Shape Robust Text Detection with Progressive Scale Expansion Network

Wenhai Wang\textsuperscript{1,4*}, Enze Xie\textsuperscript{2,5*}, Xiang Li\textsuperscript{3,4†}, Wenbo Hou\textsuperscript{1}, Tong Lu\textsuperscript{1‡}, Gang Yu\textsuperscript{5}, Shuai Shao\textsuperscript{5}

\textsuperscript{1}National Key Lab for Novel Software Technology, Nanjing University
\textsuperscript{2}Department of Computer Science and Technology, Tongji University
\textsuperscript{3}School of Computer and Engineering, Nanjing University of Science and Technology
\textsuperscript{4}Momenta
\textsuperscript{5}Megvii (Face++) Technology Inc.

\{wangwenhai362, Johnny.ez, lysicsuo\}@163.com, xiang.li.implus@qq.com
lutong@nju.edu.cn, \{yugang, shaoshuai\}@megvii.com

Abstract

Scene text detection has witnessed rapid progress especially with the recent development of convolutional neural networks. However, there still exists two challenges which prevent the algorithm into industry applications. On the one hand, most of the state-of-art algorithms require quadrangle bounding box which is inaccurate to locate the texts with arbitrary shape. On the other hand, two text instances which are close to each other may lead to a false detection which covers both instances. Traditionally, the segmentation-based approach can relieve the first problem but usually fail to solve the second challenge. To address these two challenges, in this paper, we propose a novel Progressive Scale Expansion Network (PSENet), which can precisely detect text instances with arbitrary shapes. More specifically, PSENet generates the different scale of kernels for each text instance, and gradually expands the minimal scale kernel to the text instance with the complete shape. Due to the fact that there are large geometrical margins among the minimal scale kernels, our method is effective to split the close text instances, making it easier to use segmentation-based methods to detect arbitrary-shaped text instances. Extensive experiments on CTW1500, Total-Text, ICDAR 2015 and ICDAR 2017 MLT validate the effectiveness of PSENet. Notably, on CTW1500, a dataset full of long curve texts, PSENet achieves a F-measure of 74.3% at 27 FPS, and our best F-measure (82.2%) outperforms state-of-art algorithms by 6.6%. The code will be released in the future.

1. Introduction

Scene text detection in the wild is a fundamental problem with numerous applications such as scene understanding, product identification, and autonomous driving. Many progress has been made in recent years with the rapid development of Convolutional Neural Networks (CNNs) \cite{9, 14, 31}. We can roughly divide the existing CNN based algorithm into two categories: regression-based approaches and...
For the regression-based approaches [36, 42, 32, 16, 41, 23, 11, 13, 27], the text targets are usually represented in the forms of rectangles or quadrangles with certain orientations. However, the regression-based approaches fail to deal with the text instance with arbitrary shapes, e.g., the curve texts as shown in Fig. 1 (b). Segmentation-based approaches, on the other hand, locate the text instance based on pixel-level classification. However, it is difficult to separate the text instances which are close with each other. Usually, a false detection which covers all the text instances close to each other may be predicted based on the segmentation-based approach. One example is shown in Fig. 1 (c).

To address these problems, in this paper, we propose a novel kernel-based framework, namely, Progressive Scale Expansion Network (PSENet). Our PSENet has the following two benefits. First, as a segmentation-based method, PSENet performs pixel-level segmentation, which is able to precisely locate the text instance with arbitrary shape. Second, we propose a progressive scale expansion algorithm, with which the adjacent text instances can be successfully identified as shown in Fig. 1 (d). More specifically, we assign each text instance with multiple predicted segmentation areas, which are denoted as “kernels” for simplicity. Each kernel has the similar shape to the original text instance but different scales. To obtain the final detections, we adopt a progressive scale expansion algorithm based on Breadth-First-Search (BFS). Generally, there are 3 steps: 1) starting from the kernels with minimal scales (instances can be distinguished in this step); 2) expanding their areas by involving more pixels in larger kernels gradually; 3) finishing until the complete text instances (the largest kernels) are explored.

There are three potential reasons for the design of the progressive scale expansion algorithm. First, the kernels with minimal scales are quite easy to be separated as their boundaries are far away from each other. Second, the minimal scale kernels can not cover the complete areas of text instances (see Fig. 2 (b)). Therefore, it is necessary to recover the complete text instances from the minimal scale kernels. Third, the progressive scale expansion algorithm is a simple and efficient method to expand the small kernels to complete text instances, which ensures the accurate locations of text instances.

To show the effectiveness of our proposed PSENet, we conduct extensive experiments on four competitive benchmark datasets including ICDAR 2015 [17], ICDAR 2017 MLT [11], CTW1500 [24] and Total-Text [2]. Among these datasets, CTW1500 and Total-Text are explicitly designed for curve text detection. Specifically, on CTW1500, a dataset with long curve texts, we outperform state-of-the-art results by absolute 6.6%, and our real-time model achieves a comparable performance (74.3%) at 27 FPS. Furthermore, the proposed PSENet also achieves promising performance on multi-oriented and multi-lingual text datasets: ICDAR 2015 and ICDAR 2017 MLT.

2. Related Work

Scene text detection based on deep learning methods have achieved remarkable results over the past few years. A major of modern text detectors are based on CNN framework, in which scene text detection is roughly formulated as two categories: regression-based methods and segmentation-based methods.

**Regression-based methods** often based on general object detection frameworks, such Faster R-CNN [31] and SSD [22]. TextBoxes [19] modified the anchor scales and shape of convolution kernels to adjust to the various aspect ratios of the text. EAST [42] use FCN [25] to directly predict score map, rotation angle and text boxes for each pixel. RRPN [28] adopted Faster R-CNN and developed rotation proposals of RPN part to detect arbitrary oriented text. RRD [20] extracted feature maps for text classification and regression from two separately branches to better long text detection.

However, most of the regression-based methods often require complex anchor design and cumbersome multiple stages, which might require exhaustive tuning and lead to sub-optimal performance. Moreover, the above works were specially designed for multiple oriented text detection and may fall short when handling curve texts, which are actually widely distributed in real-world scenarios.

**Segmentation-based methods** are mainly inspired by fully convolutional networks (FCN) [25]. Zhang et al. [40] first adopted FCN to extract text blocks and detect character candidates from these text blocks via MSER. Yao et al. [39] formulated one text region as various properties, such as text region and orientation, then utilized FCN to predict the corresponding heatmaps. Lyu et al. [27] utilized corner localization to find suitable irregular quadrangles for text instances. PixelLink [4] separated texts lying close to each other by predicting pixel connections between differ-
ent text instances. Recently, TextSnake [26] used ordered disks to represent curve text for curve text detection. SPC-Net [38] used instance segmentation framework and utilize context information to detect text of arbitrary shape while suppressing false positives.

The above works have achieved excellent performances over several horizontal and multi-oriented text benchmarks. Similarly, most of the above approaches have not paid special attention to curve text, except for TextSnake [26]. However, TextSnake still needs time-consuming and complicated post-processing steps (Centralizing, Striding and Sliding) during inference, while our proposed Progressive Scale Expansion needs only one clean and efficient step.

3. Proposed Method

In this section, we first introduce the overall pipeline of the proposed Progressive Scale Expansion Network (PSENet). Next, we present the details of progressive scale expansion algorithm, and show how it can effectively distinguish the text instances lying closely. Further, the way of generating label and the design of loss function are introduced. At last, we describe the implementation details of PSENet.

3.1. Overall Pipeline

A high-level overview of our proposed PSENet is illustrated in Fig. 3. We use ResNet [10] as the backbone of PSENet. We concatenate low-level texture feature with high-level semantic feature. These maps are further fused in \( F \) to encode information with various receptive views. Intuitively, such fusion is very likely to facilitate the generations of the kernels with various scales. Then the feature map \( F \) is projected into \( n \) branches to produce multiple segmentation results \( S_1, S_2, \ldots, S_n \). Each \( S_i \) would be one segmentation mask for all the text instances at a certain scale. The scales of different segmentation mask are decided by

\[
F = \mathbb{C}(P_2, P_3, P_4, P_5) = P_2 \parallel \text{Up}_{x \times 2}(P_3) \parallel \text{Up}_{x \times 4}(P_4) \parallel \text{Up}_{x \times 8}(P_5),
\]

where “\( \parallel \)” refers to the concatenation and \( \text{Up}_{x \times 2}(\cdot) \), \( \text{Up}_{x \times 4}(\cdot) \), \( \text{Up}_{x \times 8}(\cdot) \) refer to 2, 4, 8 times upsampling, respectively. Subsequently, \( F \) is fed into Conv(3, 3)-BN-ReLU layers and is reduced to 256 channels. Next, it passes through \( n \) Conv(1, 1)-Up-Sigmoid layers and produces \( n \) segmentation results \( S_1, S_2, \ldots, S_n \). Here, Conv, BN, ReLU and Up refer to convolution [18], batch normalization [15], rectified linear units [6] and upsampling.

3.3. Progressive Scale Expansion Algorithm

As shown in Fig. 1 (c), it is hard for the segmentation-based method to separate the text instances that are close to each other. To solve this problem, we propose a progressive scale expansion algorithm.

Here is a vivid example (see Fig. 4) to explain the procedure of progressive scale expansion algorithm, whose cen-
The central idea is brought from the Breadth-First-Search (BFS) algorithm. In the example, we have 3 segmentation results $S = \{S_1, S_2, S_3\}$ (see Fig. 4 (a), (e), (f)). At first, based on the minimal kernels’ map $S_1$ (see Fig. 4 (a)), 4 distinct connected components $C = \{c_1, c_2, c_3, c_4\}$ can be found as initializations. The regions with different colors in Fig. 4 (b) represent these different connected components, respectively. By now we have all the text instances’ central parts (i.e., the minimal kernels) detected. Then, we progressively expand the detected kernels by merging the pixels in $S_2$, and then in $S_3$. The results of the two scale expansions are shown in Fig. 4 (c) and Fig. 4 (d), respectively. Finally, we extract the connected components which are marked with different colors in Fig. 4 (d) as the final predictions for text instances.

The procedure of scale expansion is illustrated in Fig. 4 (g). The expansion is based on Breadth-First-Search algorithm which starts from the pixels of multiple kernels and iteratively merges the adjacent text pixels. Note that there may be conflicted pixels during expansion, as shown in the red box in Fig. 4 (g). The principle to deal with the conflict in our practice is that the confusing pixel can only be merged by one single kernel on a first-come-first-served basis. Thanks to the “progressive” expansion procedure, these boundary conflicts will not affect the final detections and the performances. The detail of scale expansion algorithm is summarized in Algorithm 1. In the pseudocode, $T, P$ are the intermediate results. $Q$ is a queue. Neighbor$(\cdot)$ represents the neighbor pixels (4-ways) of $p$. GroupByLabel$(\cdot)$ is the function of grouping the intermediate results by label. “$S_i[q] = True$” means that the predicted value of pixel $q$ in $S_i$ belongs to the text part. $C$ and $E$ are used to keep the kernels before and after expansion respectively.

### Algorithm 1 Scale Expansion Algorithm

**Require:** Kernels: $C$; Segmentation Result: $S$.

**Ensure:** Scale Expanded Kernels: $E$.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{function EXPANSION}(C, S_1)$</td>
<td></td>
</tr>
<tr>
<td>1:</td>
<td>$T \leftarrow \emptyset$; $P \leftarrow \emptyset$; $Q \leftarrow \emptyset$</td>
</tr>
<tr>
<td>2:</td>
<td>for each $c_i \in C$ do</td>
</tr>
<tr>
<td>3:</td>
<td>$T \leftarrow T \cup {(p, label) \mid (p, label) \in c_i}$</td>
</tr>
<tr>
<td>4:</td>
<td>$P \leftarrow P \cup {p \mid (p, label) \in c_i}$</td>
</tr>
<tr>
<td>5:</td>
<td>$\text{Enqueue}(Q, c_i)$ // push all the elements in $c_i$ into $Q$</td>
</tr>
<tr>
<td>6:</td>
<td>end for</td>
</tr>
<tr>
<td>7:</td>
<td>while $Q \neq \emptyset$ do</td>
</tr>
<tr>
<td>8:</td>
<td>$(p, label) \leftarrow \text{Dequeue}(Q)$ // pop the first element of $Q$</td>
</tr>
<tr>
<td>9:</td>
<td>if $3q \in \text{Neighbor}(p)$ and $q \notin P$ and $S_1[q] = \text{True}$ then</td>
</tr>
<tr>
<td>10:</td>
<td>$T \leftarrow T \cup {(q, label)}; P \leftarrow P \cup {q}$</td>
</tr>
<tr>
<td>11:</td>
<td>$\text{Enqueue}(Q, (q, label))$ // push the element $(q, label)$ into $Q$</td>
</tr>
<tr>
<td>12:</td>
<td>end if</td>
</tr>
<tr>
<td>13:</td>
<td>end while</td>
</tr>
<tr>
<td>14:</td>
<td>$E = \text{GroupByLabel}(T)$</td>
</tr>
<tr>
<td>15:</td>
<td>return $E$</td>
</tr>
<tr>
<td>16:</td>
<td>end function</td>
</tr>
</tbody>
</table>

Figure 4. The procedure of progressive scale expansion algorithm. CC refers to the function of finding connected components. EX represents the scale expansion algorithm. (a), (e) and (f) refer to $S_1$, $S_2$ and $S_3$, respectively. (b) is the initial connected components. (c) and (d) is the results of expansion. (g) is the illustration of expansion. The blue and orange areas represent the kernels of different text instances. The gray grids represent the pixels need to be involved. The red box in (g) refers to the conflicted pixel.

3.4. Label Generation

As illustrated in Fig. 3, PSENet produces segmentation results (e.g. $S_1, S_2, ..., S_n$) with different kernel scales. Therefore, it requires the corresponding ground truths with different kernel scales during training. In our practice, these ground truth labels can be conducted simply and effectively by shrinking the original text instance. The polygon with blue border in Fig. 5 (b) denotes the original text instance and it corresponds to the largest segmentation label mask.
instances and the shrunk ones respectively, and $L$ the importance between $L$ as:

$$L = \lambda L_c + (1 - \lambda)L_s,$$

where $L_c$ and $L_s$ represent the losses for the complete text instances and the shrunk ones respectively, and $\lambda$ balances the importance between $L_c$ and $L_s$.

It is common that the text instances usually occupy only an extremely small region in natural images, which makes the predictions of network bias to the non-text region, when binary cross entropy [3] is used. Inspired by [29], we adopt dice coefficient in our experiment. The dice coefficient $D(S_t, G_i)$ is formulated as in Eqn. (5):

$$D(S_t, G_i) = \frac{2 \sum_{x,y} S_{i,x,y} \times G_{i,x,y}}{\sum_{x,y} S_{i,x,y}^2 + \sum_{x,y} G_{i,x,y}},$$

where $S_{i,x,y}$ and $G_{i,x,y}$ refer to the value of pixel $(x, y)$ in segmentation result $S_i$ and ground truth $G_i$, respectively.

Furthermore, there are many patterns similar to text strokes, such as fences, lattices, etc. Therefore, we adopt Online Hard Example Mining (OHEM) [34] to $L_c$ during training to better distinguish these patterns.

$L_c$ focuses on segmenting the text and non-text region. Let us consider the training mask given by OHEM as $M$, and thus $L_c$ can be formulated as Eqn. (6).

$$L_c = 1 - D(S_n \cdot M, G_n \cdot M),$$

$L_s$ is the loss for shrunk text instances. Since they are encircled by the original areas of the complete text instances, we ignore the pixels of the non-text region in the segmentation result $S_n$ to avoid a certain redundancy. Therefore, $L_s$ can be formulated as follows:

$$L_s = 1 - \sum_{i=1}^{n-1} \frac{D(S_i \cdot W, G_i \cdot W)}{n - 1},$$

where $W$ is a mask which ignores the pixels of the non-text region in $S_n$, and $S_{n,x,y}$ refers to the value of pixel $(x, y)$ in $S_n$.

4. Experiment

In this section, we first conduct ablation studies for PSENet. Then, we evaluate the proposed PSENet on four recent challenging public benchmarks: CTW1500, TotalText, ICDAR 2015 and ICDAR 2017 MLT, and compare PSENet with state-of-the-art methods.

4.1. Datasets

CTW1500 [24] is a challenging dataset for long curve text detection, which is constructed by Yuliang et al. [24]. It consists of 1000 training images and 500 testing images. Different from traditional text datasets (e.g. ICDAR 2015, ICDAR 2017 MLT), the text instances in CTW1500 are labelled by a polygon with 14 points which can describe the shape of an arbitrarily curve text.

TotalText [2] is a newly-released dataset for curve text detection. Horizontal, multi-Oriented and curve text instances are contained in TotalText. The benchmark consists of 1255 training images and 300 testing images.

ICDAR 2015 (IC15) [17] is a commonly used dataset for text detection. It contains a total of 1500 pictures, 1000 of which are used for training and the remaining are for testing. The text regions are annotated by 4 vertices of the quadrangle.

ICDAR 2017 MLT (IC17-MLT) [1] is a large scale multi-lingual text dataset, which includes 7200 training images, 1800 validation images and 9000 testing images. The
dataset is composed of complete scene images which come from 9 languages.

4.2. Implementation Details

We use the ResNet [10] pre-trained on ImageNet [5] as our backbone. All the networks are optimized by using stochastic gradient descent (SGD). We use 7200 IC17-MLT training images and 1800 IC17-MLT validation images to train the model and report the result on IC17-MLT. Note that no extra data, e.g. SynthText [7], is adopted to train IC17-MLT. We train PSENet on IC17-MLT with batch size 16 on 4 GPUs for 180K iterations. The initial learning rate is set to $1 \times 10^{-3}$, and is divided by 10 at 60K and 120K iterations.

Two training strategies are adopted in the rest of all datasets: (1) Training from scratch. (2) Fine-tuning on IC17-MLT model. When training from scratch, we train PSENet with batch size 16 on 4 GPUs for 36K iterations, and the initial learning rate is set to $1 \times 10^{-3}$ and is divided by 10 at 12K and 24K iterations. When fine-tuning on IC17-MLT model, the number of iterations is 24K, and the initial learning rate is $1 \times 10^{-4}$ which is divided by 10 at 12K iterations.

We use a weight decay of $5 \times 10^{-4}$ and a Nesterov momentum [35] of 0.99 without dampening. We adopt the weight initialization introduced by [8].

During training, we ignore the blurred text regions labeled as DO NOT CARE in all datasets. The $\lambda$ of loss balance is set to 0.7. The negative-positive ratio of OHEM is set to 3. The data augmentation for training data is listed as follows: 1) the images are rescaled with ratio {0.5, 1.0, 2.0, 3.0} randomly; 2) the images are horizontally flipped and rotated in the range $[-10^\circ, 10^\circ]$ randomly; 3) $640 \times 640$ random samples are cropped from the transformed images. For quadrangular text, we calculate the minimal area rectangle to extract the bounding boxes. For curve text dataset, the output of PSE is applied to produce the final result.

4.3. Ablation Study

Can kernels be used as the final result? The aim of kernels is to roughly locate the text instance and separate the text instances standing closely to each other. However, the minimal scale kernels can not cover the complete areas of text instances, which does harm to the text detection and recognition. In Fig. 6 (a), the F-measures of the detector used minimal scale kernel only (the dash curves) is terrible on ICDAR 2015 and CTW1500 datasets. In addition, we use a modern text recognizer CRNN [33] to recognize the text in complete text instance and kernel, and find that CRNN failed to recognize the text in the kernel (see Fig. 2). Thus, the kernel can not be used as the final detection result.

Influence of the minimal kernel scale. We study the effect of the minimal scale $m$ by setting the number of kernels $n$ to 2 and let the minimal scale $m$ vary from 1 to 0.1. The models are evaluated on ICDAR 2015 and CTW1500 two datasets. We can find from Fig. 6 (a) that the F-measures on the test sets drops when $m$ is too large or too small. Note that when setting kernel scale 1, we only use text segmentation map as the final result and without progressive scale expansion algorithm. Obviously, without PSE the baseline’s performance is unsatisfactory because the network fails to separate the text lying closely to each other. When $m$ is too large, it is hard for PSENet to separate the text instances lying closely to each other. When $m$ is too small, PSENet often splits a whole text line into different parts incorrectly and the training cannot converge very well.

Influence of the kernel numbers. We investigate the effect of the number of kernels $n$ on the performance of PSENet. Specifically, we hold the minimal scale $m$ constant and train PSENet with different number of kernels $n$. In details, we set $m$ start from 0.4 for ICDAR 2015 and 0.6 for CTW1500 and let $n$ increase from 2 to 10. The models are evaluated on ICDAR 2015 and CTW1500 datasets. Fig. 6 (b) shows the experimental results, from which we can find that with the growing of $n$, the F-measure on the test set keeps rising and begins to level off when $n \geq 5$. The advantage of multiple kernels is that it can accurately reconstruct two text instances with large gaps of size where they lying closely to each other.

Influence of the backbone. Deeper neural networks have been proven to boost the performance of large scale image classification and object detection. To better analyze the capability of proposed PSENet, we adopt ResNet as our backbone with three different depths of {50, 101, 152} and test on the large scale dataset IC17-MLT. As shown in Table 1, under the same setting, improving the depth of backbone from 50 to 152 can clearly improve the performance from 70.8% to 72.2%, with 1.4% absolute improvement.

4.4. Comparisons with State-of-the-Art Methods

Detecting Curve Text. To test the ability of curve text detection, we evaluate our method on CTW1500 and
and “4s” means the width and height of output map is 1/1 and 1/4 of the input test image. “Ext” indicates external data. Note that EAST and SegLink were not fine-tuned on Total-Text. Therefore their results are included only for reference.

Table 2. The single-scale results on CTW1500. “P”, “R” and “F” represent the precision, recall and F-measure respectively. “1s” and “4s” means the width and height of output map is 1/1 and 1/4 of the input test image. “Ext” indicates external data.

Table 3. The single-scale results on Total-Text. “P”, “R” and “F” represent the precision, recall and F-measure respectively. “1s” and “4s” means the width and height of output map is 1/1 and 1/4 of the input test image. “Ext” indicates external data.

Table 4. The single-scale results on IC15. “P”, “R” and “F” represent the precision, recall and F-measure respectively. “1s” and “4s” means the width and height of output map is 1/1 and 1/4 of the input test image. “Ext” indicates external data.

Detecting MultiLingual Text. To test the robustness of PSENet to multiple languages, we evaluate PSENet on IC17-MLT benchmark. Due to the large scale of the dataset, in order to fully exploit the potential of the PSENet, we adopt Res50 and Res152 as the backbone. We enlarge the original image by 2 times, the proposed PSENet achieve a F-measure of 72.13%, outperforming state-of-the-art methods by absolute 5.3%. In addition, we demonstrate some test examples in Fig. 7 (a), and PSENet can accurately locate the text instances with various orientations.

Detecting Oriented Text. We evaluate the proposed PSENet on the IC15 to test its ability for oriented text detection. Only ResNet50 is adopted as the backbone of PSENet. During inference, we scale the long side of input images to 2240. We compare our method with other state-of-the-art methods in Table 4. With only single scale setting, our method achieves a F-measure of 85.69%, surpassing the state of the art results by more than 3%. In addition, we demonstrate some test examples in Fig. 7 (a), and PSENet can accurately locate the text instances with various orientations.

4.5. Speed Analyze

As shown in Table 6, PSENet can fast detect curve text instance. ResNet50 and ResNet18 are adopted as the back-
bone to trade off the speed and accuracy. We specially analyze the time consumption of PSENet in different stages. When the output feature map is 1/1 of the input image, PSENet obtains the best performance, while the time consumption of PSE is more than half of the total inference time because of the larger feature map. If the size of output feature map is 1/4 of the input images, the FPS of PSENet can be boosted from 3.9 to 8.4, while the performance slightly decrease from 84.84% to 82.09%, which is shown in Table 2. We can see the time consumption of PSE is less than 1/10 of total inference time. Furthermore, when we scale the long edge of 640, the FPS is further pushed to 22 and the detector still has good performance (75.6%).

When we use ResNet 18 as the backbone, the speed of PSENet is nearly real-time (27 FPS), while the performance is still competitive. Note that the PSENet(ResNet18) does not use external data to pretrain. Combined with Table 2, we can find PSENet surpasses EAST and CTD+TLOC in both speed and performance.

All of the above experiments are tested on CTW1500 test set. We evaluate all test images and calculate the average speed. We scale the long edge of \{1280, 960, 640\} as input to test the speed. All results in Table 6 are tested by PyTorch [30] and one 1080Ti GPU.

5. Conclusion and Future Work

We propose a novel Progressive Scale Expansion Network (PSENet) to successfully detect the text instances with arbitrary shapes in the natural scene images. By gradually expanding the detected areas from small kernels to large and complete instances via multiple semantic segmentation maps, our method is robust to shapes and can easily separate those text instances which are very close or even partially intersected. The experiments on scene text detection benchmarks demonstrate the superior performance of the proposed method.

There are multiple directions to explore in the future. Firstly, we will investigate whether the expansion algorithm can be trained along with the network end-to-end. Secondly, the progressive scale expansion algorithm can be introduced to the general instance-level segmentation tasks, especially in those benchmarks with many crowded object instances. We are cleaning our codes and will release them soon.

6. Acknowledgment

This work is supported by the Natural Science Foundation of China under Grant 61672273 and Grant 61832008, the Science Foundation for Distinguished Young Scholars of Jiangsu under Grant BK20160021, and Scientific Foundation of State Grid Corporation of China (Research on Ice-wind Disaster Feature Recognition and Prediction by Few-shot Machine Learning in Transmission Lines).
References


