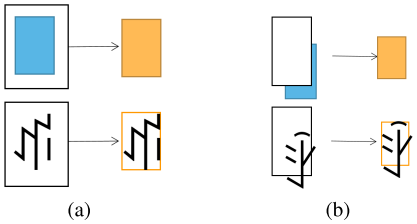


FIGURE 3. The prediction box is processed before and after by BR algorithm. (a)Characters are completely surrounded by prediction boxes; (b)The characters overlap with the prediction box.

(background removal) algorithm for this problem. We remove the excess background in the prediction box without changing the intersection area of the prediction box and the character area. This can further improve the detection effect. The effect before and after the background removal process is shown in Fig. 3.

The blue part in Fig. 3 is the area where the characters are located, the black box is the original prediction box after deep learning detection and NMS processing, and the orange box is the prediction box after being processed by the BR algorithm. The background removal algorithm is mainly to separate the character area from the background area through the pixel threshold, determine the character area according to the threshold, and then update the prediction box coordinates. Therefore, finding a suitable threshold is critical to determine the character area. The difference between the character area and the background area of each different image to be detected is also different, some are smaller, some are larger, so for the threshold, this article proposes an ATA (Adaptive threshold algorithm) algorithm. The formula for calculating the adaptive threshold is as follows:

$$T = (1 - \alpha) \times MAX\_10 + \alpha \times MIN\_10 \tag{1}$$

where α is the weight coefficient, MIN_10 represents the average value of the 10 pixels with the smallest gray value in the original detection area, MAX_10 represents the average value of the 10 pixels with the largest gray value in the original detection area, and T represents the adaptive threshold.

The weighting coefficient of the adaptive threshold algorithm here ranges from 0.0 to 1.0. Its main function is to balance the relationship between the maximum gray value and the minimum gray value, and find a suitable threshold to separate the character area and the background area. For the size setting of the weight coefficient, this article takes different values of the weight coefficient on the standard Yi, Chinese2K and English2K datasets to obtain the precision,

Algorithm 1 Adaptive Threshold Algorithm

Input: $((X_1, Y1), (X_2, Y_2)), Image$;
Output: Adaptive threshold: T.

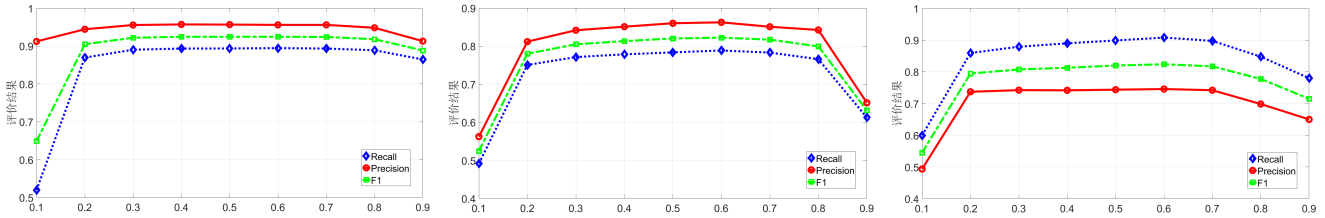
1: Grayscale Image: $Bd\_img \leftarrow Gray(Image)$;
2: **repeat**
3: Get pixels of prediction box area: $Img \leftarrow Bd\_img[X1 : X2, Y1 : Y2]$;
4: Get the largest 10 pixels in Img: $EMAX\_10 \leftarrow Get\_max(Img)$;
5: Get the smallest 10 pixels in Img: $EMIN\_10 \leftarrow Get\_min(Img)$;
6: Calculate the average of the 10 largest pixels: $MAX\_10 \leftarrow SUM(EMAX\_10)/10$;
7: Calculate the average of the 10 smallest pixels: $MIN\_10 \leftarrow SUM(EMIN\_10)/10$;
8: Initial weight coefficient: $\alpha \leftarrow 0.5$;
9: Calculate the adaptive threshold:

$$T = (1 - \alpha) \times MAX\_10 + \alpha \times MIN\_10.$$

10: **until** All coordinates are processed.

recall, and F1 value results. The results are shown in Fig. 4. As can be seen from Fig. 4, when the weight coefficient is less than 0.3 and greater than 0.7, the precision, recall, and F1 value change significantly, and when the weight coefficient is between 0.3-0.7, the precision, recall, F1 value changes tend to be smooth and the effect is better. Therefore, the weight coefficient is set to 0.5 on the standardized Yi dataset, and the other datasets are set to 0.6 in this article. Fig. 5 shows the detection results of different weight coefficients in different datasets. The first row, the second row, and the third row are the detection results on the standardized Yi, English2K, and Chinese2K. Fig. 5(a) shows the detection results before adding this method. Fig. 5(b), Fig. 5(c), Fig. 5(d) respectively represent the detection results of $\alpha = 0.3$, $\alpha = 0.5$, $\alpha = 0.7$ after adding this method.

After the pixel threshold T is determined by the adaptive threshold algorithm, the next step is to perform background removal. The prediction box coordinates are updated according to Equation (2). The Getchar function in the algorithm is mainly used to determine the relationship between character pixels and T. Its return values are True and False, which indicate that the character pixel is less than T and the character pixel is greater than T. In the Getchar function, we first assume that the pixels smaller than the threshold T belong to the character area pixels, and then we expand around by



(a) (b) (c)

FIGURE 4. The effect of weight coefficient on the result. (a)Standardize Yi; (b)English2K; (c)Chinese2K.

(a) (b) (c) (d)

FIGURE 5. The visualization results of different weight coefficients in different data sets. (a)Before processing by the method in this article; (b)$\alpha = 0.3$; (c)$\alpha = 0.5$; (d)$\alpha = 0.7$.

judging whether the edge of the prediction box contains the character area pixels, and set a counter to count the degree of expansion. Expand one pixel in any direction, and the counter will increase by one. If the assumption is correct, the counter will stop increasing to a certain extent. At this time, the Getchar function returns True. If the assumption is wrong and the background pixels are mistaken for character pixels, the prediction box will continue to expand and the counter will always increase. Therefore, we need to set a threshold for the counter. If the counter is greater than the set threshold, then the assumption is wrong, which means the pixels larger than the threshold T are the pixels that belong to the character area, and the function Getchar returns False. Here we set the threshold as the length and width of the prediction box. Algorithm 2 details the execution steps of the background removal algorithm.

$$((X_1, Y_1), (X_2, Y_2)) = \begin{cases} X_1 \leftarrow X_1 + col\_min \\ Y_1 \leftarrow Y_1 + row\_min \\ X_2 \leftarrow X_1 + col\_max \\ Y_2 \leftarrow Y_1 + row\_max \end{cases} \quad (2)$$

where row_min, row_max, col_min, col_max respectively represent the minimum index of the row, the maximum index of the row, the minimum index of the column, and the maximum index of the column in the real character area in the prediction box.

B. CANDIDATE BOX EXPANSION MODULE

After processing by the BR algorithm, we can get a more precise prediction box. At this time, the background information in the original prediction box is basically eliminated, and only the character area is included. However, the prediction box may appear to deviate from the detection box at this time. Therefore, this article proposes a candidate box expansion algorithm CBE (Candidate box expansion) for this problem. We will expand the obtained prediction box with a certain regularity, so that it can completely extract the character area. This can further improve the detection effect. The effect before and after processing by the candidate box expansion algorithm is shown in Fig. 6.

In Fig. 6(a), the blue part is the area where the characters are located, the black box is the original prediction box after deep learning detection and NMS processing, the orange is the prediction box after the BR algorithm, and the green is the prediction box after the CBE algorithm. The candidate box expansion algorithm mainly determines whether there are

(a) (b)

FIGURE 6. Before and after the prediction box is processed by BR and CBE algorithm. (a)The simulation character overlaps with the prediction box; (b)The real character overlaps with the prediction box.

Algorithm 2 Background Removal Algorithm

Input: $u((X_1, Y_1), (X_2, Y_2))$, $Image$, T;
Output: Updated $((X_1, Y1), (X_2, Y_2))$.

Grayscale Image: $Bd\_img \leftarrow Gray(Image)$;
repeat
  Get pixels of prediction box area: $Img \leftarrow Bd\_img[X1 : X2, Y1 : Y2]$;
  Calculate threshold: $T \leftarrow ATA(((X1, Y1), (X2, Y2)), Img)$;
  $Img \leftarrow Sort(Img)$;
  Determine the relationship between pixels and T in the character area: $Char \leftarrow Getchar(Img, T)$;
  if $Char == True$ **then**
    The real character area in Img is less than T;
    $X_1 \leftarrow X_1 + col\_min$;
    $Y_1 \leftarrow Y_1 + row\_min$;
    $X_2 \leftarrow X_1 + col\_max$;
    $Y_2 \leftarrow Y_1 + row\_max$;
  else
    The real character area in Img is greater than T;
    $X_1 \leftarrow X_1 + col\_min$;
    $Y_1 \leftarrow Y_1 + row\_min$;
    $X_2 \leftarrow X_1 + col\_max$;
    $Y_2 \leftarrow Y_1 + row\_max$;
  end if
until All coordinates are processed.

pixels belonging to the character area among the edge pixels through the pixel threshold. After determining the pixel range of the character area according to the threshold, it is assumed that the pixel area that is less than the threshold belongs to the character area. If there are pixels less than the threshold value in the edge pixels of the character, it is judged that the pixel belongs to the character area, and the edge is expanded, that is, the prediction box coordinates are updated according to Equation (3). Algorithm 3 details the execution steps of the candidate box expansion algorithm.

$$((X_1, Y_1), (X_2, Y_2)) = \begin{cases} X_1 \leftarrow X_1 - 1, & E3_i < T, i = 1, \ldots, n \\ Y_1 \leftarrow Y_1 - 1, & E1_i < T, i = 1, \ldots, n \\ X_2 \leftarrow X_2 + 1, & E4_i < T, i = 1, \ldots, n \\ Y_2 \leftarrow Y_2 + 1, & E2_i < T, i = 1, \ldots, n \end{cases} \quad (3)$$

where Ej_i(j = 1,...,4; i = 1,...,n) is the i-th element of the edge of the prediction box area extracted according to

Algorithm 3 Candidate Box Expansion Algorithm

Input: $u_1((X_1, Y1), (X_2, Y_2))$, $Image$, T;
Output: Updated $((X_1, Y1), (X_2, Y_2))$.

Grayscale Image: $Bd\_img \leftarrow Gray(Image)$;
repeat
  Get the gray value above the prediction box: $E1 \leftarrow Bd\_img[X1 : X2, Y1]$;
  Get the gray value under the prediction box: $E2 \leftarrow Bd\_img[X1 : X2, Y2]$;
  Get the gray value on the left side of the prediction box: $E3 \leftarrow Bd\_img[X1, Y1 : Y2]$;
  Get the gray value on the right side of the prediction box: $E4 \leftarrow Bd\_img[X2, Y1 : Y2]$;
  Get pixels of prediction box area: $Img \leftarrow Bd\_img[X1 : X2, Y1 : Y2]$;
  $Img \leftarrow Sort(Img)$;
  Determine the relationship between pixels and T in the character area: $Char \leftarrow Getchar(Img, T)$;
  if $Char == True$ **then**
    $E3i<T : X_1 \leftarrow X_1 - 1$;
    $E1i<T : Y_1 \leftarrow Y_1 - 1$;
    $E4i<T : X_2 \leftarrow X_2 + 1$;
    $E2i<T : Y_2 \leftarrow Y_2 + 1$;
  else
    $E3i>T : X_1 \leftarrow X_1 - 1$;
    $E1i>T : Y_1 \leftarrow Y_1 - 1$;
    $E4i>T : X_2 \leftarrow X_2 + 1$;
    $E2i>T : Y_2 \leftarrow Y_2 + 1$;
  end if
until All coordinates are processed.

the coordinates of the prediction box and the image to be detected. j = 1, 2, 3, 4 means the edge of up, down, left and right.

C. NON-STANDARD BOX REMOVAL MODULE

The original prediction box detected by the deep learning text detection algorithm often detects some non-character regions as characters, and the characters are misrecognized. These erroneously detected information will also lead to a decrease in algorithm precision and recall, so we need to filter out these prediction boxes and then eliminate them. In this article, a non-standard box removal algorithm NBR (Non-standard box removal) is proposed for this problem. We remove the non-standard box in all prediction boxes, which can further improve the precision and thus improve the detection effect.

(a) (b)

FIGURE 7. Before and after the prediction box is processed by NBR algorithm. (a)Before NBR algorithm processing; (b)After NBR algorithm processing.

The effect before and after processing by the non-standard box removal algorithm is shown in Fig. 7.

The black box and red box in Fig. 7(a) are the original prediction boxes after deep learning detection and NMS processing, where red is the non-standard prediction box, and Fig. 7(b) is the result after processing by the NBR algorithm. We can clearly see that the red non-standard box is removed after NBR processing. We believe that in the same text image, the size of adjacent characters is roughly the same. The design of the non-standard box removal algorithm is mainly according to this principle. The algorithm mainly judges whether the prediction box belongs to the standard box by the coordinate area threshold and the coordinate area of the prediction box. If the predicted box coordinate area is less than the maximum area threshold and greater than the minimum area threshold, it is determined that this coordinate belongs to the standard box, otherwise it does not. Algorithm 4 details the execution steps of the non-standard box removal algorithm.

Algorithm 4 Non-Standard Box Removal Algorithm

Input: The set of coordinates processed by BR, CBE algorithm: S_1;

Output: The set of S_1 is processed by NBR algorithm: S_2.

Sort all coordinates in set S1: Sort(S_1)
Initialize the set S_2 : $S_2 \leftarrow \{\emptyset\}$;
repeat
  Calculate 5 coordinate areas around the same line of S_{1i}: $Area_1, Area_2, Area_3, Area_4, Area_5$;
  Calculate the average of $Area_1, Area_2, Area_3, Area_4, Area_5$: $AVG\_C \leftarrow (Area_1 + Area_2 + Area_3 + Area_4 + Area_5)/5$;
  Calculate the maximum area threshold: $T\_MAX \leftarrow 1.4 * AVG\_C$;
  Calculate the minimum area threshold: $T\_MIN \leftarrow 0.5 * AVG\_C$;
  if $T\_MIN < Area_{S_{1i}} < T\_MAX$ **then**
    Save the i-th element in S1 to S2: $S_2.get(S_{1i})$;
  else
    $PASS$;
  end if
until All coordinates in S1 are processed.

D. REPEAT BOX REMOVAL MODULE

The original prediction box detected by the deep learning text detection algorithm often causes the two detection boxes to detect the same character, resulting in overlapping detection problems. The coordinates of the prediction box we get will be repeated after these problems are processed by the BR, CBE, and NBR algorithms. Analyzing the reasons, we found that after processing by the BR, CBE, and NBR algorithms, the results obtained by the two prediction boxes with overlap detection are the same. Therefore, this article proposes a repeat box removal algorithm RBR (Repeat box removal) for this problem. We will deduplicate the obtained prediction box coordinates. The effect before and after the three algorithms are shown in Fig. 8.

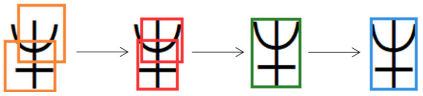


FIGURE 8. The prediction box after BR, CBE and RBR algorithm before and after processing.

The orange box in Fig. 8 is the original prediction box after deep learning detection and NMS processing. Here are the same characters detected by the two prediction boxes. Red is the prediction box processed by the BR algorithm, green is the prediction box processed by the CBE algorithm, and blue is the prediction box processed by the RBR algorithm. Here, the green actually has two prediction boxes, but it is repeated. This article creates a new empty set S3, and the repeated box removal algorithm mainly determines whether the coordinates in S2 need to be saved to S3 by judging whether the coordinates in S2 exist in S3. Algorithm 5 details the execution steps of the repeated box removal algorithm.

Algorithm 5 Repeated Box Removal Algorithm

Input: The set of S_1 is processed by NBR algorithm: S_2;

Output: The set of S_2 is processed by RBR algorithm: S_3.

Initialize the set S_3 : $S_3 \leftarrow \{\emptyset\}$;
repeat
  Take out the coordinates S_{2i} in S_2;
  if $S_{2i} \in S_3$ **then**
    $PASS$;
  else
    Save the i-th element in S2 to S3: $S_3.get(S_{2i})$;
  end if
until All coordinates in S2 are processed.

IV. EXPERIMENT AND RESULTS

In order to verify the effectiveness of the method in this article, we conducted experiments on five datasets. The five datasets are: standard Yi, Chinese2k, English2k,

ICDAR 2015 and ICDAR 2017(CTW-12k). And the standard Yi dataset is manually annotated by our team, while the Chinese2k dataset and English2k datasets are publicly released dataset of South China University of Technology in 2016. Both ICDAR 2015 and ICDAR 2017(CTW-12k) are well-known datasets in scene text detection and recognition tasks.

A. DATASETS

Standard Yi used in this article was provided by experts in Liangshan Yi Autonomous Prefecture, Sichuan Province. These samples are mainly obtained by scanning local newspapers and books into PDF documents. After we got the samples, we divided each page of the PDF document through a computer program to obtain 207 standardized Yi images, and then manually annotated them to finally get labels. The annotation format of the Standard Yi is $(x_1, y_1, x_1, y_2, x_2, y_2, x_2, y_1)$, which is the coordinates of the upper left corner, the lower left corner, the lower right corner, and the upper right corner of the Prediction box. This is our training data set. In the test data set, we obtained 50 document images and labels by the same method. Because text detection needs to face various complex scenes, not only to obtain better results in relatively clean and tidy document images, but also to obtain better performance in the presence of noise. In order to make the text detection more robust, to deal with complex scenes. We add noise to the original document image, using Gaussian noise and salt-and-pepper noise, respectively. Finally, we get 2277 image training samples.

The image of **Chinese2k and English2k** include streets, buildings, shops, office buildings, restaurants, stations, subways. The text content includes traffic signs, street signs, book covers, outdoor advertisements, notice boards, various signs. The lighting conditions of the image are diverse, including sunny and cloudy days, day and night. The current data set was released by the Human-computer Interaction and Intelligent Laboratory of South China University of Technology in 2016 [34]. Chinese2k only contains annotations for Chinese characters. The annotation format of the Chinese2k is (x, y, w, h), which is the position, width and height of the upper left corner of the rectangular box. English2k contains character annotations and word annotations. The annotation format of the English2k is (x, y, w, h, label), which is the position, width, height and category label of the upper left corner of the rectangular box. The Chinese2k and English2k can be used for research tasks such as text detection and recognition, including character location, character recognition, word location, word recognition [35].

ICDAR 2015 [38] was introduced in the ICDAR 2015 Robust Reading Competition for incidental scene text detection, consisting of 1000 training images and 500 testing images, both with texts in English. The annotations are at the word level using quadrilateral boxes. The annotation format of the ICDAR 2015 is $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$,text label), which is the coordinates of the upper left corner, the lower left corner, the lower right corner, the upper right corner and text label of the Prediction box. The text label '###' means the text is illegible.

ICDAR 2017(CTW-12k) [39] has a total of 12263 images, of which 8034 is used as the training set and 4229 is used as the test set. Use quadrilateral boxes to label text lines. Most of the datasets are natural scenes taken by cameras, and some are screenshots; it contains most scenes, such as outdoor streets, indoor scenes, mobile phone screenshots, etc. The annotation format of the datasets is $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$, the degree of difficulty in recognition, text label), which is the coordinates of the upper left corner, the lower left corner, the lower right corner, the upper right corner,the degree of difficulty in recognition and text label of the Prediction box. The degree of recognition difficulty is represented by 0 or 1, 0 means easy to recognize, that is, the text in the image is clearly visible; 1 means it is difficult to recognize, that is, the text in the image is small or fuzzy.

TABLE 1. The mark information of English2k, Chinese2k and standard Yi. NITR is the number of training set images; NCTR is the number of characters in training set; NITE is the number of test set images; NCTE is the number of characters in test set; TI is the number of total images; TC is the number of total characters.

Dataset	NITR	NCTR	NITE	NCTE	TI	TC
English2k	813	14888	349	6506	1162	21394
Chinese2k	1861	23768	355	4626	2216	28394
Standard Yi	207	104749	51	26927	257	131676
ICDAR 2015	1000	11886	500	5230	1500	17116
CTW-12k	8034	65248	4229	34281	12263	99529

B. METRICS

In order to evaluate the detection effect of text detection, we use the precision, recall and the comprehensive evaluation index F-measure defined in the natural scene text detection competition to evaluate the performance of text area detection. Recall reflects how many labeled texts are predicted correctly and precision tells us how many text predictions are correct. Here we need to know an important indicator, the Intersection over Union (IoU), the specific definition is shown in Equation (4).

$$IoU = \frac{area(C) \bigcap area(G)}{area(C) \bigcup area(G)} \tag{4}$$

where C and G are prediction boxes and true boxes.Through IoU we can get the definition of precision and recall as follows.

$$P = \frac{T_p}{C} \tag{5}$$

$$R = \frac{T_p}{T} \tag{6}$$

where Tp is the number of correct detection results, C is the number of detection result candidate boxes, and T represents the number of real boxes. The comprehensive evaluation standard F-measure is precision and recall weighted harmonic average, defined as follows.

$$F = \frac{(\alpha^2 + 1)P * R}{\alpha^2(P + R)} \tag{7}$$

When the parameter $\alpha = 1$, it is the most common F1.

$$F1 = \frac{2P * R}{P + R} \tag{8}$$

In this article, IoU is set to 0.5 and the comprehensive evaluation index is evaluated by F-measure with parameter $\alpha = 1$, which is F1.

C. IMPLEMENTATION DETAILS

The experiment was conducted under the hardware of Intel CPU i7-7700, graphics card NVIDIA GeForce GTX 2070S, solid state hard disk 256GB, memory 16GB. The development environment uses PyCharm 2018.3.8 Professional Edition, deep learning framework TensorFlow = 1.11.0, implemented in Python language, and visualize the training output and training results with the help of visualization tools such as TensorBoard and Matplotlib. The learning rate of the deep learning model is uniformly adopted by Adam and set to 0.001, to prevent the learning rate of the later network training from being too small and causing the network parameters to fall into a local optimal solution. It should be noted that some of the text detection models used in this experiment are used to detect text lines, so we need to fine-tune the network to adapt to character detection.

D. EXPERIMENTAL RESULTS AND ANALYSIS

In order to show the effectiveness of the proposed algorithm on the deep learning text detection model, the experiment verifies the proposed algorithm from four parts. The first part is to analyze the impact of the four modules BR, CBE, NBR and RBR on the detection results. The second part is to compare the effect of the mainstream deep learning text detection model before and after adding the algorithm proposed in this article. The third part is to analyze the robustness of the post-processing method proposed in this article, and use the method of this article to conduct experiments under different datasets. The third part is to analyze the robustness of the post-processing method proposed in this article, and experiment with different models under different datasets. The fourth part is to analyze the impact on the prediction processing speed before and after adding this module. The dataset used in the first two parts is the standard Yi dataset collected by ourselves, and the latter two parts use the Chinese2k, English2k, standard Yi, ICDAR 2015 and ICDAR 2017(CTW-12k) datasets.

1) INFLUENCE OF FOUR MODULES ON THE TEST RESULTS

The original deep learning model used in the first part is East [29], and its basic network is resnet50. Table 2 summarizes the detection results of adding different modules for text detection. BR, CBE, NBR, and RBR respectively represent the background removal module, the candidate box expansion module, the non-standard box removal module, and the repeated box removal module. 0 means not to add the module, 1 means to add the module, all 0 means not to add any module, which means the detection result of the original deep learning model. All 1 means all four modules are added, and the effect is the best at this time. Among the four modules, the BR module makes the distance between the prediction box and the character smaller, and the detection result is better. The CBE module mainly solves the problem of detection deviation. The NBR module mainly solves the problem of character misrecognition. The CBE module and the RBR module together solve the problem of overlapping detection. From the experimental results in Table 2, it can be seen that the performance effect of the original model without adding any post-processing strategy is the lowest. In the dataset, its precision is 0. 66, recall is 0. 69, and F1 is only 0. 67. Compared with the model performance of adding four post-processing modules, the difference is about 30%.

TABLE 2. The detection results of different strategies. BR, CBE, NBR and RBR respectively represent the four different modules proposed in this article.

Serial number	BR	CBE	NBR	RBR	Precision(%)	Recall(%)	F1(%)
1	0	0	0	0	66.4	69.3	67.8
2	0	0	0	1	67.5	69.3	68.4
3	0	0	1	0	68.2	69.3	68.7
4	0	0	1	1	72.4	69.3	70.8
5	0	1	0	0	78.6	81.6	80.1
6	0	1	0	1	84.1	81.6	82.8
7	0	1	1	0	79.4	81.6	80.5
8	0	1	1	1	83.7	81.6	82.6
9	1	0	0	0	70.6	73.9	72.2
10	1	0	0	1	71.8	73.9	72.8
11	1	0	1	0	72.2	73.9	73.0
12	1	0	1	1	75.9	73.9	74.9
13	1	1	0	0	82.5	89.4	85.8
14	1	1	0	1	88.3	89.4	88.8
15	1	1	1	0	84.6	89.4	86.9
16	1	1	1	1	**94.2**	**89.4**	**91.7**

A comparative analysis was conducted with the presence or absence of the four post-processing modules as a single variable. By comparing the 1, 9 sets of data in Table 2, it can be seen from the experimental results that adding the background removal module increased 6.3% on precision, 6.6% on recall, and 6. 5 on F1, without adding any modules. By comparing 1, 5 sets of data, it can be seen from the results in the Table 2 that adding the expansion module of the prediction box has an increase of 18.4% on precision, 17.7% on recall, and 18.1% on F1. By comparing the 1, 3 sets of data, it can be seen from the results in the Table 2 that the addition of the non-standard box removal module is 2.7% higher in precision and 1.3% higher in F1 than without any module. By comparing the 1, 2 sets of data, it can be seen from the results in the Table 2 that the addition of the repeated box removal module is 1.7% higher in precision and 0.9% higher in F1 than without adding any modules.

Fig. 9 shows the effect of using each post-processing module alone on the detection results of the original deep learning model. The abscissa indicates the four modules proposed in this article, and the ordinate indicates the improvement results of the detection after adding one module proposed in this article alone. Here, the CBE module is used as an example. In the three histograms, the light blue (Precision) indicates the

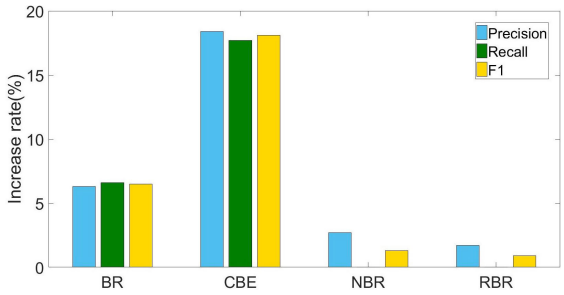


FIGURE 9. The impact of separate modules on the original model.

precision improvement result, which is equal to the precision after adding the CBE module minus the precision of the original detection. Similarly, the dark green (Recall) indicates the result of increasing the recall, and the yellow (F1) indicates the result of increasing the F1 value. The experiment fully verified that the post-processing module proposed in this article can effectively improve the text detection task, especially when all modules are used together, the improvement effect is most obvious. Through Fig. 9, we can get the influence of the four modules on the original model, which should be CBE, BR, NBR, RBR according to the impact from the largest to the smallest. Compared with the other three modules, the CBE module has improved greatly in terms of precision, recall and comprehensive evaluation F1. This is mainly because the CBE module expands the prediction box. Many of the prediction boxes before processing have detection deviation problems, and the CBE module just solves this problem, so the improvement is huge. The improvement of the BR module is also more obvious because the distance between the prediction box and the characters around the characters obtained by removing the background area through the BR module is smaller (finer), which makes the IoU larger, so the improvement is also more obvious. The BR and CBE modules have improved precision and recall in here. From Fig. 9, we can see that the NBR module and RBR module have no improvement on recall, and a little improvement on precision. This is because these two modules mainly remove the inaccurate prediction boxes in the prediction box, but the prediction box itself has not changed, but the number of prediction boxes finally obtained becomes less, so there is no improvement in recall at all, and there is a little amount of precision. In other words, the NBR and RBR modules are mainly optimized for improving precision.

2) COMPARISON OF MAINSTREAM DEEP LEARNING TEXT DETECTION METHODS

In the second part, 7 deep learning text detection models are added to compare the detection results before and after the method in this article. The comparison networks used in this article are the LSAE [33] proposed by Tian and Shu in CVPR 2019, CTPN [23] proposed by Tian and Huang *et al.* in ECCV 2016, SegLink [22] proposed by Shi and Bai *et al.* in CVPR 2017, TextBoxes [26] proposed by Liao and Shi *et al.* in AAAI 2017, EAST [29] proposed by Zhou and Yao *et al.* in CVPR 2017, TextBoxes++ [36] proposed by Liao and Shi *et al.* in IEEE Transactions on Image Processing, CRAFT [37] proposed by Baek and Han *et al.* in CVPR 2019. Since some of the original networks detect text lines, a few changes are made on the basis of each network model, including input and output, so that it can adapt to the different datasets. Fig. 10 shows the detection results of various deep learning detection models before and after using the post-processing method proposed in this article.

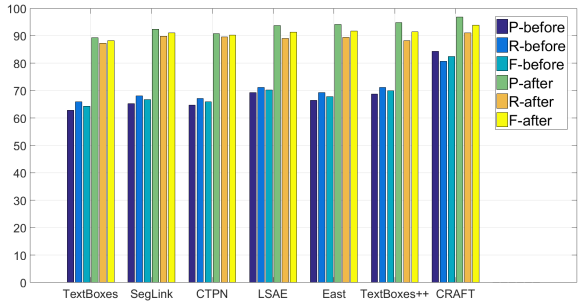


FIGURE 10. The detection results of different deep learning detection models before and after using the post-processing method proposed in this article.

In Fig. 10, the abscissa represents different text detection models, and the ordinate represents the detection results. P-before, R-before, F-before indicate the precision, recall and F1 value of the comprehensive evaluation index without adding the method in this article. P-after, R-after, and F-after respectively indicate the precision, recall, and comprehensive evaluation index F1 value after adding the method in this article.

Table 3 summarizes the detection results before and after adding this post-processing module using different deep learning text detection networks. It can be seen from the experimental results that the performance effect of the original deep learning model without adding any post-processing strategy is relatively low. Compared with the model performance of adding four post-processing modules, the difference is about 30%. This further shows that our proposed algorithm is effective.

The detection results before and after adding the post-processing module proposed in this article by comparing 7 deep learning text detection models. We can find that after adding the four post-processing modules proposed in this article, the original deep learning detection model has improved significantly in terms of precision, recall, and comprehensive evaluation of F1. This is closely related to our proposed post-processing module.

3) ROBUST ANALYSIS

In the third part, in order to verify the robustness of the method in this article, different deep learning text detection methods were used to conduct experiments on different datasets, and compared with the detection results of the

TABLE 3. The detection results are compared before and after post-processing in this article.

Serial number	Method	Precision(%)	Recall(%)	F1(%)
1	TextBoxes [26]	62.8	65.9	64.3
2	TextBoxes+ours	**89.3**	**87.3**	**88.3**
3	SegLink [22]	65.3	68.1	66.7
4	SegLink+ours	**92.4**	**89.9**	**91.1**
5	CTPN [23]	64.7	67.2	65.9
6	CTPN+ours	**90.8**	**89.6**	**90.2**
7	LSAE [33]	69.3	71.2	70.2
8	LSAE+ours	**93.7**	**89.1**	**91.3**
9	East [29]	66.4	69.3	67.8
10	East+ours	**94.2**	**89.4**	**91.7**
11	TextBoxes++ [36]	68.7	71.2	69.9
12	TextBoxes++ +ours	**93.7**	**88.3**	**90.9**
13	CRAFT [37]	84.3	80.7	82.5
14	CRAFT+ours	**95.8**	**91.0**	**93.4**

TABLE 4. The detection results of the method in this article on English2k and Chinese2k datasets.

		English2k			Chinese2k		
Num	Method	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
1	CTPN [23]	78.3	73.7	75.9	69.3	87.5	77.3
2	CTPN+ours	**83.8**	**77.6**	**80.6**	**72.7**	**89.9**	**80.4**
3	East [29]	84.1	75.1	79.3	71.1	88.4	78.8
4	East+ours	**86.2**	**78.9**	**82.4**	**74.6**	**90.8**	**81.9**
5	TextBoxes [26]	79.2	70.6	74.7	70.9	84.5	77.1
6	TextBoxes+ours	**82.6**	**74.2**	**78.2**	**73.7**	**87.3**	**79.9**
7	SegLink [22]	75.8	71.7	73.7	71.8	77.9	74.7
8	SegLink+ours	**77.9**	**73.5**	**75.6**	**74.2**	**79.6**	**76.8**
9	LSAE [33]	86.7	80.5	83.5	78.3	81.6	79.9
10	LSAE+ours	**88.3**	**82.9**	**85.5**	**81.1**	**83.4**	**82.2**
11	TextBoxes++ [36]	79.6	76.8	78.2	74.6	79.8	77.1
12	TextBoxes++ +ours	**81.6**	**79.7**	**80.6**	**76.8**	**81.9**	**79.39**
13	CRAFT [37]	87.5	84.6	86.0	82.6	79.8	81.2
14	CRAFT+ours	**90.4**	**87.1**	**88.7**	**84.7**	**82.4**	**83.5**

post-processing method added in this article. The comparison networks used here LSAE [33], CTPN [23], SegLink [22], TextBoxes [26], EAST [29], TextBoxes++ [36] and CRAFT [37]. Table 4 and Table 5 summarizes the detection results before and after using the post-processing method of this article on different datasets. From the experimental results in Table 4, it can be seen that on the English2k and Chinese2k datasets, the method of this article has improved at least 3% in precision, recall, and F1 value. As can be seen from the experiment results of Table 5, and in ICDAR 2015 and ICDAR 2017(CTW-12k) datasets, the methods described of this article in terms of accuracy, recall and F1 value aspects of at least a 1.6% increase.

Fig. 11 shows the detection results of EAST [29], LSAE [33] and CRAFT [37] models before and after using the method proposed in this article in different datasets. We can see more clearly that our method has a certain improvement on the results of text detection on the five datasets. The abscissa represents different datasets, and the ordinate represents the evaluation results in Fig. 11. Fig. 11(a), Fig. 11(b), and Fig. 11(c) respectively show the evaluation results of the precision, recall, and F1 value before and after using different methods on different datasets. In the figure, the dark blue (EAST-before) represents the detection results of the EAST model, the light blue (EAST-after)

TABLE 5. The detection results of the method in this article on ICDAR 2015 and ICDAR 2017(CTW-12k) datasets.

		ICDAR 2015			CTW-12k		
Num	Method	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
1	CTPN [23]	70.1	49.2	57.8	54.8	46.2	50.1
2	CTPN+ours	**72.9**	**50.6**	**59.7**	**57.4**	**48.8**	**52.8**
3	East [29]	75.3	69.4	72.2	52.6	45.7	48.9
4	East+ours	**80.5**	**72.7**	**76.4**	**56.2**	**48.6**	**52.1**
5	TextBoxes [26]	73.6	70.7	72.1	59.1	51.4	55.0
6	TextBoxes+ours	**78.1**	**73.5**	**75.7**	**63.8**	**54.7**	**58.9**
7	SegLink [22]	69.5	72.3	70.9	66.2	57.8	61.7
8	SegLink+ours	**71.9**	**75.8**	**73.8**	**69.4**	**59.5**	**64.1**
9	LSAE [33]	80.9	81.6	81.2	71.3	62.1	66.4
10	LSAE+ours	**84.5**	**85.1**	**84.8**	**74.9**	**64.2**	**69.1**
11	TextBoxes++ [36]	82.7	71.7	76.8	67.1	60.3	63.5
12	TextBoxes++ +ours	**86.8**	**74.6**	**80.2**	**69.8**	**62.4**	**65.9**
13	CRAFT [37]	83.6	79.8	81.7	77.6	67.4	72.1
14	CRAFT+ours	**87.8**	**84.3**	**86.0**	**79.3**	**68.5**	**73.5**

represents the detection results of the EAST model with the post-processing method of this article. The dark yellow (LSAE-before) represents the detection results of the LSAE model and the yellow (LSAE-after) indicates the detection results of the LSAE model after adding this post-processing method. The blue-green (CRAFT-before) represents the detection results of the CRAFT model and the green (CRAFT-after) indicates the detection results of the CRAFT model after adding this post-processing method. As can be seen from Fig. 11, our method has great results on 5 datasets such as ICDAR 2015 and ICDAR 2017(CTW-12k). Among them, the precision, recall, and F1 value have been improved by at least 2.2%, 1.6%, and 1.9%, respectively.

Fig. 12 shows the results of testing and evaluating different IoU values when we use the same text detection model to add the post-processing method of this article. In the Fig. 12, the abscissa represents different IoU values, and the ordinate represents the evaluation results. Among them, red represents the recall, green represents the precision, and blue represents the F1 value. We take 0.2, 0.3, 0.4, . . ., 0.9 for the IoU values, respectively. The most ideal test result for the test task is that the test result between the IoU value and the small value should not change much. Fig. 12(a), Fig. 12(b), Fig. 12(c), Fig. 12(d), Fig. 12(e) show the detection results of using the datasets of English2k, Chinese2k, Standard yi, ICDAR 2015, ICDAR 2017(CTW-12k) to take different IoU values. From Fig. 12, we can see that when the IoU value is less than 0.8, on the English2k and Chinese2k datasets, whether it is the precision, the recall, or the F1 value, the test results can achieve a good result. And when the IoU value is less than 0.6, on the standard Yi dataset, the value of precision, recall and F1, the test results can achieve a good result. In this article calls the IoU values 0.8 and 0.6 here as change nodes, and we analyze the reasons that cause the threshold change nodes to be different. After visualizing the test set annotation data, the comparison with the model detection effect diagram found that the reason for the different change nodes is the problem of the annotation method. In English2k and Chinese2k, there is a small distance between the characters

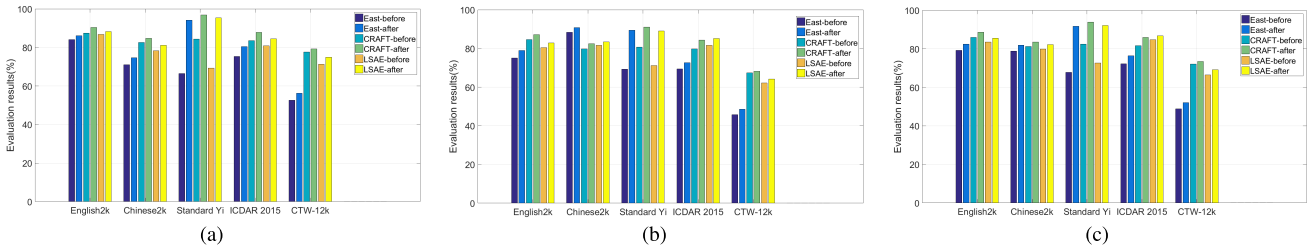


FIGURE 11. The detection results of different datasets before and after using the method proposed in this article. (a)Precision; (b)Recall; (c)F1.

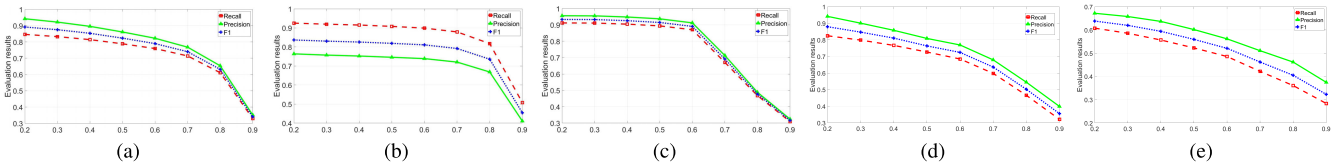


FIGURE 12. The evaluation results of different thresholds on different datasets by the post-processing method in this article. (a)English2k; (b)Chinese2k; (c)Standard yi; (d)ICDAR 2015; (e)ICDAR 2017(CTW-12k).

of each labeled data and the labeled coordinates, and there is a certain distance between the characters and the coordinates in the labeled data of the standard Yi dataset. After being processed by our method, the distance between our predicted coordinates and characters is very small, so it causes the difference of the node of the IoU value change, that is, the change node of the standard Yi dataset is smaller. In a word, the detection results of different IoU on 5 datasets also show that our method is great.

4) TIME COMPLEXITY ANALYSIS

In the fourth part, in order to verify the effect of the method in this article on the processing time of text detection, experiments were performed on different datasets using the same model, and the processing time after adding the method of this article was compared and analyzed. Fig. 13 is a histogram of the processing speed on different datasets before and after adding the text module. Here, the abscissa indicates different datasets, the ordinate indicates the evaluation result, and the light blue(Before) indicates the processing speed when the text module is not added, orange(After) means the processing speed after adding this module. Since we are analyzing the processing speed, FPS is used here for evaluation. It indicates how many pictures are processed in one second. The larger the value, the faster the processing speed and the better the effect. It can be seen from Fig. 13 that on the English2k, Chinese2k and ICDAR 2015 datasets, the processing speed is much faster than that on the standard Yi. This is because the number of characters per image in the test set on the English2k, Chinese2k and ICDAR 2015 datasets is relatively small. From Table 1, it can be seen that the average is 19, 13 and 11, respectively, while the average number of characters in each image of the standard Yi is approximately 528. On the datasets of English2k, Chinese2k, standard Yi, ICDAR 2015 and ICDAR 2017(CTW-12k), the processing speed after adding this method has dropped by 7.2%, 4.3%, 6.7%, 4.6%

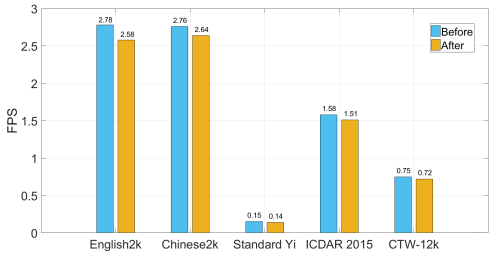


FIGURE 13. The speed of processing before and after adding this article's modules.

and 4.1% respectively, which are all within 10%, so it is acceptable.

V. CONCLUSION

General deep learning text detection is divided into network prediction and post-processing. Network prediction outputs the coordinates of the candidate box, and the post-processing further processes the coordinates of the candidate box. Generally, NMS and adjacent connection merge are used to make the text detection result more accurate. This article optimizes the post-processing of text detection, and proposes four closely connected post-processing modules, which solves the problems of character misrecognition, detection deviation, overlap detection, etc., realizes the refinement of text detection, and improves the text detection effect. The CBE module solves the problem of detection deviation, the CBE module and the RBR module together solve the problem of overlapping detection, and the NBR module solves the problem of character misrecognition. This article combines deep learning and fine post-processing to make some contributions to the field of text detection. Among the four modules, the BR module and CBE module have improved the precision and recall of text detection, while the NBR module and RBR module have improved the precision significantly. However, the method in this article is not applicable to all text detection,

and has certain defects. Since the method in this article mainly performs post-processing on the rectangular prediction box, the method in this article can only be used for character detection and text line detection that use rectangular boxes as detection results. Therefore, we need to further research the post-processing of curved text detection in the future.

REFERENCES

[1] Z. Ying, Y. Zhao, C. Xuan, and W. Deng, "Layout analysis of document images based on multifeature fusion," *J. Image Graph.*, vol. 25, no. 2, pp. 311–320, 2020.

[2] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal, "Offline recognition of Devanagari script: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 41, no. 6, pp. 782–796, Nov. 2011, doi: 10.1109/TSMCC.2010.2095841.

[3] P. Shivakumara, T. Q. Phan, and C. L. Tan, "A Laplacian approach to multi-oriented text detection in video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 2, pp. 412–419, Feb. 2011, doi: 10.1109/TPAMI.2010.166.

[4] M. Buta, L. Neumann, and J. Matas, "FASText: Efficient unconstrained scene text detector," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1206–1214, doi: 10.1109/ICCV.2015.143.

[5] C. Liu, C. Wang, and R. Dai, "Text detection in images based on unsupervised classification of edge-based features," in *Proc. 8th Int. Conf. Document Anal. Recognit. (ICDAR)*, Seoul, South Korea, 2005, pp. 610–614, doi: 10.1109/ICDAR.2005.228.

[6] C. Yu, Y. Song, Q. Meng, Y. Zhang, and Y. Liu, "Text detection and recognition in natural scene with edge analysis," *IET Comput. Vis.*, vol. 9, no. 4, pp. 603–613, Aug. 2015, doi: 10.1049/iet-cvi.2013.0307.

[7] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, San Francisco, CA, USA, Jun. 2010, pp. 2963–2970, doi: 10.1109/CVPR.2010.5540041.

[8] L. Neumann and J. Matas, "A method for text localization and recognition in real-world images," in *Proc. 10th Asian Conf. Comput. Vis. (ACCV)*. Berlin, Germany: Springer, 2010, pp. 770–783.

[9] G. Zhou, Y. Liu, Z. Tian, and Y. Su, "A new hybrid method to detect text in natural scene," in *Proc. 18th IEEE Int. Conf. Image Process.*, Brussels, Belgium, Sep. 2011, pp. 2605–2608, doi: 10.1109/ICIP.2011.6116199.

[10] L. Sun, Q. Huo, W. Jia, and K. Chen, "Robust text detection in natural scene images by generalized color-enhanced contrasting extremal region and neural networks," in *Proc. 22nd Int. Conf. Pattern Recognit.*, Stockholm, Sweden, Aug. 2014, pp. 2715–2720, doi: 10.1109/ICPR.2014.469.

[11] S. Tian, Y. Pan, C. Huang, S. Lu, K. Yu, and C. L. Tan, "Text flow: A unified text detection system in natural scene images," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 4651–4659, doi: 10.1109/iccv.2015.528.

[12] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Kauai, HI, USA, Dec. 2001, p. I, doi: 10.1109/CVPR.2001.990517.

[13] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004, doi: 10.1023/b:visi.0000013087.49260.fb.

[14] W. Kim and C. Kim, "A new approach for overlay text detection and extraction from complex video scene," *IEEE Trans. Image Process.*, vol. 18, no. 2, pp. 401–411, Feb. 2009, doi: 10.1109/TIP.2008.2008225.

[15] X. Bai, C. Yao, and W. Liu, "Strokelets: A learned multi-scale mid-level representation for scene text recognition," *IEEE Trans. Image Process.*, vol. 25, no. 6, pp. 2789–2802, Jun. 2016, doi: 10.1109/TIP.2016.2555080.

[16] W. Kai and S. Belongie, "Word spotting in the wild," in *Proc. 11th Eur. Conf. Comput. Vis. (ECCV)*. Berlin, Germany: Springer, 2010, pp. 591–604.

[17] B. Su, S. Lu, S. Tian, J. H. Lim, and C. L. Tan, "Character recognition in natural scenes using convolutional co-occurrence HOG," in *Proc. 22nd Int. Conf. Pattern Recognit.*, Stockholm, Sweden, Aug. 2014, pp. 2926–2931, doi: 10.1109/ICPR.2014.504.

[18] S. Tian, S. Lu, B. Su, and C. L. Tan, "Scene text recognition using co-occurrence of histogram of oriented gradients," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Washington, DC, USA, Aug. 2013, pp. 912–916, doi: 10.1109/ICDAR.2013.186.

[19] H. Xu and F. Su, "A robust hierarchical detection method for scene text based on convolutional neural networks," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Turin, Italy, Jun. 2015, pp. 1–6, doi: 10.1109/ICME.2015.7177494.

[20] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, Nov. 2017, doi: 10.1109/TPAMI.2016.2646371.

[21] Y. Liu and L. Jin, "Deep matching prior network: Toward tighter multi-oriented text detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 3454–3461, doi: 10.1109/CVPR.2017.368.

[22] B. Shi, X. Bai, and S. Belongie, "Detecting oriented text in natural images by linking segments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 3482–3490, doi: 10.1109/CVPR.2017.371.

[23] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, "Detecting text in natural image with connectionist text proposal network," in *Proc. 14th Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2016, pp. 56–72.

[24] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *Int. J. Comput. Vis.*, vol. 116, no. 1, pp. 1–20, Jan. 2016.

[25] D. Xiang, Q. Guo, and Y. Xia, "Robust text detection with vertically-regressed proposal network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2016, pp. 351–363.

[26] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "TextBoxes: A fast text detector with a single deep neural network," in *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2017, pp. 4161–4167.

[27] T. He, W. Huang, Y. Qiao, and J. Yao, "Accurate text localization in natural image with cascaded convolutional text network," 2016, *arXiv:1603.09423*. [Online]. Available: http://arxiv.org/abs/1603.09423

[28] W. He, X.-Y. Zhang, F. Yin, and C.-L. Liu, "Deep direct regression for multi-oriented scene text detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 745–753, doi: 10.1109/ICCV.2017.87.

[29] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: An efficient and accurate scene text detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 2642–2651, doi: 10.1109/CVPR.2017.283.

[30] X. Yang, D. He, Z. Zhou, D. Kifer, and C. L. Giles, "Learning to read irregular text with attention mechanisms," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, p. 3, doi: 10.24963/ijcai.2017/458.

[31] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai, "Multi-oriented text detection with fully convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 4159–4167, doi: 10.1109/CVPR.2016.451.

[32] D. Deng, H. Liu, X. Li, and D. Cai, "PixelLink: Detecting scene text via instance segmentation," in *Proc. 32st AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, 2018, pp. 1–8. [Online]. Available: https://arxiv.org/abs/1801.01315

[33] Z. Tian, M. Shu, P. Lyu, R. Li, C. Zhou, X. Shen, and J. Jia, "Learning shape-aware embedding for scene text detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 4229–4238, doi: 10.1109/CVPR.2019.00436.

[34] S. Zhang, M. Lin, T. Chen, L. Jin, and L. Lin, "Character proposal network for robust text extraction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Shanghai, China, Mar. 2016, pp. 2633–2637, doi: 10.1109/ICASSP.2016.7472154.

[35] S. Y. Zhang, "Deep model and its application to visual text analysis," Ph.D. dissertation, South China Univ. Technol., Guangzhou, China, 2016.

[36] M. Liao, B. Shi, and X. Bai, "TextBoxes++: A single-shot oriented scene text detector," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3676–3690, Aug. 2018, doi: 10.1109/TIP.2018.2825107.

[37] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 9357–9366, doi: 10.1109/CVPR.2019.00959.

[38] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny, "ICDAR 2015 competition on robust reading," in *Proc. 13th Int. Conf. Document Anal. Recognit. (ICDAR)*, Tunis, Tunisia, Aug. 2015, pp. 1156–1160, doi: 10.1109/ICDAR.2015.7333942.

[39] B. Shi, C. Yao, M. Liao, M. Yang, P. Xu, L. Cui, S. Belongie, S. Lu, and X. Bai, "ICDAR2017 competition on reading Chinese text in the wild (RCTW-17)," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Kyoto, Japan, Nov. 2017, pp. 1429–1434, doi: 10.1109/ICDAR.2017.233.

XIAOGANG QIU was born in Guangyuan, China, in 1996. He received the bachelor's degree from Mianyang Normal University, Mianyang, China, in 2018. He is currently pursuing the master's degree with Southwest University. His research interests include computer vision and deep learning.

SHANXIONG CHEN was born in Chongqing, China, in 1981. He received the Ph.D. degree in computer science from Chongqing University, Chongqing, China, in 2013. His research interests include data mining, pattern recognition, and compressed sensing.

RANKANG LI was born in Chongqing, China, in 1995. He received the bachelor's degree from Xinjiang University, Xinjiang, China, in 2018. He is currently pursuing the master's degree with Southwest University. His research interests include computer vision and deep learning.

DINGWANG WANG was born in Chongqing, China, in 1993. He received the bachelor's degree from the Chongqing University of Technology, Chongqing, in 2017. He is currently pursuing the master's degree with Southwest University. His research interests include computer vision and deep learning.

XIAOYU LIN was born in Bazhong, China, in 1997. She received the bachelor's degree from Yangtze Normal University, Chongqing, China, in 2018. She is currently pursuing the master's degree in software engineering with Southwest University. Her research interests include deep learning and image processing.