Received October 29, 2020, accepted November 3, 2020, date of publication November 6, 2020, date of current version November 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3036545

**TSER: A Two-Stage Character Segmentation Network With Two-Stream Attention and Edge Refinement**

JINYINGMING ZHANG\(^1\), JIN LIU\(^1\), (Member, IEEE), XIONGWEI XU\(^2\), PEIZHU GONG\(^1\), AND MINGYANG DUAN\(^1\)

\(^1\)College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China
\(^2\)College of Education, Shanghai Normal University, Shanghai 200234, China

Corresponding author: Xiongwei Xu (xiongweixushnu@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 61872231 and Grant 61701297.

**ABSTRACT** Segmenting characters in an image is a classic yet challenging task in computer vision. Correctly determining boundaries of adhesive characters with various scales and shapes is essential for character segmentation, especially for separating handwritten characters. Nevertheless, there is seldom work in the literature which can achieve satisfactory performance. In this article, by leveraging the ability of deep neural networks, we proposed a two-stage character segmentation network with two-stream attention and edge refinement (TSER) to tackle this problem. TSER firstly locates every character by object detection, then extracts their corresponding contours. In the process, a novel two-stream attention mechanism (TSAM) is proposed to make the network focus more on the discrepancy of character boundaries. Furthermore, a novel generating method is used to dynamically generate anchors on different feature levels to improve model’s sensitivity on the shapes and scales of characters. Eventually a cascaded edge refinement network is used to obtain contour of each character. To prove the efficiency and generalization ability of our model, we compared TSER with traditional algorithms and other deep learning models on two commonly used datasets in different segmentation tasks. The comparative result indicated that TSER reached state-of-the-art performance.

**INDEX TERMS** Character segmentation, deep learning, two-stream attention mechanism, cascaded edge refinement network.

**I. INTRODUCTION**

Character segmentation is a vital step in traditional optical text recognition process. Although its importance for text recognition declined with the development of deep-learning-based methods, such as encoder-decoder based models [1], [2]. It is still the core technology in lots of important real-world applications. For instance, recognizing CAPTCHA [3], which is an important technology in modern automatic systems, demands a robust algorithm to segment characters accurately for further recognition. In addition, recognizing inscriptions and handwritten documents also needs an effective segmentation method. However, correctly segmenting characters in various scenarios is challenging and still remains unsolved. The main reason is the boundaries of adhesive characters with different scales and shapes are difficult to determine, especially for handwritten and pictographic characters. Figure 1 shows some wrong cases that prediction models can easily make.

![FIGURE 1. Some segmentation results of the traditional projection algorithm. The wrong cases are labeled by red lines.](image-url)

For a long time, rule-based algorithms are the mainstream choice. For example, the connected component analysis is usually regarded as an effective method. An important concept existing in this method is called connected component.
It refers to the region that consists of adjacent pixels with the same pixel value. Le et al. [4] found out all the connected components and labeled them. Another conventional approach of great concern is the projection algorithm. Kesiman et al. [5] segmented characters by projecting images from two directions. Commonly, if the spacing of all characters is apparent, these methods are effectual and time-saving. However, when characters are overlapping or adhesive together, under-segmentation or over-segmentation often occurs. Therefore, it is difficult for the traditional methods to enhance the ability of segmentation from the data with a variety of distributions, which makes the above problems tough to be solved.

Inspired by several knowledge from different fields, deep learning has been developing rapidly. Deep learning is a representative approach of signal processing [6], [7] and it can address many tricky problems. For character segmentation, semantic segmentation networks such as FCN [8], SegNet [9], U-Net [10], etc. are preferred. However, all of them cannot obtain the subtle boundaries of characters, therefore cannot accurately segment characters. At the meantime, various instance segmentation methods, such as FCIS [11], Mask R-CNN [12] and PANet [13] can possibly tackle this problem by predicting coarse segmentation masks for characters within each fixed-size ROI patches. However, these methods are proposed to detect and delineate each distinct object of interest appearing in an image but fail to focus on character segmentation. Character segmentation task is different with the common instance segmentation task. For example, the feature and arrangement of characters may have some discrepancy with the common objects. Therefore, directly utilizing these instance segmentation networks to character segmentation cannot achieve good performance.

In this paper, we propose a two-stage character segmentation network, which can accurately segment characters under various situations, namely normal spacing, subtle spacing, adhesive characters, partially overlapping characters, characters with deflection angles, and characters with different scales and shapes, from text line images with random noise.

The main contributions of this paper are summarized as follows:

- We proposed a novel character segmentation network based on a two-stage detection network. This model is efficient in segmenting adhesive characters and can be generalized to other similar character segmentation tasks.
- A two-stream attention mechanism is proposed to guide the feature selection process of model. This attention mechanism contributes to distinguish the boundary of adhesive characters and assists to find out every character instance.
- A guided anchoring method is applied to produce sparse and appropriate anchors instead of the dense and redundant ones in traditional region proposal network. This module can reduce computational cost and generate anchors that meet various character shapes and scales.
- A cascaded edge refinement network is put forward as an auxiliary module to generate contours for characters from every bounding box because contours can surround characters better than rectangle boxes in most cases.

The remaining of this article is structured as follows: In Sect. 2, relevant researches in this field are presented. In Sect. 3, we present formal definitions for character segmentation task, and elaborate our solution in detail. In Sect. 4, we describe model training details. In Sect. 5, we show the results of comparative experiment and ablation experiment. Finally, in Sect. 6, some conclusions are drawn.

II. RELATED WORK

A. TRADITIONAL SOLUTION

Almost all traditional character segmentation methods are based on rules and conventional machine learning techniques. They are relatively simple and easy to understand, which makes them popular in the past. Projection algorithm used horizontal projection to slice the text line and vertical projection to slice the individual character. Le et al. proposed a connected-component-based method by introducing a classifier to further evaluate whether the connected component is text or not. Phan et al. [14] proposed a gradient-vector flow based method for video character segmentation. This method finds the spaces between characters by using minimum cost path estimation. Sharma et al. [15] proposed a character segmentation method that is sensitive to dominant points for multi-oriented video. Liang et al. [16] proposed a novel wavelet Laplacian method to segment the characters with arbitrary orientation. This method explores zero crossing points to find spaces between words or characters. These approaches perform well only when being applied to regular text and with little interference. If the image has some interference on the text, their performance degrades significantly.

B. SEMANTIC SEGMENTATION

Deep-learning-based character segmentation methods can handle the images with various interference. For semantic segmentation, FCN was firstly proposed to split the input image into different semantically interpretable categories. The encoder-decoder architecture of FCN was efficient and easy to be further expand, thus more FCN based approaches were proposed, such as U-Net, Deeplab [17], RefineNet [18] and so on. All of them can segment the characters with subtle spacing in the images accompanied by slight interference. However, these networks still fail to achieve a satisfactory segmentation result because they cannot completely handle adhesive characters with different scales, especially when there are some apparent interferences in the images.

C. OBJECT DETECTION

In terms of the object detection networks (ODN), almost all of them can be classified into single-stage or two-stage networks. Some popular two-stage ODNs such as R-CNN [19], Fast R-CNN [20] and Faster R-CNN [21] can be used to segment characters with particular improvements. For instance,
MS-CNN [22] introduced FPN [23] to boost the effect of detection on multi-scale images. R-FCN proposed a position-sensitive pooling method to accelerate the training process. To obtain better multi-scale feature maps, PANet applied an improved FPN with deeper sampling layers to their backbone network. In the meantime, many single-stage detection networks are also proposed continuously, such as SSD [24], YOLO [25], RetinaNet [26]. They are faster but have slightly lower accuracy than the two-stage models.

D. INSTANCE SEGMENTATION

Instance segmentation network (ISN), which creatively combines semantic segmentation network with ODN, can be used to character segmentation. It is an effective way that first uses ODN to find out different instances, and then classifies each pixel in every instance to segment instances precisely. To distinguish between different instances of the same category, ISN also requires creating an independent mask for each object. At present, the most popular ISNs are FCIS, Mask R-CNN, Tensor Mask [27], etc. ISN provides the possibility of segmenting the adhesive characters, we used the same idea in this work.

E. ATTENTION MECHANISM

Attention mechanism are widely used in computer vision [28], [29] and natural language processing [30]. Wang et al. [31] proposed a non-local network for video classification, which is based on a space-time dependency attention mechanism. Hu et al. [32] focused on the relationship between channels and features, introducing a SE module to adaptively obtain the channel-wise features. Similarly, CCnet [33] proposed a novel criss-cross attention module, which can be leveraged to capture contextual information from the long-range dependency in an effective way. Therefore, attention mechanism can be used to make model focus more on the boundary of characters.

F. ANCHOR

Anchor is a set of frames designed manually for object classification and box regression in ODN. No matter in a single-stage or two-stage detector, anchors are used to give approximate location of characters. A common way to generate anchors is sliding window method. This method firstly defines a certain number of anchors with a specific scale and aspect ratio. Then these anchors slide from top to bottom, left to right in a certain step on the image to detect characters. This technique is widely used in conventional detection networks such as Faster R-CNN, SSD, and RetinaNet. However, as object detection task becomes more complex, it becomes necessary to generate sparse anchors with high flexibility in shape and position. Therefore, guided anchoring was proposed by Wang et al. [34] to generate anchors by neural networks. The anchors generated in this way can have adaptive scales; in the meantime, redundant anchors are also reduced. In pursuit of an efficient anchor generation approach, our method also employs neural networks to produce anchors automatically.

So far, almost all character segmentation methods fail to accurately find out the boundary of multi-scale and adhesive characters. In the meantime, each method is designed for a specific task. There is no such a universal method that can be generalized to different character segmentation tasks. In this paper, we propose a novel attention mechanism to improve the feature representation ability of our model, and simultaneously utilize the guided anchoring approach to generate sparse anchors with high quality. Finally, an edge refinement network is proposed to get accurate contours of characters for further post-processing.

III. METHODS

A. PROBLEM DEFINITION

An RGB image $X_{h\times w\times c}$ with arbitrary size is the input for TSER, which can be defined as follow:

$$X_{h\times w\times c} = \begin{bmatrix} p_{1c} & \cdots & p_{1wc} \\ \vdots & \ddots & \vdots \\ p_{hc} & \cdots & p_{hwc} \end{bmatrix}, \quad c = 1, \ldots, n \quad (1)$$

where $h$ and $w$ represent the height and width of the inputted image. $c \in 1, 2, \ldots, n$ denotes the channel of inputted image. For RGB images, $n$ is equal to 3.

The objective of TSER is two folds: first, to locate every character precisely by providing a unique bounding box to every character respectively, and then to segment every character from images by semantic segmentation. Then, the whole process can be split into two sections correspondingly. Therefore, for the first section, output are the located boxes which can be defined as the following:

$$B_i = (x, y, w, h) \quad i = 1, 2, \ldots, m \quad (2)$$

where $B$ is a set of bounding boxes that marks the location of every character. $(x, y)$ denotes the coordinate of the center point of box, $(w, h)$ denotes the width and height of boxes. $I$ represents the number of characters in an image. In the second section, output are final character contours which can be defined as the following formula.

$$I_i = [(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)] \quad i = 1, 2, \ldots, m \quad (3)$$

where $I$ denotes a set of polygon coordinates that represents all the character contours in the image. Note that TSER also outputs a score list $S$ as Eq. (4).

$$S = [S_1, S_2, \ldots, S_m] \quad (4)$$

where $S$ represents the probability of whether the detected boxes are valid. It is an auxiliary value for determining whether the boxes are expected results.

B. OVERALL NETWORK ARCHITECTURE

Like most two-stage detection networks, TSER includes region proposal network and target detection network. As can be seen from Figure 2, the entire network consists of
three components. The first component is a multi-scale features extracting network, ResNet-FPN-TSAM, which adds TSAM with ResNet-FPN to make the obtained multi-scale feature maps retain more information from different feature layers. This mechanism is necessary for ODN, which requires both semantic and location information. Moreover, since the extracted features are shared between two stages, will directly affect the final detection results.

The second component is GA-RPN for generating sparse anchors and obtaining the region of interest (ROI). Generally speaking, the traditional region proposal network (RPN) is the first choice to generate ROIs, although the generated anchors are dense with redundant ones. In the meantime, the quality of generated anchors is dependent on hyper-parameters settings.

Thus, using neural network can generate more appropriate and sparse anchors that fit characters better, since the location and shape of the anchors can be predicted this way. As a result, effectively eliminating redundant ones. The third component is a character segmentation network that detects the final character position and generates contours for every character.

This component takes the proposal generated by GA-RPN as input, and then adjusts the proposal to precisely obtain the final detection result; eventually, performs binary mask segmentation in each ROI to obtain contours. Moreover, with help of a cascaded edge refinement network, the inter-class difference is amplified so that the obtained contours are more suitable for characters.

C. TWO-STREAM ATTENTION MECHANISM
High-level feature contains a lot of semantic information and larger receptive field, making it suitable for detecting larger objects. In contrast, low-level feature contains much detailed information and smaller receptive field, thus, is suitable for detecting smaller objects. Therefore, features from different levels are directly added in traditional FPN based models. However, these methods ignore the influence of different channels in feature maps. Usually, different channels describe different features, which means that each channel demands different attention weights. For character segmentation, most attention should be paid to those channels containing important information about characters such as boundary, shape, aspect ratio, etc.

To address this issue, TSER uses a novel two-stream attention mechanism (TSAM) to improve feature expression ability, by combining information of feature maps from different levels. On the one hand, TSAM uses efficient feature representation ability of high-level feature maps to guide the feature selection of low-level feature maps. On the other hand, TSAM takes advantage of the detailed information described by low-level feature maps to change the attention distribution on each pixel of high-level feature maps. In this way, TSAM enables feature extraction process to retain both the detailed information and semantic information as much as possible. The whole architecture of TSAM and how to embed it in ResNet-FPN is respectively shown in Figure 3 and Figure 4. First, a channel-wise attention module is used to generate a probability vector by global pooling operation and softmax activation function. This vector describes the relative importance among different features in different channels and can be used to influence the feature distribution of low-level feature maps. As shown in Eq. (5) and Eq. (6), the high-level feature maps are squeezed to a weight vector that affects the low-level feature maps channel-wisely.

\[
Z = \text{Concat}^n_{c=1} \left( \sum_{j=1}^{m} \frac{P_j}{m} \right)_{c} 
\]

\[
\text{CAG} (X_L, X_H) = \text{Concat}^n_{c=1} \left( \frac{e^{z_i} \times X_L}{\sum_{i=1}^{k} e^{z_i}} \right)_{c}, \quad i = 1, 2, \ldots, k 
\]
where $CAG$ represents the output guided by the channel-wise attention. $Z$ denotes the result of the global average pooling, which squeezes the feature map to a weight vector. $n$ denotes the number of channels of the inputted high-level feature map. $m$ is the number of pixels in each channel of the inputted feature map. $P_j$ represents the pixel values of every channel in the high-level feature map. $X_L$ is the low-level feature map. Because the combination of different-level feature maps are processed channel by channel, therefore in this paper $\text{Concat}_{c=1}^n$ denotes concatenating the processed feature map together from channel 1 to channel $n$. As shown in Eq. (7), the weight matrix makes the model focus on important features.

$$SAG (X_L, X_H) = \text{Concat}_{c=1}^n \left[ \frac{1}{1 + e^{-\text{FCN}(X_L)}} \right] c \cdot X_H$$  \hspace{1cm} (7)$$

where $SAG$ denotes the output guided by the spatial-wise attention and $\text{FCN}$ is a fully convolutional network. Similarly, $X_H$ is the high-level feature map. Since the features in the low-level feature map are sensitive to position, we utilize it to influence the high-level feature map for better feature selection. Eventually, the final attention fusion is obtained by combining the output guided by the channel-wise attention and the spatial-wise attention as shown in Eq. (8).

$$\text{AFFM} = \text{Conv}_{1 \times 1} \left( \text{Concat} (CAG, SAG) \right) + X_H$$ \hspace{1cm} (8)$$

where $\text{AFFM}$ represents the attention fusion feature map guided by two kinds of attention, and $\text{Conv}_{1 \times 1}$ denotes a convolutional layer with $1 \times 1$ kernel size to adjust the dimension.

D. GUIDED ANCHORING REGION PROPOSAL NETWORK

The generation and selection of anchor strongly affect the performance of ODN. For character segmentation, the text in an image is usually centralized in a certain area. If the anchor is generated by the sliding window method as the traditional RPN, a large number of dense anchors will cause the positive and negative samples unbalanced and increase the computational cost. Moreover, the configuration of anchor scale and aspect ratio is set manually in RPN, which is not a flexible way to generate adaptive anchors. Especially when the shape of the character is abnormal, it is prone to neglect the small-scale characters.

In order to relieve this problem, TSER uses a novel guided anchoring method that is inspired by Wang’s approach. Our guided anchoring approach creatively using neural networks to predict the center point and shape of anchors. These two
steps are independent of each other. Therefore, this process can be divided into anchor location prediction and anchor shape prediction. We define \((x, y, w, h)\) as an anchor. \((x, y)\) represents the coordinate of the center point, and \((w, h)\) represents the width and height of anchors. The distribution of all the anchors can be defined as follow:

\[
P(x, y, w, h|I) = P(x, y|I) \cdot P(w, h|x, y, I)
\] (9)

The anchor location prediction uses a probability map of the same size as the input feature map to decide where the center points of the candidate anchors may locate. This module can be easily implemented via a \(1 \times 1\) convolution kernel and a sigmoid function. The anchor shape prediction obtains the most appropriate \((w, h)\) by maximizing the generalized intersection over union (GIOU) between anchors and ground truth box, as shown in Eq. (10):

\[
(w, h) = \arg\max_{w>0, h>0} (\text{GIOU} (\text{anchor}, \text{gt}))
\] (10)

However, the values of \(w\) and \(h\) are always too large and may cause gradient explosion. So, the best choice is to rescale the predicted values to \([-1, 1]\) as following Eq. (11):

\[
w = \sigma \cdot s \cdot e^{dw}, \: h = \sigma \cdot s \cdot e^{dh}
\] (11)

where \(\sigma\) is an empirical scale factor and \(s\) is stride of the feature map. \(dw\) and \(dh\) are the predicted values of \(w\) and \(h\). This nonlinear transformation can project the output value from \([0, 1000]\) to \([-1, 1]\). This module is implemented by a \(1 \times 1\) convolution kernel, too. After the prediction of anchor location and shape, TSER uses a deformable convolutional layer with \(3 \times 3\) kernel to make the feature map more adaptive to the various features. The architecture of the whole GA-RPN is shown in Figure 5 and some comparative results of RPN and GA-RPN are shown in Figure 7. As shown in Figure 7, a large number of anchor boxes are generated by traditional RPN but most boxes fail to surround characters well. These redundant boxes increase much computational cost. By contrast, the anchor boxes generated by guided anchoring RPN (GA-RPN) are sparse and accurate. Almost all boxes are generated near the corresponding characters and the scales and aspect ratio are more appropriate. Therefore, TSER with GA-RPN can locate characters faster and more accurate than other two-stage ISNs with traditional RPN.

### E. CASCADED EDGE REFINEMENT

Since there is only one character in a ROI, contour segmentation can also be regarded as semantic segmentation. We find that the contours predicted by FCN cannot match the corresponding ground truth nicely because the boundary of character is usually ambiguous. To better distinguish characters from background, we propose a cascaded edge refinement network to refine the results during the segmentation process, making the predicted contours more conformable to the characters.

This module is composed of several refining blocks with residual structure. Each block similarly applies attention mechanism to distinguish whether this area belongs to a character or not. In Figure 6, the input of this module is a set of feature maps and the output is the corresponding contour. Eq. (13) describes the calculation in every refining block.

\[
y_n = \sum_{i=1}^{D} w_{i,j} x_{i,j}
\] (12)

\[
RB = \max (0, \max (0, y_n) + x)
\] (13)
where $x$ is the input feature map and $w$ represents the trainable weight matrix. $n$ denotes the number of channels and $n \in 1, 2, \ldots, N$. $RB$ denotes the output of refining block. This module can be cascaded and the number of cascaded units will influence the performance. In this paper, we conclude that the performance is improved with the increasing of units at the beginning. However, if the unit is cascaded excessively, the performance tends to be stable while the number of parameters become much larger. Thus in our experiment, we set the number of cascaded units as 5 after balancing the performance and efficiency.

**F. OTHER REMEDIES**

1) **GIOU** [35]

Usually, intersection over union (IOU) is the first choice to evaluate overlap between the proposal box (PB) and the corresponding ground truth (GT). However, when there is no overlap between PB and GT, the IOU and the gradient would be 0, which means that the model is hard to be further optimized. Additionally, accuracy of the coverage, i.e., PB correctly covers GT, can be totally different even with the same IOU value. For example, as shown in Figure 8, it is apparent that the accuracy of the coverage decreases from left to right. Therefore, TSER uses GIOU to evaluate the overlap between PB and the corresponding GT. The definition of GIOU is Eq. (14).

$$GIOU = \left| \frac{A \cap B}{A \cup B} - \frac{C \setminus (A \cup B)}{C} \right|$$  \hspace{1cm} (14)

where $A$ and $B$ are two intersecting boxes and $C$ is the smallest enclosing box which can enclose them simultaneously. $C(A \cup B)$ denotes the area included in $C$ but excluded from $A \cup B$. The main idea is to consider both the influence of the joint area and disjoint area. GIOU can keep the immutability for scale and $GIOU \in [-1, 1]$. Especially, if two boxes don’t overlap together, the value of GIOU will be close to $-1$ instead of 0, which can avoid vanishing gradient problem to some extent.

Usually some common regression loss functions like MSE and smooth L1 loss are used. Nevertheless, these loss functions never take the influence of the IOU into consideration. Hence, we use GIOU to estimate the overlap between two boxes, and regard it as the bounding box regression loss. The pseudocode of loss calculating is shown in Algorithm 1.

**G. SOFT-NMS** [36]

Although TSER has been combined with TSAM which improves the performance to some extent, we still find that the recall of the model is not satisfactory enough. By checking the wrong cases, we observed that errors often occur when multilingual characters are adhesive together. For instance, in Figure 9, the network should output two boxes for two different characters. However, normal NMS may remove the yellow box because the IOU may surpass a certain threshold. Thus, we replace normal NMS with Soft-NMS. The main process of the Soft-NMS is defined as the following formula:

$$s_i = s_i \cdot e^{-\frac{GIOU(M, b_i)^2}{\sigma}}, \quad \forall b_i \notin D$$  \hspace{1cm} (15)
two detection results. The red box is the possible result with the normal 
which are classification loss \( L_{\text{cls}} \), regression loss \( L_{\text{reg}} \), anchor location loss \( L_{\text{LOC}} \) and anchor shape 
the whole network consists of GA-RPN loss and character 
Since TSER is a two-stage segmentation network, the loss of 
in Algorithm 2.

FIGURE 9. This image demonstrates the process of the normal NMS (non-maximum suppression). The blue box and the yellow box denote two detection results. The red box is the possible result with the normal NMS.

where \( s_i \) denotes the score of bounding boxes. This score represents the possibility of whether this box is a character. In Eq. (15), Soft-NMS introduces a penalty to the origin scores. The penalty will be zero if there is no overlap between two boxes; and it will be larger with a higher overlap. This way, the situation that neighbor boxes are mistakenly suppressed can be avoided. The pseudocode of the Soft-NMS is shown in Algorithm 2.

IV. MODEL TRAINING
Since TSER is a two-stage segmentation network, the loss of the whole network consists of GA-RPN loss and character segmentation loss. Therefore, we use joint training strategy to separately define the objective function for these two parts and train them respectively.

Loss of GA-RPN can be treated as four independent ones which are classification loss \( L_{\text{cls}} \), regression loss \( L_{\text{reg}} \), anchor location loss \( L_{\text{LOC}} \) and anchor shape loss \( L_{\text{Shape}} \). All of them are jointly optimized with the following equation:

\[
L_{\text{GA-RPN}} = L_{\text{cls}} + L_{\text{reg}} + L_{\text{LOC}} + L_{\text{Shape}} 
\] (16)

Then we map the ground truth box \((x_{gt}, y_{gt}, w_{gt}, h_{gt})\) to the different scale feature map as \((x'_{gt}, y'_{gt}, w'_{gt}, h'_{gt})\) and sample some points from different regions as the label for location prediction. In this paper, we denote \( R(x, y, w, h) \) as a rectangle region whose center point is \((x, y)\), width and height

---

**Algorithm 1** GIOU as Bounding Box Regression Loss

**Input:** Predicted box \( B^p = (x^p_1, y^p_1, x^p_2, y^p_2) \) and ground truth box \( B^g = (x^g_1, y^g_1, x^g_2, y^g_2) \)

**Output:** \( L_{\text{GIOU}}(L_{\text{reg}}) \)

1. For the predicted box \( B^p \), ensuring \( x^p_2 > x^p_1 \) and \( y^p_2 > y^p_1 \):
   \[
   \hat{x}^p_1 = \min(x^p_1, x^p_2), \quad \hat{x}^p_2 = \max(x^p_1, x^p_2), \quad \hat{y}^p_1 = \min(y^p_1, y^p_2), \quad \hat{y}^p_2 = \max(y^p_1, y^p_2). 
   \]

2. Calculating area of \( B^p \): \( A^p = (\hat{x}^p_2 - \hat{x}^p_1) \times (\hat{y}^p_2 - \hat{y}^p_1) \)

3. Calculating area of \( B^g \): \( A^g = (\hat{x}^g_2 - \hat{x}^g_1) \times (\hat{y}^g_2 - \hat{y}^g_1) \)

4. Calculating intersection \( I \) between \( B^p \) and \( B^g \):
   \[
   x^*_1 = \max(\hat{x}^p_1, \hat{x}^g_1), \quad x^*_2 = \min(\hat{x}^p_2, \hat{x}^g_2), \quad y^*_1 = \max(\hat{y}^p_1, \hat{y}^g_1), \quad y^*_2 = \min(\hat{y}^p_2, \hat{y}^g_2). 
   \]
   \[
   I = \begin{cases} 
   (x^*_2 - x^*_1) \times (y^*_2 - y^*_1) & \text{if } x^*_2 > x^*_1, y^*_2 > y^*_1 \\
   0 & \text{otherwise} 
   \end{cases} 
   \]

5. Finding the coordinate of the smallest enclosing box \( B^c \):
   \[
   x^c_1 = \min(\hat{x}^p_1, \hat{x}^g_1), \quad x^c_2 = \max(\hat{x}^p_2, \hat{x}^g_2), \quad y^c_1 = \min(\hat{y}^p_1, \hat{y}^g_1), \quad y^c_2 = \max(\hat{y}^p_2, \hat{y}^g_2). 
   \]

6. Calculating area of \( B^c \): \( A^c = (x^c_2 - x^c_1) \times (y^c_2 - y^c_1) \)

7. GIOU = \( \frac{I}{A^p + A^g - I} \), where \( U = A^p + A^g - I \)

8. \( L_{\text{GIOU}} = 1 - \text{GIOU} \)

**Algorithm 2** Soft-NMS

**Input:** \( B = \{B_1, \ldots, B_N\}, S = \{S_1, \ldots, S_N\}, N_t \)

\( B \) is the list of initial detection boxes.

\( S \) contains corresponding detection scores

\( N_t \) is the NMS threshold

**Output:** Proposals \( D \) and corresponding scores \( S \)

1. \( D \leftarrow \{\} \)

2. while \( B \neq \text{empty} \) do

3. \( m \leftarrow \text{argmax} \ S \)

4. \( M \leftarrow B_m \)

5. \( D \leftarrow D \cup M ; B \leftarrow B - M \)

6. for \( b_i \) in \( B \) do

7. \( S_i \leftarrow S_i \cdot \frac{e^{-\text{GIOU}(M,b_i)^2}}{s}, \forall b_i \notin D \)

8. end

9. end

10. return \( D, S \)
are \((w, h)\). Each box can be separated into center region, ignore region or outside region. These regions can be defined as the following formula:

\[
CR = R(x'^{\prime}_{gt}, y'^{\prime}_{gt}, \sigma_1 w'_{gt}, \sigma_1 h'_{gt}) \quad 0 < \sigma_1 < 1 \tag{17}
\]

\[
IR = R(x^{\prime}_{gt}, y^{\prime}_{gt}, \sigma_2 w'_{gt}, \sigma_2 h'_{gt}) - CR \quad \sigma_1 < \sigma_2 < 1 \tag{18}
\]

where \(CR\) denotes the center area of an anchor. \(IR\) denotes the ignore region that is useless for training process. The outside region \(OR\) is the rest region excluding \(IR\) and \(CR\). We randomly pick points from center region as the positive samples, and ones from outside region as the negative samples. 

\[
\text{min} L\left(u, u^{\prime}\right) = \begin{cases}\gamma & \text{if } u \neq u^{\prime} \\ \log u' & \text{otherwise} \end{cases}
\]

\[
\text{max} L\left(u, u^{\prime}\right) = \begin{cases}\gamma & \text{if } u \neq u^{\prime} \\ \log (1 - u') & \text{otherwise} \end{cases}
\]

are shown in Figure 11. We balance the ratio of training set, colors. Some sample text images and data annotation images are shown in Figure 12. The font in each image is randomly selected from seven font types. Moreover, every image is accompanied by random degrees of Gaussian blur and different background colors. Some sample text images and data annotation images are shown in Figure 11. We balance the ratio of training set, validation set and test set to 4:1:2 as shown in Figure 10. For the convenience of description, we name this dataset as the Arti-Text, which is used for optical character segmentation task.

\[
L_{cls} \text{ and } L_{mask}. \text{ Since the classification in this section is just to classify character or background, simple binary cross-entropy is enough to evaluate its loss. It is shown in Eq. (19). Similarly, the bounding box regression in this section is also the same as the regression in GA-RPN, so the definition of } L_{reg} \text{ is shown in Eq. (24).}
\]

Semantic segmentation is employed to get the contours of every character. During the experiment, we observe that for each bounding box detected by this network, the number of positive and negative pixels are unbalanced. Therefore, directly using cross-entropy as loss function will lead to a poor segmentation performance. In this paper, we apply the focal loss proposed by Liu et al. [30] to tackle this problem:

\[
L(u, u^{\prime}) = \begin{cases} -\alpha (1 - u')^\gamma \log u' & u = 1 \\ (1 - \alpha) u'^\gamma \log (1 - u') & u = 0 \end{cases}
\]

\[
L_{mask} = \frac{1}{n} \sum_{i=1}^{n} L(u_i, u_i^{\prime})
\]

where \(u\) represents the pixel-wise classification result and \(u^{\prime}\) denotes the corresponding ground truth box. Both of their values are either 0 or 1. \(n\) is the number of predicted masks and each instance corresponds to one mask.

According to the above loss functions, the character segmentation network can be jointly optimized by the following function:

\[
L_{seg} = L_{cls} + L_{reg} + L_{mask}
\]

V. EXPERIMENTS

A. EXPERIMENT SETTINGS

To prove the performance and generalization capacity of TSER, we choose two different datasets, Arti-Text and CASIA-HWDB to evaluate the performance of optimal character segmentation and handwritten character segmentation.

1) ARTI-TEXT

To verify TSER’s performance on mixed characters, since such data sets are particularly rare, we create a text character segmentation dataset with mixed characters and complicated cases. This dataset is generated by open-cv toolkit in conjunction with a character generation algorithm. The number of characters in each image ranges from 5 to 15. All characters are randomly extracted from 6583 characters (including Chinese, English, punctuation, numbers, etc.) and affixed into the image with unequal spacing and random deflection. The font in each image is randomly selected from seven font types. Moreover, every image is accompanied by random degrees of Gaussian blur and different background colors. Some sample text images and data annotation images are shown in Figure 11. We balance the ratio of training set, validation set and test set to 4:1:2 as shown in Figure 10. For the convenience of description, we name this dataset as the Arti-Text, which is used for optical character segmentation task.
TABLE 1. Comparative results on the Arti-Text. The subscript 50 and 75 mean the value of the iou threshold.

| Method           | P (%) | P_{50} (%) | P_{75} (%) | R (%) | R_{50} (%) | R_{75} (%) |
|------------------|-------|------------|------------|-------|------------|------------|
| Traditional      |       |            |            |       |            |            |
| Projection       | 40.9  | -          | -          | 37.8  | -          | -          |
| Water droplet [37]| 44.1  | -          | -          | 39.6  | -          | -          |
| Clustering [38]  | 37.6  | -          | -          | 39.3  | -          | -          |
| Deep learning    |       |            |            |       |            |            |
| YOLO v2          | 44.7  | 72.1       | 55.6       | 50.3  | 71.6       | 57.3       |
| SSD              | 53.6  | 80.9       | 65.5       | 59.6  | 80.7       | 66.9       |
| RetinaNet        | 62.3  | 91.7       | 76.4       | 67.8  | 91.8       | 76.6       |
| FCIS             | 55.6  | 87.3       | 66.9       | 62.3  | 88.5       | 73.5       |
| Mask R-CNN       | 65.0  | 96.3       | 79.9       | 71.1  | 96.9       | 81.4       |
| Ours             |       |            |            |       |            |            |
| TSER             | 68.8  | 98.2       | 84.7       | 71.8  | 95.9       | 81.0       |
| TSER+OHem        | 69.9  | 98.6       | 85.6       | 72.1  | 96.9       | 82.4       |

FIGURE 10. The data distribution histogram of the Arti-Text. The ratio of the training set, validation set, and test set is about 4:1:2.

FIGURE 11. Some examples of the generated text image and corresponding annotation image. Interference is purposely added to the images with Gaussian blur to make them more difficult to be segmented correctly.

2) CASIA-HWDB

CASIA-HWDB [39] is an offline Chinese handwriting database, which is built by the National Laboratory of Pattern Recognition (NLPR), Institute of Automation of Chinese Academy of Sciences (CASIA). The original data are not handwritten text images, but handwritten character images. Therefore, we concatenate these characters randomly to form the complete text images. Some examples of CASIA-HWDB are shown in Figure 12.

FIGURE 12. Some examples of processed CASIA-HWDB.

FIGURE 13. Comparative results on the Arti-Text. P-bar and R-bar denote precision and recall in the bar graph. P-line and R-line are precision and recall in the line graph.

FIGURE 14. Training loss and validation loss of TSER and Mask R-CNN on the Arti-Text.

3) IMPLEMENTATION DETAILS

In this work, we use ResNet101- TSAM-FPN pre-trained on ImageNet as the backbone network and FCN as the basic
TABLE 2. Comparative results on CASIA-HWDB.

| Method     | P (%) | R (%) |
|------------|-------|-------|
| Projection | 27.8  | 19.5  |
| Water droplet | 30.6  | 23.3  |
| Liu et al. [40] | 45.6  | 41.6  |
| Xu et al. [41] | 21.2  | 58.2  |
| **TSER**   | **47.1** | **53.2** |

Comparative results on CASIA-HWDB

FIGURE 15. This figure shows the precision and recall between TSER and other methods on CASIA-HWDB.

TABLE 3. The performance of TSER in different case on the Arti-Text D, A, N denote deflection, adhesion and normal.

| Character type | D     | A     | N     | P (%) | R (%) |
|----------------|-------|-------|-------|-------|-------|
| Chinese        | ✓     | ✓     | ✓     | 64.3  | 67.0  |
|                | ✓     | ✓     | ✓     | 63.9  | 63.4  |
|                | ✓     | ✓     | ✓     | **81.5** | **83.9** |
| Non-Chinese    | ✓     | ✓     | ✓     | 62.3  | 64.0  |
|                | ✓     | ✓     | ✓     | 57.9  | 60.4  |
|                | ✓     | ✓     | ✓     | **75.4** | **78.9** |
| Mixed          | ✓     | ✓     | ✓     | 58.8  | 66.2  |
|                | ✓     | ✓     | ✓     | 54.1  | 57.3  |
|                | ✓     | ✓     | ✓     | **71.5** | **74.7** |

TABLE 4. The influence of each module in TSER on the Arti-Text AF, GR, ER denote TSAM, GA-RPN and edge refinement.

| AF | GR | ER | P (%) | R (%) |
|----|----|----|-------|-------|
| ✓  | ✓  | ✓  | 66.3  | 69.8  |
| ✓  | ✓  | ✓  | 65.7  | 71.2  |
| ✓  | ✓  | ✓  | 66.7  | 68.9  |
| ✓  | ✓  | ✓  | 66.3  | 72.7  |
| ✓  | ✓  | ✓  | 68.3  | 72.9  |
| ✓  | ✓  | ✓  | **69.1** | **72.5** |

TABLE 5. Result with different combinations of operation OP₁ and OP₂ in TSAM. OP₁ and OP₂ denote two operations pointed by two red arrows in Figure 3.

| OP₁    | OP₂     | P (%) | R (%) |
|--------|---------|-------|-------|
| Add    | Add     | 63.7  | 68.9  |
| Add    | Concatenate | 66.3  | 70.1  |
| Concatenate | Concatenate | 62.5  | 70.3  |
| Concatenate | Add | 68.8  | 71.7  |

FIGURE 16. This figure shows how the edge refinement network influences character segmentation task.

TABLE 5. Result with different combinations of operation OP₁ and OP₂ in TSAM. OP₁ and OP₂ denote two operations pointed by two red arrows in Figure 3.

B. COMPARATIVE RESULTS

In this section, we present quantitative results. Firstly, we compare TSER with traditional methods, popular ODNs and instead of the performance improvement brought by different backbone networks. To make the convergence faster, we pre-train the backbone network with TSAM on the Arti-Text and CASIA-HWDB for character recognition task. The input image is resized to 1024 × 800 without any other preprocess. We set σ₁ = 0.2, σ₂ = 0.5 in GA-RPN and λ₁ = 1, λ₂ = 0.1 in the loss function. As for the focal loss parameters, we set α = 0.25, γ = 2.

We train TSER by Adam optimizer over 1 GPU with 2 images per batch and set β₁ = 0.9, β₂ = 0.999, ε = 10⁻⁸ in Adam. The model has been trained for 30 epochs and 2000 iterations per epoch in total with an initial learning rate of 0.0001 and the decay rate of 0.004 on GTX2080Ti.
ISNs on the Arti-Text and evaluate the performance by precision (P) and recall (R). What’s more, we define Mask R-CNN as the baseline model on which the architecture of TSER is built. In Table 1, we observe that P_{50}(R_{50}) is much larger than P_{75}(R_{75}). It means that with the increasing of the IOU threshold, correctly and completely locating every character becomes trickier. As a result, it is inappropriate to merely choose the P or R of some certain IOU threshold as the key metric. In this paper, we select P and R (range from 0.5~0.75) as the decisive factor to evaluate all the methods.

As shown in Table 1, it is apparent that TSER outperforms other deep learning models and rule-based methods by a certain margin. Particularly, for Mask R-CNN, TSER improves P by 3.8% and R by 0.7% respectively. In addition, we replace the original backbone network with ResNeXt101-TSAM-FPN and use OHEM to obtain a better model than the base TSER. From Table 1, the performance of the improved TSER outperforms the base TSER about 1.1% of P and 0.3% of R. Figure 13 intuitively shows the performance comparison for TSER and other approaches. Except for Mask R-CNN, TSER far exceeds other methods. To further explore the difference between TSER and Mask R-CNN, we present the loss curve of them, which is shown in Figure 14. The descending tendency of TSER is faster and more obvious than that of Mask R-CNN, indicating TSER is easier to converge to the optimal.

Secondly, to evaluate the model performance in handwritten character segmentation task, we compare TSER with other common methods on CASIA-HWDB. As shown in Table 2, TSER brings about 1.5 points higher P compared with Liu et al. Although its R is 5% lower than Xu et al., TSER still performs efficiently on handwriting database because its P and R are balanced. This experimental result demonstrates that TSER performs well in handwritten character segmentation, which can be intuitively observed from Figure 15.

Additionally, we test the performance of TSER in different cases on the Arti-Text. As shown in Table 3, our TSER performs well if the data have slight deflection and adhesion. Nevertheless, the performance of TSER will decrease obviously when characters are adhesive seriously or have a certain deflection angle. Especially, various character types further increase the difficulty of segmentation. We observe that TSER can achieve P of 81.5% and R of 83.9% in the best condition (pure Chinese character without any interference) but P of 58.8% and R of 66.2% in the worst condition (mixed character with interference). The apparent disparity shows that mixed character with interference is the trickiest case in character segmentation.

C. ABLATION STUDY

In this section, we dismantle different components in TSER to investigate the influence of each component, including TSAM, GA-RPN and edge refinement network. From Table 4, it is apparent that TSAM and GA-RPN contributes greatly to the P and R of TSER, which leads to an improvement of 2% to 6% than the baseline model. In addition, the impact of edge refinement network seems to be smaller than other modules. The reason is that it’s designed for cases that characters are so adhesive that their bounding boxes have overlapping areas, and these cases are not common in all the datasets. We show such a case in Figure 16, by obtaining the refined contours, characters can be better segmented.

In TSER, TSAM is introduced to improve the feature extraction ability and enhance the semantic boundary. From Table 4, our attention mechanism has a great influence on both character location and character segmentation. We also consider whether the operations pointed by two red arrows in Figure 3 can be replaced by other operations. As shown in Table 5, we try four possible combinations and find that the last combination is the most appropriate one. In order to intuitively understand how the attention in TSAM works, we visualize the middle result of final layer in Figure 17. It can be seen that TSAM guides the attention concentrate on boundary, shape and location of characters. Thus, TSAM enhances feature expression ability of model, which helps TSER find out accurate location of characters.
FIGURE 20. Some comparative results between TSER and the baseline model (Mask R-CNN). The top row is the original image, the middle row is the result of the baseline model and the bottom row is the result of our TSER.

Due to the introduction of GA-RPN, it’s important to decide the number of generated anchors because it directly affects the amount of computation and decides the efficiency of the whole network. We use $\epsilon$ to denote the location threshold and those whose prediction scores are less than this threshold will be discarded as negative samples. As shown in Table 6, as the threshold increases, the number of generated anchors goes down rapidly. This makes the recall suffer a bit decrease but the computational cost also declines sharply. After balancing the performance and efficiency, we consider that it is worthy to promote the whole efficiency at the cost of some performance. Therefore we eventually set $\epsilon$ as 0.01.

TABLE 6. The influence caused by different $\epsilon$ $\epsilon$ denotes the location threshold and “anchors” is the corresponding number of generated anchors under this $\epsilon$.

| $\epsilon$ | anchors | R (%) | Time(s) |
|------------|---------|-------|---------|
| 0          | 35100(100%) | 72.3  | 6.6     |
| 0.01       | 12983(36.9%) | 72.0  | 6.3     |
| 0.05       | 3047(8.7%) | 71.8  | 6.2     |
| 0.1        | 1812(5.2%) | 71.1  | 6.2     |

In Figure 18, we compare GA-RPN in TSER with other cases under 200, 500, 1000 proposals. Figure 18 shows that the model with GA-RPN behaves better than the one with traditional RPN, which proves the efficiency of GA-RPN. Moreover, more proposals will bring a better recall. But we don’t need too many proposals because GA-RPN in TSER performs well enough and the extra computational cost increased by excessive proposals is a great burden. So, in this experiment, we set the number of generated proposals as 500.

The cascaded edge refinement network is a semantic segmentation network which generates contours that fit the corresponding characters. Coincidentally, we observe that cascading the unit can improve the quality of generated character contours. In this module, we define $\delta$ to denote the cascaded frequency of the unit which is shown in Figure 6. According to the result in Figure 19, as the $\delta$ increases, the performance of segmentation becomes better. However, we still set $\delta$ to 5, because the extra computational expense caused by increasing $\delta$ is unworthy. To prove that the remedies applied in TSER is meaningful, we test the performance of TSER in different combination, NMS-IOU, NMS-GIOU, Soft-NMS-IOU and Soft-NMS-GIOU. From Table 7, it is apparent that Soft-NMS-GIOU does helped to the increasing of $P$ and $R$.

TABLE 7. The result caused by different combinations of NMS, SOFT-NMS, IOU AND GIOU on TSER.

| NMS  | Soft-NMS | IOU | GIOU | P (%) | R (%) |
|------|----------|-----|------|-------|-------|
| ✔    | ✔        | ✔   | ✔    | 66.4  | 71.3  |
| ✔    | ✔        | ✔   | ✔    | 66.6  | 71.7  |
| ✔    | ✔        | ✔   | ✔    | 67.9  | 71.8  |
| ✔    | ✔        | ✔   | ✔    | 68.8  | 71.8  |

D. TIME COST

1) TRAINING

TSER does not require too much training data and the speed of the training process is very fast. Pre-training ResNet101-FPN-TSAM on the Arti-Text and CASIA-HWDB takes approximately 32 hours. Training TSER takes 27 hours on the Arti-Text (about 1.5~2 hours per epoch).

2) INFERENCE

We use the ResNet101-FPN-TSAM model to extract features, apply GA-RPN to generate proposals, and finally adopt the character segmentation network to segment contours.
of characters. This model runs at 3.3s per image with the shape of 1024 × 800 on a single GTX2080Ti.

E. QUALITATIVE RESULTS
In this section, we show some qualitative results on the Arti-Text and CASIA-HWDB respectively. As shown in Figure 20, every case has 3 rows. The first row is the origin image. The second row is the result generated by the baseline model. The third row is the result generated by TSER. In the second row, many characters are missed, and the obtained contours do not fit the corresponding characters well. Yet, in the third row, almost all characters are found out and the contours are much better. We also show the results on handwritten dataset in Figure 21. It can be seen that handwritten characters can also be well segmented by TSER although under-segmentation still exists in some cases. Thus, we can conclude TSER is an effective text line segmentation method both in optimal character segmentation and handwritten character segmentation.

VI. CONCLUSION
In this paper, we proposed a novel character segmentation network TSER that aims at segmenting characters from text line in images. TSER has four major contributions: 1) A novel two-stage segmentation network focusing on character segmentation task was proposed. 2) TSER employed the two-stream attention mechanism to improve the feature expression ability. 3) Compared with other ISNs, TSER utilized the idea of GA-RPN to generate less but better anchors. 4) A cascaded edge refinement network was used to pixel-wisely segment characters more accurately. On our benchmark dataset (Arti-Text), TSER achieved about 3.8% higher precision and 1.0% higher recall with 90% fewer anchors than the baseline model. On CASIA-HWDB, TSER also outperformed other traditional and deep-learning-based methods. Experimental results proved that TSER is an effective method with strong generalization for character segmentation task. Possible future improvements may include reducing the number of parameters and simplifying the whole architecture to make it lighter.

REFERENCES
[1] S. Chang and J. Liu, “Multi-lane capsule network for classifying images with complex background,” IEEE Access, vol. 8, pp. 79876–79886, 2020, doi: 10.1109/ACCESS.2020.2990700.
[2] S. Shang, J. Liu, and Y. Yang, “Multi-layer transformer aggregation encoder for answer generation,” IEEE Access, vol. 8, pp. 90410–90419, 2020, doi: 10.1109/ACCESS.2020.2993875.
[3] J. Yan and A. S. El Ahmad, “A low-cost attack on a microsoft captcha,” in Proc. 15th ACM Conf. Comput. Commun. Secur. (CCS), 2008, pp. 543–554.
[4] V. P. Le, N. Nayef, M. Visani, J.-M. Ogier, and C. D. Tran, “Text and non-text segmentation based on connected component features,” in Proc. 13th Int. Conf. Document Anal. Recognit. (ICDAR), Aug. 2015, pp. 1096–1100.
[5] M. W. A. Kesiman, J.-C. Burie, and J.-M. Ogier, “A new scheme for text line and character segmentation from gray scale images of palm leaf manuscript,” in Proc. 15th Int. Conf. Frontiers Handwrit. Recognit. (ICFHR), Shenzhen, China, Oct. 2016, pp. 325–330, doi: 10.1109/ICFHR.2016.19368.
[6] H. Zhang, Y. Liu, and Y. Wang, “Observer-based finite-time adaptive fuzzy control for nontriangular nonlinear systems with full-state constraints,” IEEE Trans. Cybern., early access, Apr. 29, 2020, doi: 10.1109/TICYB.2020.2984791.
[7] H. Zhang, Y. Liu, J. Dai, and Y. Wang, “Command filter based adaptive fuzzy finite-time control for a class of uncertain nonlinear systems with hysteresis,” IEEE Trans. Fuzzy Syst., early access, Jun. 18, 2020, doi: 10.1109/TFUZZ.2020.3003499.
[8] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 3431–3440.
[9] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 12, pp. 2411–2426, Dec. 2017.
[10] O. Ronneberger, F. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent. Cham, Switzerland: Springer, 2015, pp. 234–241.
[11] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, “Fully convolutional instance-aware semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, Jul. 2017, pp. 4438–4446, doi: 10.1109/CVPR.2017.472.
[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Venice, Italy, Oct. 2017, pp. 2980–2988, doi: 10.1109/ICCV.2017.322.
[13] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. , Salt Lake City, UT, USA, Jun. 2018, pp. 8759–8768, doi: 10.1109/CVPR.2018.0913.
[14] T. Q. Phan, P. Shivakumara, B. Su, and C. L. Tan, “A gradient vector flow-based method for video character segmentation,” in Proc. Int. Conf. Document Anal. Recognit., Beijing, China, Sep. 2011, pp. 1024–1028, doi: 10.1109/ICDAR.2011.207.
[15] N. Sharma, P. Shivakumara, U. Pal, M. Blumenstein, and C. L. Tan, “A new method for character segmentation from multi-oriented video words,” in Proc. 12th Int. Conf. Document Anal. Recognit., Washington, DC, USA, Aug. 2013, pp. 413–417, doi: 10.1109/ICDAR.2013.90.
[16] G. Liang, P. Shivakumara, T. Lu, and C. L. Tan, “A new wavelet-Laplacian method for arbitrarily-oriented character segmentation in video text lines,” in Proc. 13th Int. Conf. Document Anal. Recognit. (ICDAR), Tunis, Tunisia, Aug. 2015, pp. 926–930, doi: 10.1109/ICDAR.2015.7333897.
[17] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, Atrous convolution, and fully connected CRFs,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 4, pp. 834–848, Apr. 2018.
[18] G. Lin, A. Milan, C. Shen, and I. Reid, “RefineNet: Multi-path refinement networks for high-resolution semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, Jul. 2017, pp. 5168–5177, doi: 10.1109/CVPR.2017.549.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Columbus, OH, USA, Jun. 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81.

[20] R. Girshick, “Fast R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Santiago, Chile, Dec. 2015, pp. 1440–1448, doi: 10.1109/ICCV.2015.169.

[21] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.

[22] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, “A unified multi-scale deep convolutional neural network for fast object detection,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2016, pp. 354–370.

[23] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, Jul. 2017, pp. 936–944, doi: 10.1109/CVPR.2017.106.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2016, pp. 21–37.

[25] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Long Beach, CA, USA, Jun. 2017, pp. 6517–6525, doi: 10.1109/ICCV.2017.690.

[26] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 2, pp. 318–327, Feb. 2020, doi: 10.1109/TPAMI.2018.2858826.

[27] X. Chen, R. Girshick, K. He, and P. Dollar, “TensorMask: A foundation for dense object segmentation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Seoul, South Korea, Oct. 2019, pp. 2061–2069, doi: 10.1109/ICCV.2019.00015.

[28] Y. Yang, X. Wang, Q. Zhao, and T. Sui, “Two-level attentions and grouping attention convolutional network for fine-grained image classification,” Appl. Sci., vol. 9, no. 9, p. 1939, May 2019.

[29] Y. Yang, C. Xu, F. Dong, and X. Wang, “A new multi-scale convolutional model based on multiple attention for image classification,” Appl. Sci., vol. 10, no. 1, p. 101, Dec. 2019.

[30] J. Liu et al., “Attention-based BiGRU-CNN for Chinese question classification,” J. Ambient Intell. Hum. Comput., 2019, doi: 10.1007/s12652-019-01944-9.

[31] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Salt Lake City, UT, USA, Jun. 2018, pp. 7794–7803, doi: 10.1109/CVPR.2018.00813.

[32] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, “Squeeze-and-excitation networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 8, pp. 2011–2023, Aug. 2020, doi: 10.1109/TPAMI.2019.2913372.

[33] Z. Huang, X. Wang, Y. Wei, L. Huang, H. Shi, W. Liu, and T. S. Huang, “CCNet: Criss-cross attention for semantic segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., early access, Jul. 3, 2020, doi: 10.1109/TPAMI.2020.3007032.

[34] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin, “Region proposal by guided anchoring,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Long Beach, CA, USA, Jun. 2019, pp. 2960–2969, doi: 10.1109/CVPR.2019.00308.

[35] H. Rezaftoghi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized intersection over union: A metric and a loss for bounding box regression,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 658–666.

[36] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, “Soft-NMS—Improving object detection with one line of code,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Venice, Italy, Oct. 2017, pp. 5562–5570, doi: 10.1109/ICCV.2017.593.

[37] G. Congedo, G. Dimmuro, S. Impedovo, and G. Pirlo, “Segmentation of numeric strings,” Proc. 3rd Int. Conf. Document Anal. Recognit., Montreal, QC, Canada, vol. 2, 1995, pp. 1038–1041, doi: 10.1109/ICDAR.1995.602080.

[38] L. O’Gorman, “The document spectrum for page layout analysis,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 15, no. 11, pp. 1162–1173, Nov. 1993.

[39] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang, “CASIA online and offline Chinese handwriting databases,” in Proc. Int. Conf. Document Anal. Recognit., Beijing, China, Sep. 2011, pp. 37–41, doi: 10.1109/ICDAR.2011.17.

[40] C.-L. Liu, M. Koga, and H. Fujisawa, “Lexicon-driven segmentation and recognition of handwritten character strings for Japanese address reading,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 11, pp. 1425–1437, Nov. 2002.

[41] L. Xu, F. Yin, Q. Wang, and C. Liu, “Touching character separation in Chinese handwriting using visibility-based foreground analysis,” in Proc. Int. Conf. Document Anal. Recognit., Beijing, China, 2011, pp. 859–863, doi: 10.1109/ICDAR.2011.176.

JIN YING MING ZHANG received the B.E. degree from Shanghai Maritime University, in 2019, where he is currently pursuing the M.E. degree. His current research interests include deep learning, computer vision, and natural language processing.

JIN LIU (Member, IEEE) received the B.S. degree from Lanzhou University, the M.S. degree from the University of Electrical Science and Technology of China, and the Ph.D. degree from Washington State University. He is currently a Professor with Shanghai Maritime University. His research interests include deep learning, nature language processing, and computer vision. He is a member of CAAI and CCF.

XIONGWEI XU received the ED.D. degree from Shanghai Normal University, in 2015. He is currently a Professor with the College of Education, Shanghai Normal University. His research interests include ICT uses in education, higher education management, and related topics.

PEIZHU GONG received the B.E. degree from Shanghai Maritime University, in 2019, where he is currently pursuing the M.E. degree. His current research interests include machine learning, computer vision, and natural language processing.

MINGYANG DUAN received the B.E. degree from Shanghai Maritime University, in 2019, where he is currently pursuing the M.E. degree. His current research interests include data mining, machine learning, and natural language processing.

***