Efficient and Accurate Arbitrary-Shaped Text Detection with Pixel Aggregation Network

Wenhai Wang*[1], Enze Xie*[2,4], Xiaoge Song[1], Yuhang Zang[3], Wenjia Wang[2], Tong Lu†[1], Gang Yu[4], and Chunhua Shen[5]

[1]National Key Lab for Novel Software Technology, Nanjing University
[2]Tongji University
[3]University of Electronic Science and Technology of China
[4]Megvii (Face++) Technology Inc.
[5]The University of Adelaide
{wangwenhai362, Johnny_ez, sxg514}@163.com, yuhangzang@foxmail.com, wwj940312@126.com, lutong@nju.edu.cn, yugang@megvii.com, chunhua.shen@adelaide.edu.au

Abstract

Scene text detection, an important step of scene text reading systems, has witnessed rapid development with convolutional neural networks. Nonetheless, two main challenges still exist and hamper its deployment to real-world applications. The first problem is the trade-off between speed and accuracy. The second one is to model the arbitrary-shaped text instance. Recently, some methods have been proposed to tackle arbitrary-shaped text detection, but they rarely take the speed of the entire pipeline into consideration, which may fall short in practical applications. In this paper, we propose an efficient and accurate arbitrary-shaped text detector, termed Pixel Aggregation Network (PAN), which is equipped with a low computational-cost segmentation head and a learnable post-processing. More specifically, the segmentation head is made up of Feature Pyramid Enhancement Module (FPEM) and Feature Fusion Module (FFM). FPEM is a cascadable U-shaped module, which can introduce multi-level information to guide the better segmentation. FFM can gather the features given by the FPEMs of different depths into a final feature for segmentation. The learnable post-processing is implemented by Pixel Aggregation (PA), which can precisely aggregate text pixels by predicted similarity vectors. Experiments on several standard benchmarks validate the superiority of the proposed PAN. It is worth noting that our method can achieve a competitive F-measure of 79.9% at 84.2 FPS on CTW1500.

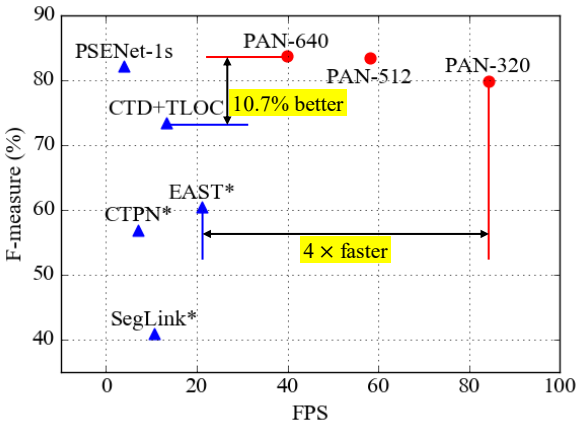

Figure 1. The performance and speed on curved text dataset CTW1500. PAN-640 is 10.7% better than CTD+TLOC, and PAN-320 is 4 times faster than EAST. * indicates the results from [31].

1. Introduction

Scene text detection is a fundamental and critical task in computer vision, as it is a key step in many text-related applications, such as text recognition, text retrieval, license plate recognition and text visual question answering. In virtue of recent development of object detection [41, 30, 11, 9, 5, 8, 57] and segmentation [1, 56, 33, 54, 6, 7] based on CNN [15, 20, 48, 25], scene text detection has witnessed great progress [47, 42, 58, 31, 35, 50, 24]. Arbitrary-shaped text detection, one of the most challenging tasks in text detection, is receiving more and more research attention Some new methods [31, 35, 50, 24] have been put forward to detect curve text instance. However, many of these methods

*Authors contributed equally.
†Corresponding author.

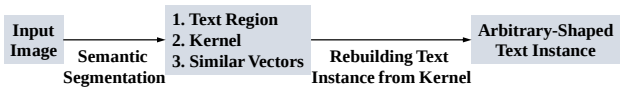

Figure 2. The overall pipeline of PAN.

suffer from low inference speed, because of their heavy models or complicated post-processing steps, which limits their deployment in the real-world environment. On the other hand, previous text detectors [58, 32] with high efficiency are mostly designed for quadrangular text instances, which have flaws when detecting curved text. Therefore, "how to design an efficient and accurate arbitrary-shaped text detector" remains largely unsolved.

To solve these problems, here we propose an arbitrary-shaped text detector, namely Pixel Aggregation Network (PAN), which can achieve a good balance between speed and performance. PAN makes arbitrary-shaped text detection following the simple pipeline as shown in Fig. 2, which only contains two steps: i) Predicting the text regions, kernels and similarity vectors by segmentation network. ii) Rebuilding complete text instances from the predicted kernels. For high efficiency, we need to reduce the time cost of these two steps. First and foremost, a lightweight backbone is required for segmentation. In this paper, we use ResNet18 [14] as the default backbone of PAN. However, the lightweight backbone is relatively weak in feature extraction, and thus its features typically have small receptive fields and weak representation capabilities. To remedy this defect, we propose a low computation-cost segmentation head, which is composed of two modules: Feature Pyramid Enhancement Module (FPEM) and Feature Fusion Module (FFM). FPEM is a U-shaped module built by separable convolutions (see Fig. 4), and therefore FPEM is able to enhance the features of different scales by fusing the low-level and high-level information with minimal computation overhead. Moreover, FPEM is cascadable, which allows us to compensate for the depth of lightweight backbone by appending FPEMs after it (see Fig. 3 (d)(e)). To gather low-level and high-level semantic information, before final segmentation, we introduce FFM to fuse the features generated by the FPEMs of different depths. In addition, to reconstruct complete text instances accurately, we propose a learnable post-processing method, namely Pixel Aggregation (PA), which can guide the text pixels to correct kernels through the predicted similarity vectors.

To show the effectiveness of our proposed PAN, we conduct extensive experiments on four challenging benchmark datasets including CTW1500 [31], Total-Text [2], ICDAR 2015 [22] and MSRA-TD500 [52]. Among these datasets, CTW1500 and Total-Text are new datasets designed for curve text detection. As shown in Fig. 1, on CTW1500, the F-measure of PAN-640 is 83.7% which is 10.7% better than CTD+TLOC [31], and the FPS of PAN-320 is 84.2 which is 4 times faster than EAST [58]. Meanwhile, PAN also has promising performance on multi-oriented and long text datasets.

In summary, our contributions are three-fold. Firstly, we propose a lightweight segmentation neck consisting of Feature Pyramid Enhancement Module (FPEM) and Feature Fusion Module (FFM) which are two high-efficiency modules that can improve the feature representation of the network. Secondly, we propose Pixel Aggregation (PA), in which the text similarity vector can be learned by the network and be used to selectively aggregate pixels nearby the text kernels. Finally, the proposed method achieves state-of-the-art performance on two curved text benchmarks while still keeping the inference speed of 58 FPS. To our knowledge, ours is the first algorithm which can detect curved text precisely in real-time.

2. Related Work

In recent years, text detectors based on deep learning have achieved remarkable results. Most of these methods can be roughly divided into two categories: anchor-based methods and anchor-free methods. Among these methods, some use a heavy framework or complicated pipeline for high accuracy, while others adopt a simple structure to maintain a good balance between speed and accuracy.

Anchor-based text detectors are usually inspired by object detectors such as Faster R-CNN [41] and SSD [41]. TextBoxes [27] directly modifies the anchor scales and shape of convolution kernels of SSD to handle text with extreme aspect ratios. TextBoxes++ [26] further regresses quadrangles instead of horizontal bounding boxes for multi-oriented text detection. RRD [28] applies rotation-invariant and sensitive features for text classification and regression from two separate branches for better long text detection. SSTD [16] generates text attention map to enhance the text region of the feature map and suppress background information, which is beneficial for tiny texts. Based on Faster R-CNN, RRPN [38] develops rotated region proposals to detect titled text. Mask Text Spotter [36] and SPCNet [50] regard text detection as an instance segmentation problem and use Mask R-CNN [12] for arbitrary text detection. The above-mentioned methods achieve remarkable results on several benchmarks. Nonetheless, most of them rely on complex anchor setting, which makes these approaches heavy-footed and prevent them from applying to real-world problems.

Anchor-free text detectors formulate text detection as a text segmentation problem, which are often built upon fully convolutional networks (FCN) [34]. Zhang et al. [55] first estimatetext blocks with FCNs and detectcharacter candidates from those text blocks with MSER. Yao et al. [53] use FCN to predict three parts of a text instance, text/non-text, character classes, and character linking orientations, then

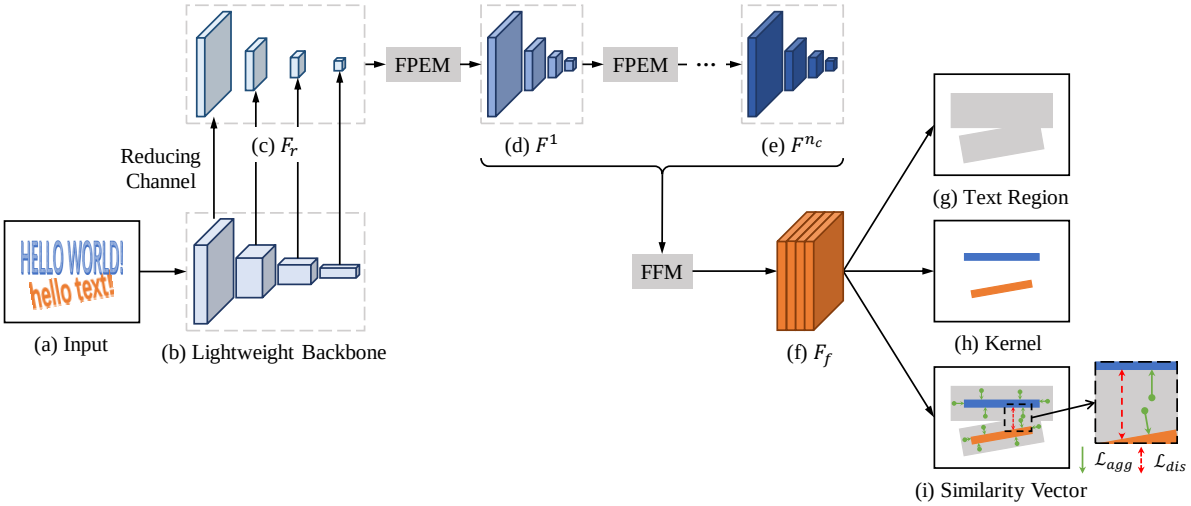

Figure 3. The overall architecture of PAN. The features from lightweight backbone network are enhanced by a low computational-cost segmentation head which is composed of Feature Pyramid Enhancement Module (FPEM) and Feature Fusion Module (FFM). The network predicts text regions, kernels and similarity vectors to describe the text instances.

apply a group process for text detection. To separate adjacent text instances, PixelLink [3] performs text/non-text and links prediction in pixel level, then applies post-processing to obtain text boxes and excludes noises. EAST [58] and DeepReg [17] adopt FCNs to predict shrinkable text score maps and perform per-pixel regression, followed by a post-processing NMS. TextSnake [35] models text instances with ordered disks and text center lines, which is able to represent text in arbitrary shapes. PSENet [24] uses FCN to predict text instances with multiple scales, then adopts progressive scale expansion algorithm to reconstruct the whole text instance. Briefly speaking, the main differences among anchor-free methods are the way of text label generation and post-processing. Nevertheless, among these methods, only TextSnake and PSENet are designed for detecting curved text instances which also widely appear in natural scenes. However, they suffer from a heavy framework or a complicated pipeline, which usually slows down their inference speed.

Real time text detection requires a fast way to generate high-quality text prediction. EAST [58] directly employs FCNs to predict the score map and corresponding coordinates and, followed by a simple NMS. The whole pipeline of EAST is concise so that it can maintain a relatively high speed. MCN [32] formulates text detection problem as a graph-based clustering problem and generates bounding boxes without using NMS, which can be fully parallelized on GPUs. However, these methods are designed for quadrangular text detection and fail to locate the curve text instances.

3. Proposed Method

3.1. Overall Architecture

PAN follows a segmentation-based pipeline (see Fig. 2) to detect arbitrary-shaped text instances. For high efficiency, the backbone of the segmentation network must be lightweight. However, the features offered by a lightweight backbone often have small receptive fields and weak representation capabilities. For this reason, we propose segmentation head that is computationally efficient to refine the features. The segmentation head contains two key modules, namely Feature Pyramid Enhancement Module (FPEM) and Feature Fusion Module (FFM). As shown in Fig. 3 (d)(e) and Fig. 4, FPEM is cascadable and has low computational cost, which can be attached behind the backbone to make features of different scales deeper and more expressive. After that, we employ the Feature Fusion Module (FFM) to fuse the features produced by the FPEMs of different depths into a final feature for segmentation. PAN predicts text regions (see Fig. 3 (g)) to describe the complete shapes of text instances, and predicts kernels (see Fig. 3 (h)) to distinguish different text instances. The network also predicts similarity vector (see Fig. 3 (i)) for each text pixel, so that the distance between the similarity vectors of pixel and kernel from the same text instance is small.

Fig. 3 shows the overall architecture of PAN. We employ a lightweight model (ResNet-18 [14]) as the backbone network of PAN. There are 4 feature maps (see Fig. 3 (b)) generated by conv2, conv3, conv4, and conv5 stages of backbone, and note that they have strides of 4, 8, 16, 32 pixels with respect to the input image. We use 1×1 convolution to reduce the channel number of each feature map to 128, and get a thin feature pyramid F_r. The feature pyra-

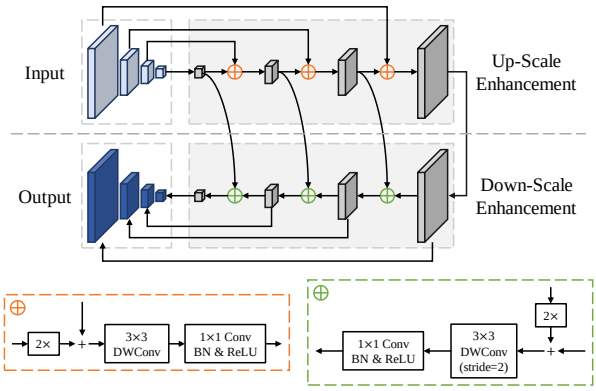

Figure 4. The details of FPEM. "+", "2×", "DWConv", "Conv" and "BN" represent element-wise addition, 2× linear upsampling, depthwise convolution [18], regular convolution [23] and Batch Normalization [21] respectively.

mid is enhanced by n_c cascaded FPEMs. Each FPEM produces an enhanced feature pyramid, and thus there are n_c enhanced feature pyramids F^1, F^2,..., F^{n_c}. FFM fuses the n_c enhanced feature pyramids into a feature map F_f, whose stride is 4 pixels and the channel number is 512. F_f is used to predict text regions, kernels and similarity vectors. Finally, we apply a simple and efficient post-processing algorithm to obtain the final text instances.

3.2. Feature Pyramid Enhancement Module

FPEM is a U-shaped module as illustrated in Fig. 4. It consists of two phases, namely, up-scale enhancement and down-scale enhancement. The up-scale enhancement acts on the input feature pyramid. In this phase, the enhancement is iteratively performed on the feature maps with strides of 32, 16, 8, 4 pixels. In the down-scale phase, the input is the feature pyramid generated by up-scale enhancement, and the enhancement is conducted from 4-stride to 32-stride.

Meanwhile, the output feature pyramid of down-scale enhancement is the final output of FPEM. We employ separable convolution [18] (3×3 depthwise convolution [18] followed by 1×1 projection) instead of the regular convolution to build the join part $\oplus$ of FPEM (see the dashed frames in Fig. 4). Therefore, FPEM is capable of enlarging the receptive field (3×3 depthwise convolution) and deepening the network (1×1 convolution) with a small computation overhead.

Similar to FPN [29], FPEM is able to enhance the features of different scales by fusing the low-level and high-level information. In addition, different from FPN, there are two other advantages of FPEM. Firstly, FPEM is a cascadable module. With the increment of cascade number n_c, the feature maps of different scales are fused more adequately and the receptive fields of features become larger. Secondly, FPEM is computationally cheap. FPEM is built by separable convolution, which needs minimal computation. The FLOPS of FPEM is about $1/5$ of FPN.

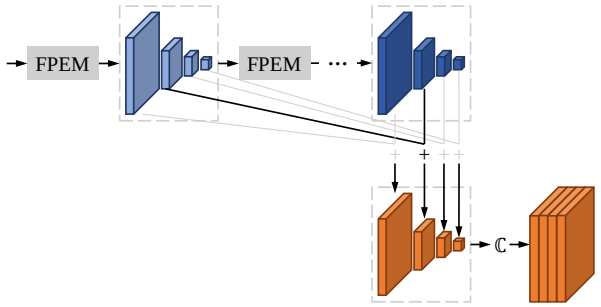

Figure 5. The detail of FFM. "+" is element-wise addition. "$\mathcal{C}$" is the operation of upsampling and concatenating.

3.3. Feature Fusion Module

Feature Fusion Module is applied to fuse the feature pyramids F^1, F^2,..., F^{n_c} of different depths. Because both low-level and high-level semantic information are important for semantic segmentation. A direct and effective method to combine these feature pyramids is to upsample and concatenate them. However, the fused feature map given by this method has a large channel number ($4\times128\times n_c$), which slows down the final prediction. Thus, we propose another fusion method as shown in Fig. 5. We firstly combine the corresponding-scale feature maps by element-wise addition. Then, the feature maps after addition are upsampled and concatenated into a final feature map which only has 4×128 channels.

3.4. Pixel Aggregation

The text regions keep the complete shape of text instances, but the text regions of the text instances lying closely are often overlapping (see Fig. 3 (g)). Contrarily, the text instances can be well distinguished using the kernels (see Fig.3 (h)). However, the kernels are not the complete text instance. To rebuild the complete text instances, we need to merge the pixels in text regions to kernels. We propose a learnable algorithm, namely Pixel Aggregation, to guide the text pixels towards correct kernels.

In Pixel Aggregation, we borrow the idea of clustering to reconstruct the complete text instances from the kernels. Let us consider the text instances as clusters. The kernels of text instances are cluster centers. The text pixels are the samples to be clustered. Naturally, to aggregate the text pixels to the corresponding kernels, the distance between the text pixel and kernel of the same text instance should be small. In the training phase, we use aggregation loss $\mathcal{L}_{agg}$ as Equ. 1 to implement this rule.

$$\mathcal{L}_{agg} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|T_i|}\sum_{p\in T_i} ln(\mathcal{D}(p, K_i) + 1), \qquad (1)$$

$$\mathcal{D}(p, K_i) = max(\|\mathcal{F}(p) - \mathcal{G}(K_i)\| - \delta_{agg}, 0)^2, \quad (2)$$

where the N is the number of text instances. The T_i is the ith text instance. $D(p, K_i)$ defines the distance between text pixel p and the kernel K_i of text instance T_i. δ_{agg} is a constant, which is set to 0.5 experimentally and used to filter easy samples. $\mathcal{F}(p)$ is the similarity vector of the pixel p. $\mathcal{G}(\cdot)$ is the similarity vector of the kernel K_i, which can be calculated by $\sum_{q \in K_i} \mathcal{F}(q) / |K_i|$.

In addition, the cluster centers need to keep discrimination. Therefore, the kernels of different text instances should maintain enough distance. We use discrimination loss $\mathcal{L}_{dis}$ as Equ. 3 to describe this rule during the training.

$$\mathcal{L}_{dis} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} ln(\mathcal{D}(K_i, K_j) + 1), \quad (3)$$

$$\mathcal{D}(K_i, K_j) = max(\delta_{dis} - \|\mathcal{G}(K_i) - \mathcal{G}(K_j)\|, 0)^2. \quad (4)$$

$\mathcal{L}_{dis}$ try to keep the distance among the kernels not less than δ_{dis} which is set to 3 in all our experiments.

In the testing phase, we use the predicted similarity vector to guide the pixels in the text area to the corresponding kernel. The detailed post-processing steps are as follows: i) Finding the connected components in the kernels' segmentation result, and each connected component is a single kernel. ii) For each kernel K_i, conditionally merging its neighbor text pixel (4-way) p in predicted text regions while the Euclidean distance of their similarity vectors is less than d. iii) Repeating step ii) until there is no eligible neighbor text pixel.

3.5. Loss Function

Our loss function can be formulated as:

$$\mathcal{L} = \mathcal{L}_{tex} + \alpha \mathcal{L}_{ker} + \beta(\mathcal{L}_{agg} + \mathcal{L}_{dis}), \quad (5)$$

where $\mathcal{L}_{tex}$ is the loss of the text regions and $\mathcal{L}_{ker}$ is the loss of the kernels. The α and β are used to balance the importance among $\mathcal{L}_{tex}$, $\mathcal{L}_{ker}$, $\mathcal{L}_{agg}$ and $\mathcal{L}_{dis}$, and we set them to 0.5 and 0.25 respectively in all experiments.

Considering the extreme imbalance of text and non-text pixels, we follow [24] and adopt dice loss [39] to supervise the segmentation result P_{tex} of the text regions and P_{ker} of the kernels. Thus $\mathcal{L}_{tex}$ and $\mathcal{L}_{ker}$ can be written as follows:

$$\mathcal{L}_{tex} = 1 - \frac{2 \sum_i P_{tex}(i) G_{tex}(i)}{\sum_i P_{tex}(i)^2 + \sum_i G_{tex}(i)^2}, \quad (6)$$

$$\mathcal{L}_{ker} = 1 - \frac{2 \sum_i P_{ker}(i) G_{ker}(i)}{\sum_i P_{ker}(i)^2 + \sum_i G_{ker}(i)^2}, \quad (7)$$

where $P_{tex}(i)$ and $G_{tex}(i)$ refer to the value of the ith pixel in the segmentation result and the ground truth of the text regions respectively. The ground truth of the text regions is a binary image, in which text pixel is 1 and non-text pixel is 0. Similarly, $P_{ker}(i)$ and $G_{ker}(i)$ means the ith pixel value in the prediction and the ground truth of the kernels. The ground truth of the kernels is generated by shrinking original ground truth polygon, and we follows the method in [24] to shrink the original polygon by ratio r. Note that, we adopt Online Hard Example Mining (OHEM) [43] to ignore simple non-text pixels when calculating L_{tex}, and we only take the text pixels in ground truth into consideration while calculating $\mathcal{L}_{ker}$, $\mathcal{L}_{agg}$ and $\mathcal{L}_{dis}$.

4. Experiment

4.1. Datasets

SynthText [10] is a large scale synthetically generated dataset containing 800K synthetic images. Following [42, 37, 35], we pre-train our model on this dataset.

CTW1500 [31] is a recent challenging dataset for curve text detection. It has 1000 training images and 500 testing images. The dataset focus on curve text instances which are labeled by 14-polygon.

Total-Text [2] is also a newly-released dataset for curve text detection. This dataset includes horizontal, multi-oriented and curve text instances and consists of 1255 training images and 300 testing images.

ICDAR 2015 (IC15) [22] is a commonly used dataset for text detection. It contains a total of 1500 images, 1000 of which are used for training and the remaining are for testing. The text instances are annotated by 4 vertices of the quadrangle.

MSRA-TD500 includes 300 training images and 200 test images with text line level annotations. It is a dataset with multi-lingual, arbitrary-oriented and long text lines. Because the training set is rather small, we follow the previous works [58, 37, 35] to include the 400 images from HUST-TR400 [51] as training data.

4.2. Implementation Details

We use the ResNet [15] or VGG16 [44] pre-trained on ImageNet [4] as our backbone. The dimension of the similarity vector is set to 4. All the networks are optimized by using stochastic gradient descent (SGD). The pre-trained model is trained on SynthText for 50K iterations with a fixed learning rate of 1×10^{-3}. Two training strategies are adopted in other experiments: i) Training from scratch. ii) Fine-tuning on SynthText pre-trained model. When training from scratch, we train PAN with batch size 16 on 4 GPUs for 36K iterations, and the initial learning rate is set to 1×10^{-3}. Similar to [56], we use the "poly" learning rate strategy in which the initial rate is multiplied by $(1 - \frac{iter}{max\_iter})^{power}$, and the $power$ is set to 0.9 in all experiments. When fine-tuning on SynthText pre-trained model,

#FPEM	GFLOPS	ICDAR 2015		CTW1500	
		F	FPS	F	FPS
0	**42.17**	78.4	**33.7**	78.8	**49.7**
1	42.92	79.9	29.5	80.4	44.7
2	43.67	80.3	26.1	81.0	39.8
3	44.43	80.4	23.0	81.3	35.2
4	45.18	**80.5**	20.1	**81.5**	32.4

Table 1. The results of models with different number of cascaded FPEMs. "#FPEM" means the number of cascaded FPEMs. "F" means F-measure. The FLOPS are calculated for the input of $640 \times 640 \times 3$.

the number of iterations is 36K, and the initial learning rate is 1×10^{-3}. We use a weight decay of 5×10^{-4} and a Nesterov momentum [45] of 0.99. We adopt the weight initialization introduced by [13].

In the training phase, we ignore the blurred text regions labeled as DO NOT CARE in all datasets. The negative-positive ratio of OHEM is set to 3. We apply random scale, random horizontal flip, random rotation and random crop on training images. On ICDAR 2015 and MSRA-TD500, we fit a minimal area rectangle for each predicted text instance. The shrink ratio r of the kernels is set to 0.5 on ICDAR 2015 and 0.7 on other datasets. In the testing phase, the distance threshold d is set to 6.

4.3. Ablation Study

To make the conclusion of ablation studies more generalized, all experiments of ablation studies are conducted on ICDAR 2015 (a quadrangle text dataset) and CTW1500 (a curve text dataset). Note that, in these experiments, all models are trained without any external dataset. The short sides of test images in ICDAR 2015 and CTW1500 are set to 736 and 640 respectively.

The influence of the number of cascaded FPEMs. We study the effect of the number of cascaded FPEMs by varying n_c from 0 to 4. Note that, when $n_c = 0$, we upsample and concatenate the feature maps in F_r to get F_f. From Table 1, we can find that the F-measures on the test sets keep rising with the growth of n_c and begins to level off when $n_c \geq 2$. However, a large n_c will slow down the model despite the low computational cost of FPEM. For each additional FPEM, the FPS will decrease by about 2-5 FPS. To keep a good balance of performance and speed, we set n_c to 2 by default in the following experiments.

The effectiveness of FPEM. We design two groups of experiments to verify the effectiveness of FPEM. Firstly, we make a comparison between the model with FPEM and without FPEM. As shown in Table 1, compared to the model without FPEM ($n_c = 0$), the model with one FPEM ($n_c = 1$) can make about 1.5% improvement on F-measure while bringing tiny extra computation. Secondly, we make comparison between a lightweight model equipped with FPEMs and a widely-used segmentation model. To ensure a fair comparison, under the same setting, we employ "ResNet18 + 2 FPEMs + FFM" or "ResNet50 + PSPNet [56]" as the segmentation network. As shown in Table. 2, even the backbone of "ResNet18 + 2 FPEMs + FFM" is lightweight, it can reach almost same performance as "ResNet50 + PSPNet [56]". In addition, "ResNet18 + 2 FPEMs + FFM" enjoy over 5 times faster speed than "ResNet50 + PSPNet [56]". The model size of "ResNet18 + 2 FPEMs + FFM" is 12.25M.

Method	ICDAR 2015		CTW1500	
	F	FPS	F	FPS
ResNet18 + 2 FPEMs + FFM	80.3	**26.1**	81.0	**39.8**
ResNet50 + PSPNet [56]	**80.5**	4.6	**81.1**	7.1

Table 2. The comparison between "ResNet18 + 2 FPEMs + FFM" with "ResNet50 + PSPNet [56]". "F" means F-measure.

#	Backbone	Fuse	PA	ICDAR 2015		CTW1500	
				F	FPS	F	FPS
1	ResNet18	FFM	✓	80.3	26.1	81.0	39.8
2	ResNet18	-	✓	79.7	**26.2**	80.2	**40.0**
3	ResNet18	Concat	✓	80.4	22.3	81.2	35.9
4	ResNet18	FFM	-	79.3	26.1	79.8	39.9
5	ResNet50	FFM	✓	81.4	16.7	**81.6**	26.0
6	VGG16	FFM	✓	**81.9**	6.6	81.5	10.1

Table 3. The results of models with different settings. "Fuse" means the fusion method. "Concat" means direct concatenation. "F" means F-measure.

The effectiveness of FFM. To investigate the effectiveness of FFM, we firstly remove FFM and concatenate the feature maps in the last feature pyramid F^{n_c} to make final segmentation. The F-measure drop 0.6%-0.8% when the FFM is removed (see Table 3 #1 and #2), which indicates that besides the features from deep layers, the shallow features are also important to semantic segmentation. We then compare FFM with the direct concatenation mentioned in Sec. 3.3. The proposed FFM can achieve performance comparable to the direct concatenation (see Table 3 #1 and #3), while FFM is more efficient.

The effectiveness of PA. We study the validity of PA by removing it from the pipeline. Specifically, we set β to 0 in Eqn. 5 in the training phase and merge all neighbor text pixels in step ii) of post-processing. Comparing the method with PA (see Table 3 #1), the F-measure of the model without PA (see Table 3 #4) drops over 1%, which indicate the effectiveness of PA.

The influence of the backbone. To better analyze the capability of the proposed PAN, we replace the lightweight backbone (ResNet18) to heavier backbone (ResNet50 and VGG16). As shown in Table 3 #5 and #6, under the same setting, both of ResNet50 and VGG16 can bring over 1% improvement on ICDAR 2015 and over 0.5% improvement on CTW1500. However, the reduction of FPS brought by the heavy backbone is apparent.

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

Curve text detection. To evaluate the performance of our method for detecting curved text instance, we compare

Method	Ext.	Venue	CTW1500			
			P	R	F	FPS
CTPN* [47]	-	ECCV'16	60.4*	53.8*	56.9*	7.14
SegLink* [42]	-	CVPR'17	42.3*	40.0*	40.8*	10.7
EAST* [58]	-	CVPR'17	78.7*	49.1*	60.4*	21.2
CTD+TLOC [31]	-	ICDAR'18	77.4	69.8	73.4	13.3
PSENet-1s [24]	-	CVPR'19	80.6	75.6	78.0	3.9
PAN-320	-	-	82.2	72.6	77.1	**84.2**
PAN-512	-	-	83.8	77.1	80.3	58.1
PAN-640	-	-	84.6	77.7	81.0	39.8
TextSnake [35]	✓	ECCV'18	67.9	85.3	75.6	-
PSENet-1s [24]	✓	CVPR'19	84.8	79.7	82.2	3.9
PAN-320	✓	-	82.7	77.4	79.9	**84.2**
PAN-512	✓	-	85.5	**81.5**	83.5	58.1
PAN-640	✓	-	**86.4**	81.2	**83.7**	39.8

Table 4. The single-scale results on CTW1500. "P", "R" and "F" represent the precision, recall and F-measure respectively. "Ext." indicates external data. * indicates the results from [31].

Method	Ext.	Venue	Total-Text			
			P	R	F	FPS
SegLink* [42]		CVPR'17	30.3*	23.8*	26.7*	-
EAST* [58]	-	CVPR'17	50.0*	36.2*	42.0*	-
DeconvNet [2]	-	ICDAR'18	33.0	40.0	36.0	-
PSENet-1s [24]	-	CVPR'19	81.8	75.1	78.3	3.9
PAN-320	-	-	84.0	71.3	77.1	**82.4**
PAN-512	-	-	86.7	78.4	82.4	57.1
PAN-640	-	-	88.0	79.4	83.5	39.6
TextSnake [35]	✓	ECCV'18	82.7	74.5	78.4	-
PSENet-1s [24]	✓	CVPR'19	84.0	78.0	80.9	3.9
SPCNet [50]	✓	AAAI'19	83.0	82.8	82.9	-
PAN-320	✓	-	85.6	75.0	79.9	**82.4**
PAN-512	✓	-	**89.4**	79.7	84.3	57.1
PAN-640	✓	-	89.3	**81.0**	**85.0**	39.6

Table 5. The single-scale results on Total-Text. "P", "R" and "F" represent the precision, recall and F-measure respectively. "Ext." indicates external data. * indicates the results from [35].

the proposed PAN with other state-of-the-art methods on CTW1500 and Total-Text which include many curve text instances. In the testing phase, we set the short side of images to different scales (320, 512, 640) and evaluate the results using the same evaluation method with [31] and [2]. We report the single-scale performance of PAN on CTW1500 and Total-Text in Table 4 and Table 5, respectively. Note that the backbone of PAN is set to ResNet18 by default.

On CTW1500, PAN-320 (the short side of input image is 320), without external data pre-training, achieve the F-measure of 77.1% at an astonishing speed (84.2 FPS), in which the F-measure surpasses most of the counterparts, including the methods with external data pre-training, and the speed is 4 times faster than the fastest method. When fine-tuning on SynthText pre-trained model, the F-measure of PAN-320 can further be boosted to 79.9%, and PAN-512 outperform all other methods in F-measure by at least 1.2% while still keeping nearly real-time speed (58 FPS).

Similar conclusions can be obtained on Total-Text. Without external data pre-training, the speed of PAN-320 is real-time (82.4 FPS) while the performance is still very competitive (77.1%), and PAN-640 achieves the F-measure of 83.5%, surpassing all other state-of-the-art methods (including those with external data) over 0.6%. With SynthText pre-training, the F-measure of PAN-320 boosting to 79.9%, and the best F-measure achieve by PAN-640 is 85.0%, which is 2.1% better than second-best SPCNet [50]. Meanwhile, the speed can still maintain nearly 40 FPS.

The performance on CTW1500 and Total-Text demonstrates the solid superiority of the proposed PAN to detect arbitrary-shaped text instances. We also illustrate several challenging results in Fig. 6 (e)(f), which clearly demonstrate that PAN can elegantly distinguish very complex curve text instances.

Oriented text detection. We evaluate PAN on the ICDAR 2015 to test its ability for oriented text detection. By default, ResNet18 is adopted as the backbone of PAN. During testing, we scale the short side of input images to 736. The comparisons with other state-of-the-art methods are shown in Table 6. PAN achieves the F-measure of 80.4% at 26.1 FPS without external data pre-training. Compared with EAST [58], our method outperforms EAST 2.1% in F-measure, while the FPS of our method is 2 times of EAST. Fine-tuning on SynthText can further improve the F-measure to 82.9% which is on par with TextSnake [35], but our method can run 25 times faster than TextSnake. Although the performance of our method is not as well as some methods (e.g. PSENet, SPCNet), our method has a least 16 times faster speed (26.1 FPS) than these methods. Some qualitative illustrations are shown in Fig. 6 (g). The

Method	Ext.	Venue	ICDAR 2015			
			P	R	F	FPS
CTPN [47]	-	ECCV'16	74.2	51.6	60.9	7.1
EAST [58]	-	CVPR'17	83.6	73.5	78.2	13.2
RRPN [38]	-	TMM'18	82.0	73.0	77.0	-
DeepReg [17]	-	ICCV'17	82.0	80.0	81.0	-
PixelLink [3]	-	AAAI'18	82.9	81.7	82.3	7.3
PAN	-	-	82.9	77.8	80.3	**26.1**
SegLink [42]	✓	CVPR'17	73.1	76.8	75.0	-
SSTD [16]	✓	ICCV'17	80.2	73.9	76.9	7.7
WordSup [19]	✓	CVPR'17	79.3	77.0	78.2	-
Lyu et al. [37]	✓	CVPR'18	**94.1**	70.7	80.7	3.6
RRD [28]	✓	CVPR'18	85.6	79.0	82.2	6.5
MCN [32]	✓	CVPR'18	72.0	80.0	76.0	-
TextSnake [35]	✓	ECCV'18	84.9	80.4	82.6	1.1
PSENet-1s [24]	✓	CVPR'19	86.9	84.5	85.7	1.6
SPCNet [50]	✓	AAAI'19	88.7	**85.8**	**87.2**	-
PAN	✓	-	84.0	81.9	82.9	**26.1**

Table 6. The single-scale results on ICDAR 2015. "P", "R" and "F" represent the precision, recall and F-measure respectively. "Ext." indicates external data.

Method	Ext.	Venue	MSRA-TD500			
			P	R	F	FPS
EAST [58]	-	CVPR'17	87.3	67.4	76.1	13.2
RRPN [38]	-	TMM'18	82.0	68.0	74.0	-
DeepReg [17]	-	ICCV'17	77.0	70.0	74.0	1.1
PAN	-	-	80.7	77.3	78.9	**30.2**
SegLink [42]	✓	CVPR'17	86.0	70.0	77.0	8.9
PixelLink [3]	✓	AAAI'18	83.0	73.2	77.8	3.0
Lyu et al. [37]	✓	CVPR'18	87.6	76.2	81.5	5.7
RRD [28]	✓	CVPR'18	87.0	73.0	79.0	10
MCN [32]	✓	CVPR'18	**88.0**	79.0	83.0	-
TextSnake [35]	✓	ECCV'18	83.2	73.9	78.3	1.1
PAN	✓	-	84.4	**83.8**	**84.1**	**30.2**

Table 7. The single-scale results on MSRA-TD500. "P", "R" and "F" represent the precision, recall and F-measure respectively. "Ext." indicates external data.

(a) Final Result (b) Predicted Text Region (c) Predicted Kernel (d) Predicted Similarity Vector

(e) CTW1500 (f) Total-Text (g) ICDAR 2015 (h) MSRA-TD500

Figure 6. Qualitative results of PAN. (a) is the final result of PAN. (b) is the predicted text regions. (c) is the predicted kernels. (d) is the visualization of similarity vectors, which is the best viewed in color and scatter diagram. (e)-(h) are results on four standard benchmarks.

Method	F	Time consumption (ms)			FPS
		Backbone	Head	Post	
PAN-320	77.10	**4.4**	**5.4**	**2.1**	**84.2**
PAN-512	80.32	6.4	7.3	3.5	58.1
PAN-640	**81.00**	9.8	10.1	5.2	39.8

Table 8. Time consumption of PAN on CTW-1500. The total time consists of backbone, segmentation head and post-processing. "F" represents the F-measure.

proposed PAN successfully detects text instances of arbitrary orientations and sizes.

Long straight text detection. To test the robustness of PAN to long straight text instance, we evaluate PAN on MSRA-TD500 benchmark. To ensure fair comparisons, we resize the short edge of test images to 736 as ICDAR 2015. As shown in Table. 7, the proposed PAN achieve F-measures of 78.9% and 84.1% when the external data is not used and used respectively. Compared with other state-of-the-art methods, PAN can achieve higher performance and run at a faster speed (30.2 FPS). Thus, PAN is also robust for long straight text detection (see Fig. 6 (h)) and can indeed be deployed in complex natural scenarios.

4.5. Result Visualization and Speed Analysis

Result visualization. An example of PAN prediction is shown in Fig. 6 (a-d). Fig. 6 (b) is the predicted text regions which keep the complete shape information of text instances. Fig. 6 (c) is the predicted kernels which clearly distinguish different text instances. Fig. 6 (d) is a visualization of similarity vectors. The dimensions of these vectors are reduced to 3 and 2 by PCA [49] for visualization. We can easily find that pixels belonging to its kernels have similar color and narrow distance with its cluster center (kernels).

Speed analysis. We specially analyze the time consumption of PAN in different stages. As shown in Table 8, the time costs of backbone and segmentation head are similar, and the time cost of post-processing is half of them. In practical applications, an obvious way to increase speed is to run the network and post-processing in parallel through a basic producer-consumer model, which can reduce the time cost to the original 4/5. The above experiments are conducted on CTW1500 test set. We evaluate all test images and calculate the average speed. All results in this paper are tested by PyTorch [40] with batchsize of 1 on one 1080Ti GPU and one 2.20GHz CPU in a single thread.

5. Conclusion

In this paper, we have proposed an efficient framework to detect arbitrary-shaped text in real-time. We firstly introduce a light-weight segmentation head consisting of Feature Pyramid Enhancement Module and Feature Fusion Module, which can benefit the feature extraction while bringing minor extra computation. Moreover, we propose Pixel Aggregation to predict similarity vectors between text kernels and surrounding pixels. These two advantages make the PAN become an efficient and accurate arbitrary-shaped text detector. Extensive experiments on Total-Text and CTW1500 demonstrate the superior advantages in speed and accuracy when compared to previous state-of-the-art text detectors.

Acknowledgments

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 Icewind Disaster Feature Recognition and Prediction by Few-shot Machine Learning in Transmission Lines).

References

[1] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[2] Chee Kheng Ch'ng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition. In *Proc. Int. Conf. Document Analysis Recogn.*, 2017.

[3] Dan Deng, Haifeng Liu, Xuelong Li, and Deng Cai. Pixellink: Detecting scene text via instance segmentation. In *Proc. AAAI Conf. Artificial Intell.*, 2018.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009.

[5] Deng-Ping Fan, Ming-Ming Cheng, Jiang-Jiang Liu, Shang-Hua Gao, Qibin Hou, and Ali Borji. Salient objects in clutter: Bringing salient object detection to the foreground. In *Proc. Eur. Conf. Comp. Vis.*, 2018.

[6] Deng-Ping Fan, Ming-Ming Cheng, Yun Liu, Tao Li, and Ali Borji. Structure-measure: A new way to evaluate foreground maps. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2017.

[7] Deng-Ping Fan, Cheng Gong, Yang Cao, Bo Ren, Ming-Ming Cheng, and Ali Borji. Enhanced-alignment measure for binary foreground map evaluation. *arXiv preprint arXiv:1805.10421*, 2018.

[8] Deng-Ping Fan, Zheng Lin, Jia-Xing Zhao, Yun Liu, Zhao Zhang, Qibin Hou, Menglong Zhu, and Ming-Ming Cheng. Rethinking rgb-d salient object detection: Models, datasets, and large-scale benchmarks. *arXiv preprint arXiv:1907.06781*, 2019.

[9] Deng-Ping Fan, Wenguan Wang, Ming-Ming Cheng, and Jianbing Shen. Shifting more attention to video salient object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019.

[10] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 2961–2969, 2017.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proc. Eur. Conf. Comp. Vis.*, 2016.

[16] Pan He, Weilin Huang, Tong He, Qile Zhu, Yu Qiao, and Xiaolin Li. Single shot text detector with regional attention. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2017.

[17] Wenhao He, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Deep direct regression for multi-oriented scene text detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2017.

[18] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[19] Han Hu, Chengquan Zhang, Yuxuan Luo, Yuzhuo Wang, Junyu Han, and Errui Ding. Wordsup: Exploiting word annotations for character based text detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2017.

[20] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[22] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *Proc. Int. Conf. Document Analysis Recogn.*, 2015.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[24] Xiang Li, Wenhai Wang, Wenbo Hou, Ruo-Ze Liu, Tong Lu, and Jian Yang. Shape robust text detection with progressive scale expansion network. *arXiv preprint arXiv:1806.02559*, 2018.

[25] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019.

[26] Minghui Liao, Baoguang Shi, and Xiang Bai. Textboxes++: A single-shot oriented scene text detector. *IEEE Trans. Image Process.*, 2018.

[27] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu. Textboxes: A fast text detector with a single deep neural network. In *Proc. AAAI Conf. Artificial Intell.*, 2017.

[28] Minghui Liao, Zhen Zhu, Baoguang Shi, Gui-song Xia, and Xiang Bai. Rotation-sensitive regression for oriented scene text detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018.

[29] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017.

[30] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proc. Eur. Conf. Comp. Vis.*, 2016.

[31] Yuliang Liu, Lianwen Jin, Shuaitao Zhang, and Sheng Zhang. Detecting curve text in the wild: New dataset and new solution. 2017.

[32] Zichuan Liu, Guosheng Lin, Sheng Yang, Jiashi Feng, Weisi Lin, and Wang Ling Goh. Learning markov clustering networks for scene text detection. *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018.

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[34] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[35] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. *Proc. Eur. Conf. Comp. Vis.*, 2018.

[36] Pengyuan Lyu, Minghui Liao, Cong Yao, Wenhao Wu, and Xiang Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In *Proc. Eur. Conf. Comp. Vis.*, 2018.

[37] Pengyuan Lyu, Cong Yao, Wenhao Wu, Shuicheng Yan, and Xiang Bai. Multi-oriented scene text detection via corner localization and region segmentation. *arXiv preprint arXiv:1802.08948*, 2018.

[38] Jianqi Ma, Weiyuan Shao, Hao Ye, Li Wang, Hong Wang, Yingbin Zheng, and Xiangyang Xue. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia*, 2018.

[39] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *Proc. Int. Conf. 3D Vision*, 2016.

[40] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. Advances in Neural Inf. Process. Syst.*, 2015.

[42] Baoguang Shi, Xiang Bai, and Serge Belongie. Detecting oriented text in natural images by linking segments. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017.

[43] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. Int. Conf. Learn. Representations*, 2015.

[45] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[46] Zhiqiang Tang, Xi Peng, Shijie Geng, Yizhe Zhu, and Dimitris Metaxas. Cu-net: Coupled u-nets. In *BMVC*, 2018.

[47] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. In *Proc. Eur. Conf. Comp. Vis.*, 2016.

[48] Wenhai Wang, Xiang Li, Tong Lu, and Jian Yang. Mixed link networks. In *Proc. Int. Joint Conf. Artificial Intell.*, 2018.

[49] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, 1987.

[50] Enze Xie, Yuhang Zang, Shuai Shao, Gang Yu, Cong Yao, and Guangyao Li. Scene text detection with supervised pyramid context network. In *Proc. AAAI Conf. Artificial Intell.*, 2019.

[51] Cong Yao, Xiang Bai, and Wenyu Liu. A unified framework for multioriented text detection and recognition. *IEEE Transactions on Image Processing*, 23(11):4737–4749, 2014.

[52] Cong Yao, Xiang Bai, Wenyu Liu, Yi Ma, and Zhuowen Tu. Detecting texts of arbitrary orientations in natural images. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012.

[53] Cong Yao, Xiang Bai, Nong Sang, Xinyu Zhou, Shuchang Zhou, and Zhimin Cao. Scene text detection via holistic, multi-channel prediction. *arXiv preprint arXiv:1606.09002*, 2016.

[54] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proc. Eur. Conf. Comp. Vis.*, 2018.

[55] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. Multi-oriented text detection with fully convolutional networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[56] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017.

[57] Jia-Xing Zhao, Yang Cao, Deng-Ping Fan, Ming-Ming Cheng, Xuan-Yi Li, and Le Zhang. Contrast prior and fluid pyramid integration for rgbd salient object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019.

[58] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. *arXiv preprint arXiv:1704.03155*, 2017.

6. Appendix

6.1. Robustness Analysis

To further demonstrate the robustness of the proposed PAN, we evaluate the model by training on one dataset and testing on other datasets. Based on the annotation level, we divide the datasets into two groups which are word level and text line level datasets. SynthText, ICDAR 2015 and Total-Text are annotated at word level, while CTW1500 and MSRA-TD500 are annotated at text line level. For fair comparisons, we train all model without any external dataset, and the short side of test images in ICDAR 2015, MSRA-TD500, CTW1500 and Total-Text are set to 736, 736, 640, 640 respectively.

The cross-dataset results of PAN are shown in Table 9. Notably, the proposed PAN trained on SynthText (a synthetic dataset) have reluctantly satisfied performance on ICDAR 2015 and Total-Text, which indicates that even without any manually annotated data, PAN can satisfy the scene with low precision requirements. The PAN trained on manually annotated dataset has over 64% F-measure in the cross-dataset evaluation, which is still competitive. Furthermore, in the cross-dataset evaluation at text line level, all models achieve the F-measure of nearly 75% even the training and the testing are performed on quadrangle and curved text datasets respectively. These cross-dataset experiments demonstrate that the proposed PAN is robust in generalizing to brand new datasets.

6.2. Comparisons with Other Semantic Segmentation Methods

Unlike common semantic segmentation tasks, text detection needs to distinguish different text instances that lie closely. So feature map resolution matters and cannot be too small. However, most of high efficiency segmentation methods (i.e. BiSeNet [54]) make prediction on $1/8$ feature map, sacrificing accuracy for speed. Their speed will reduce sharply if using $1/4$ feature map directly. Thus, 'how to keep the high efficiency and the high resolution feature map simultaneously?" is a challenging problem, and our answer is "ResNet18 + 2FPEM + FFM". We compare our method with two methods BiSeNet [54] and CU-Net [46] on CTW1500. For fair comparisons, we set the backbone of BiSeNet to ResNet18 and use one of default settings of CU-Net, which has the similar speed with our method. As shown in Table 10, our method enjoys obviously better accuracy (+2.2% and +4.6%) at the similar speed.

Annotation	Train Set→Test Set	P	R	F
Word	SynthText → ICDAR 2015	65.9	46.9	54.8
	SynthText → Total-Text	69.1	40.8	51.3
	ICDAR 2015 → Total-Text	72.0	57.8	64.1
	Total-Text → ICDAR 2015	77.6	65.5	71.1
Text Line	CTW1500 → MSRA-TD500	76.6	73.1	74.8
	MSRA-TD500 → CTW1500	82.4	69.1	75.2

Table 9. Cross-dataset results of PAN on word-level and line-level datasets. "P", "R" and "F" represent the precision, recall and F-measure respectively.

Methods	Ext.	F (%)	FPS
BiSeNet (ResNet18) [54]	-	78.8	25.9
CU-Net-2 (m=128, n=32) [46]	-	76.4	39.3
Ours (ResNet18 + 2FPEM + FFM)	-	**81.0**	**39.8**

Table 10. The results on CTW1500 of different segmentation methods. "F" means F-measure. "Ext." indicates external data.

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

Figure 7. Failure Samples.

6.3. Failure Samples

As demonstrated in previous experiments, the proposed PAN works well in most cases of arbitrary-shaped text detection. It still fails for some difficult cases, such as large character spacing (see Fig. 7 (a)), symbols (see Fig. 7 (b)) and false positives (see Fig. 7 (c)(d)). Large character spacing is an unresolved problem which also exists in other state-of-the-art methods such as RRD [28]. For symbol detection and false positives, PAN is trained on small datasets (about 1000 images) and we believe this problem will be alleviated when increasing training data.

6.4. More Detected Results on CTW1500, Total Text, ICDAR 2015 and MSRA-TD500

In this section, we show more test examples produced by PAN on different datasets in Fig. 8 (CTW1500) Fig. 9 (Total-Text), Fig. 10 (ICDAR 2015) and Fig. 11 (MSRA-TD500). From these results, we can find that the proposed PAN have the following abilities: i) separating adjacent text instances with narrow distances; ii) locating the arbitrary-shaped text instances precisely; iii) detecting the text instances with various orientations; iv) detecting the long text instances; v) detecting the multiple Lingual text. Meanwhile, thanks to the strong feature representation, PAN can also locate the text instances with complex and unstable il-

lumination, different colors and variable scales.

Figure 8. Detection results on CTW1500.

Figure 9. Detection results on Total-Text.

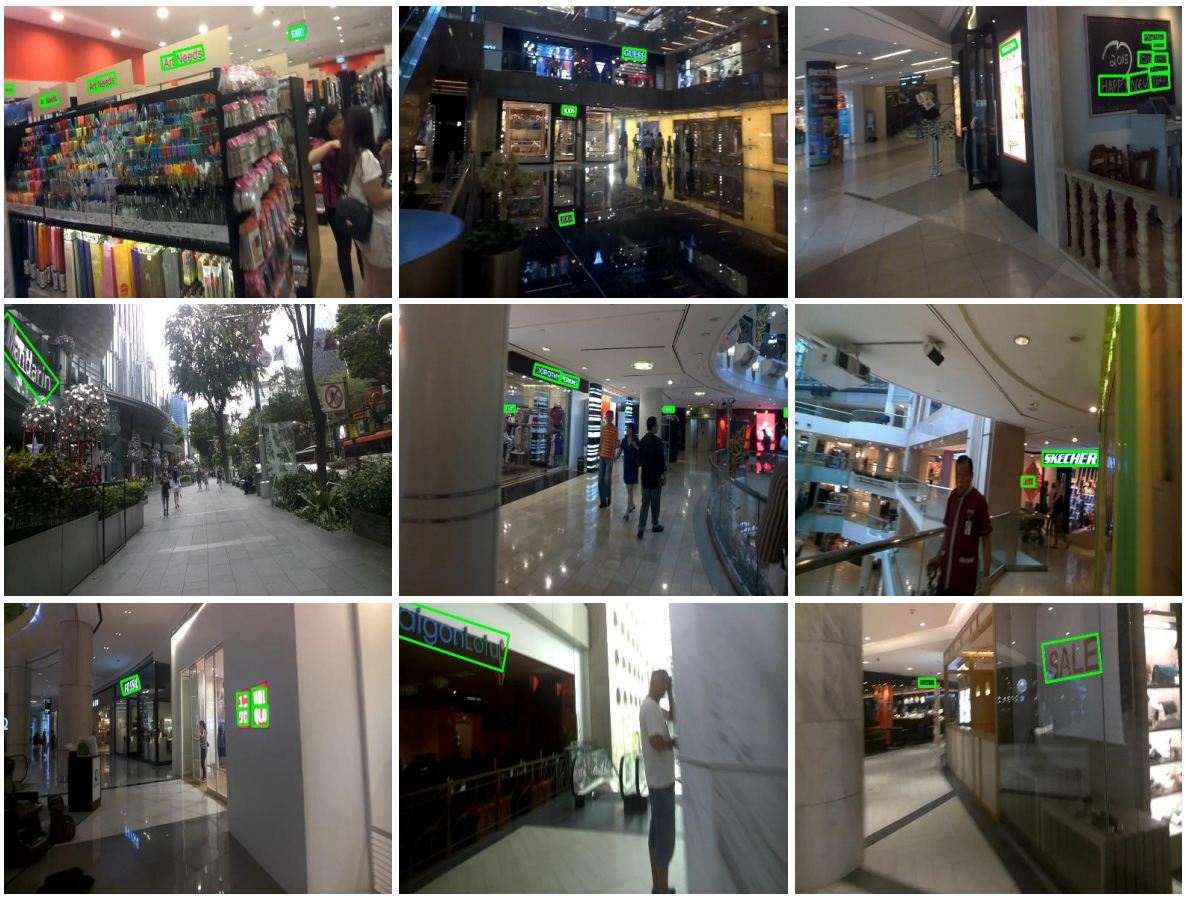

Figure 10. Detection results on ICDAR 2015.

Figure 11. Detection results on MSRA-TD500.