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

Zhong Hu-Men
Nanjing University
zhonghumen@smail.nju.edu.cn

Abstract

Scene Text Detection has witnessed rapid development thanks to the great power of Convolutional Neural Networks. In the recent years, the computer vision community has made a great progress especially in the object detection domain. However, when it comes to the Scene Text Detection problem, those models designed specially for object detection seem to have a worse performance. They may have difficulties when dealing with some troublesome problems like how to seperate crowded text, how to adjust the hyperparameters, how to detect Arbitrary-Shaped text, etc. In 2019CVPR and 2019ICCV, two successive papers [9, 10] with similar ideas come out. In these two papers, the authors use the object segmentation pipeline to sovle this problem and achieves great success in both accuracy and efficiency. In this article, I will show the core idea of these two papers and explain some details that may cause trouble during reimplementation. The entire training and testing code has been released, you can check the code in: `https://github.com/HumanZhong/PAN-reimplement`.

1. Introduction

Scene text detection is an important problem in the computer vision community. Many applications like autonomous driving, scene understanding require a powerful and efficient scene text detector to better understand what the machine see. With the development of CNN, researcher have made great progress in recent years. Their methods can be roughly divided into two kinds i.e. segmentation-based methods and regression-based methods. Generally speaking, segmentation-based methods utilize segmentation-style pipeline and try to classify a region with text/non-text labels. On the other hand, regression-based methods mainly utilize traditional one-stage or two-stage object detection framework and take text as the detection target. However, previous methods all have difficulties when dealing with crowded and arbitrary-shaped text. Segmentation-based methods can hardly seperate different text instances correctly especially when facing crowded texts. For regression-based methods, due to the rectangle-style or quadrangle-style bounding box format, it is hard to detect texts with arbitrary shapes.

In recent years, more methods [5, 12, 8] try to solve these problems, and in this article I will make a brief introduction to two of them whose central ideas follow the segmentation pipeline and propose a brilliant 'kernel' approach to deal with the problems I mentioned above. The two methods are two successive methods i.e. PSENet [9] and PAN [10]. Meanwhile, the detailed information and personal modifications during my re-implementation will alse be provided as a guideline.

2. Related Work

Deep learning-based scene text detection have made great progress in the past few years, and scene text detectors can be divided into two categories due to their different pipeline, i.e. segmentation-based methods and regression-based methods.

Segmentation-based methods often take the segmentation pipeline and follow a FCN-like [4] structure. These methods consider text detection as a segmentation problem by giving each pixel a text/non-text label.

Regression-based methods however often take the traditional object detector [7, 6] as their main framework. The only difference between scene text detection and object detection is that scene text detection problem take texts which may have arbitrary shapes instead of general objects as the detection targets.

These two kinds of methods both have achieved remarkable performances in the past. But there are still drawbacks, for example, regression-based methods like EAST [13] can't handle the arbitrary-shape problem and segmentation-based methods can hardly seperate near-by text instances.The PSENet and PAN that I will discuss later however use a brilliant 'kernel' idea to nicely solve out this

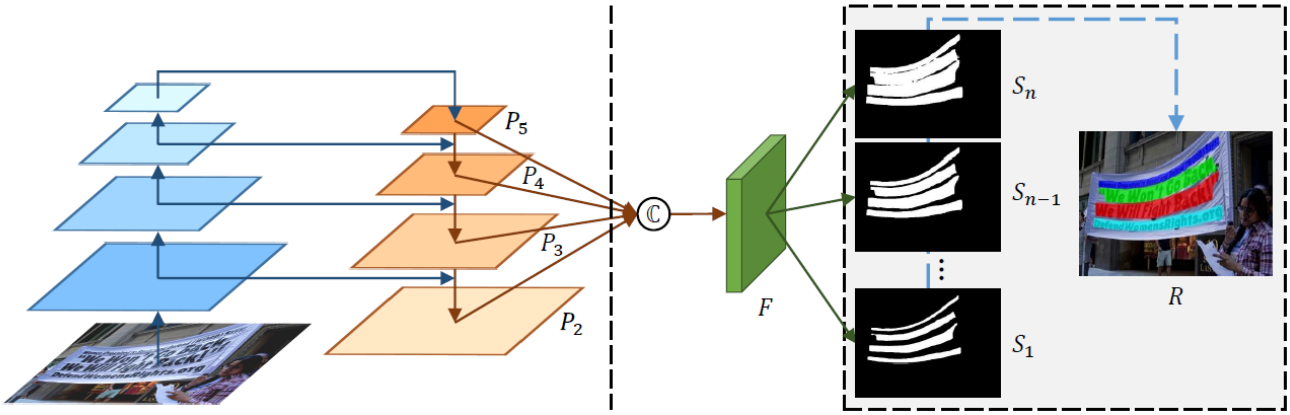

Figure 1. Pipeline of PSENet.

problem.

3. Methodology

In this section, I will introduce the core idea of PSENet and PAN, and the entire pipeline will be discussed.

3.1. PSENet

Previous segmentation-based methods have some trouble in dealing with crowded texts since they only treat each pixel as a 2-class classification problem. In some complex situation like several text instances are placed closely, text detector can not seperate them correctly.

As illustrated in Figure 1., PSENet follows a FCN-like segmentation pipeline. To solve the above problem, PSENet proposed a novel idea which utilized several kernels with different shrink ratio to seperate adjacent text instances. Concretely, PSENet defines the shrinked text region of a text instance as 'kernel' and defines the entire text region of a text instance as 'text'.

To better locate the text instance and avoid the big gap between the small 'kernel' and the entire 'text', this paper uses several kernels with different shrink ratio. During training, PSENet take a FCN-like segmentation pipeline and outputs a K+1-channel result map which have the same length and width to the input image. K indicates the number of different kernels, and 1 indicates the channel which predicts the entire 'text' region. Meanwhile, each channel of the result map is a 2-class classification indicating text/non-text labels.

During inference, PSENet outputs a K+1-channel result map.The postprocessing starts from the smallest kernel and gradually expand to a larger kernel and then expand to the entire 'text' region. This algorithm is named as 'Progressive Scale Expansion(PSE)' which is implemented following a BFS pipeline. This algorithm enables the model to nicely seperate different text instances because the minimal kernels of text instances are well seperated, and if we take the minimal kernels as the first step, text instances can be seperated from the very beginning. What we do in the following steps mainly focus on predicting an accurate boundary for a specific text instance.

3.2. PAN

With the help of PSE algorithm, PSENet is able to seperate adjacent text instances. However, PSENet still have some drawbacks which are mainly caused by the complex PSE algorithm. During the PSE algorithm, the model starts from the smallest kernel and gradually expand to the largest 'text' region. This algorithm may cost too much time because of the number of kernels is large which causes a high time complexity. But naively reducing the number of kernels may lead to a worse performance.

In PAN, the author utilized the idea of embedding feature to reduce the number of kernels and successively avoid the performance drop. The fundamental idea of PAN is to learn an embedding feature for text instances so that pixels that belong to the same text instance have similar embedding feature and pixels that belong to different text instances have outputs embedding features which have large distances between each other.

The above embedding learning mainly focus on reducing time cost during postprocessing. Besides, PAN also take measures to reduce the time cost in the model inference stage. Concretely, PAN choose a light-weight res18 as the backbone of the network and propose a novel Feature Pyramid Enhancement Module(FPEM) to replace the traditional FPN and provide enhanced multi-scale features.

FPEM is a U-shaped module as illustrated in Fig. 3. 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

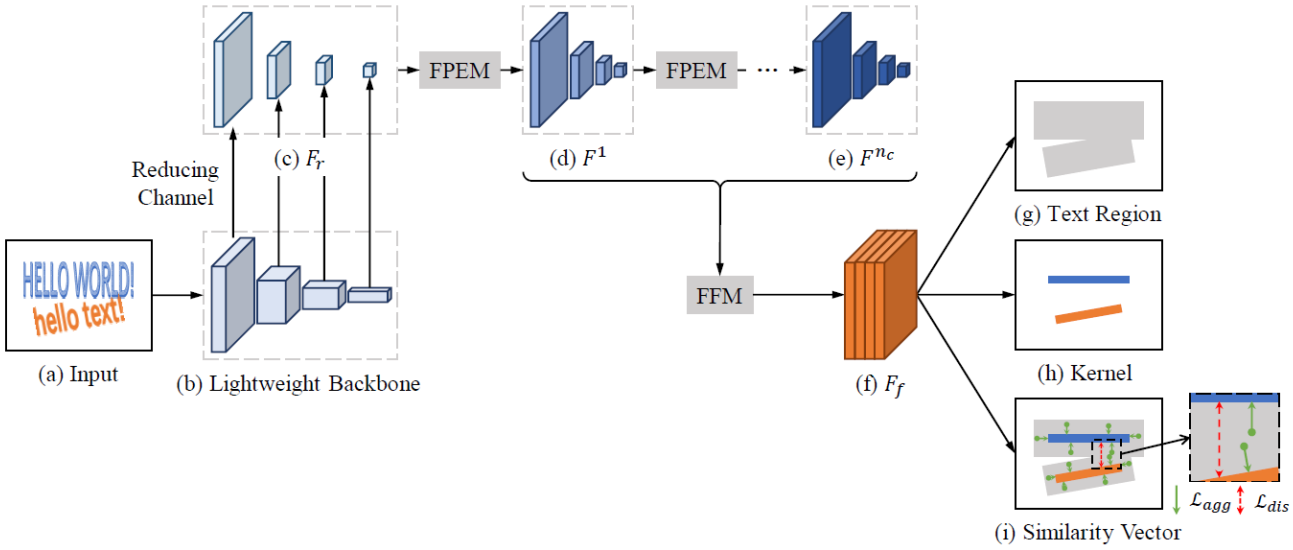

Figure 2. Pipeline of PAN.

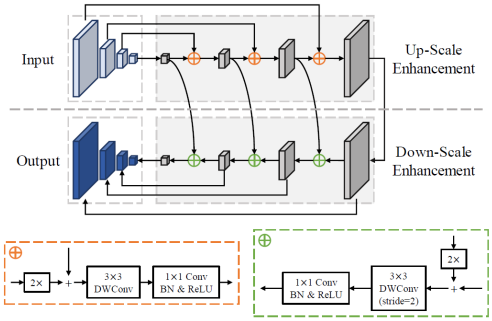

Figure 3. Illustration of FPEM. It consists of two sub-module i.e. up-scale enhancement and down-scale enhancement module.

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.

With the help of embedding feature and FPEM, PAN achieved a remarkable inference speed as well as state-of-the-art accuracy.

4. Personal Modifications

In this section, I will introduce some personal modifications which mainly focus on looking for a better method to represent a text instance. PSENet and PAN represent a text instance by its boundary points, but this method may cause trouble when applied to a following recognition task because the machine can hardly extract the corresponding features using ROI pooling layer with only the boundary points.

To deal with this problem, I proposed a new method to represent a text instance by its top/bot edge.

4.1. Top/Bot Edge Prediction

Top/Bot edge prediction also follows a FCN-like segmentation pipeline. It appends two additional channel to the result map of the original PAN i.e. top edge prediction channel and bottom edge prediction channel, each of them is a binary classification problem tagging every pixel with top/non-top or bot/non-bot labels.

The groundtruth of top and bottom edge is generated from the word-level text instance bounding box. For top edge groundtruth generation, I divide the points on the word-level bounding box into two categories i.e. top points and bottom points. Let's take a bounding box with 14 vertices as an example, after the division, the 14 vertices are divided into 7 top vertices and 7 bottom vertices notated as T_i and B_i where $i \in \{1, \cdots, 7\}$.

Then, the top vertices of the generated top edge are the same as the top vertices of the word-level bounding box. And the bottom vertices of the generated top edge are computed by the following formulation:

$$B_i^\star = k * T_i + (1 - k) * B_i$$

where $B_i^\star$ means the i-th bottom vertices of the generated top edge, and k means the level that measures how close the bottom vertices of the top edge is with the top vertices of the word-level bounding box. k is emprirically set to 0.75.

After the top/bot edge groundtruth is generated, the entire model still follows a segmentation pipeline and outputs a top edge prediction map, a bottom edge prediction map, a 2-channel kernel/text prediction map and a N-channel embedding feature prediction map, where N represents the

Figure 4. Qualitative experimental results. The above three images respectively represent (a)text region detection result generated by the original PAN, (b) text region detection and top/bot edge detection result generated by the PAN-edge, and (c)final detection result generated by the PAN-edge. The illustrated result shows that PAN with top/edge prediction sub-branch i.e. PAN-edge can have a better representation for a text instance.

number of channels of the embedding feature and is set to 4 in my experiment.

During inference, taking top edge prediction as an example, I will shrink the predicted top edge region and get the top edge result. Concretely, I will extract the top boundary of the predicted top edge region by doing a conv. operation with a conv kernel $[-1, 1, 0]^T$. After doing the conv. operation, if the value remains unchanged, this pixel is labelled as top edge point.

5. Experiment

In this section, I will briefly introduce several datasets and present the details of my re-implementation.

5.1. Datasets

ICDAR2015 [3] 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.

CTW1500 [11] is a challenging dataset for long curve text detection, which is constructed by Yuliang et al. 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.

5.2. Reimplementation Details

I follow the implementation details that is provided in the original PAN paper, using ResNet [2] pretrained on ImageNet [1] as the backbone. The dimension of embedding feature is set to 4. The entire network is optimized by using stochastic gradient descent(SGD). I train PAN with a bacthsize of 16 on 4 GPUs for 600 epochs, and the initial learning rate is set to 0.001. Besides, We use a weight decay of 0.0005 and a momentum of 0.99.

Method	R	P	F-score
PSENet(paper)	79.7	81.5	80.6
PSENet(reimplementation)	-	-	80.5
PAN-640(paper)	77.8	82.9	80.3
PAN-640(reimplementation)	-	-	79.9

Table 1. Single-scale results on ICDAR2015 with R meaning 'recall' and P meaning 'Precision'.

Method	R	P	F-score
PSENet(paper)	75.6	80.6	78.0
PSENet(reimplementation)	75.8	80.7	78.2
PAN-640(paper)	77.7	84.6	81.0
PAN-640(reimplementation)	-	-	80.5
PAN-edge	-	-	80.1

Table 2. Single-scale results on CTW1500, where PAN-edge indicates the PAN with top/edge prediction sub-branch. R means 'recall' and P means 'Precision'.

What's more, I also did some additional ablation study which mainly focus on the influence of applying or not applying top/bot edge prediction.

5.3. Experimental Results

My reimplementation follows nearly all of the hyper-parameter settings which are provided in the PAN paper and achieves a similar performance. However, because of some unknown issues, my reimplementaion still can't fully keep up with the original PAN with a gap of about 0.4-0.5 f-score as illustrated in Table 1 and Table 2.

Some qualitative results are shown in Figure 4. The PSENet and PAN successfully detect arbitrary-shaped texts. Besides, the top/bot edge prediction sub-branch also shows good performance on detecting text instances' top edge and bottom edge.

6. Conclusion

In this article, I give a brief introduction to PSENet and PAN. Based on the original PAN, I also make some small modifications in order to improve the representative ability. Besides, the details of re-implementation are released and the experimental results of the re-implementation are provided.

References

[1] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[3] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160, 2015.

[4] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[5] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[8] B. Shi, X. Bai, and S. Belongie. Detecting oriented text in natural images by linking segments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao. Shape robust text detection with progressive scale expansion network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[10] W. Wang, E. Xie, X. Song, Y. Zang, W. Wang, T. Lu, G. Yu, and C. Shen. Efficient and accurate arbitrary-shaped text detection with pixel aggregation network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[11] L. Yuliang, J. Lianwen, Z. Shuaitao, and Z. Sheng. Detecting curve text in the wild: New dataset and new solution, 2017.

[12] C. Zhang, B. Liang, Z. Huang, M. En, J. Han, E. Ding, and X. Ding. Look more than once: An accurate detector for text of arbitrary shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[13] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: An efficient and accurate scene text detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.