PGNet: Real-time Arbitrarily-Shaped Text Spotting with Point Gathering Network

Pengfei Wang,∗ Chengquan Zhang,∗ Fei Qi,∗ Shanshan Liu, Xiaoqiang Zhang, Pengyuan Lyu, Junyu Han, Jingtuo Liu, Errui Ding, Guangming Shi

1School of Artificial Intelligence, Xidian University, 2Department of Computer Vision Technology, Baidu Inc.
pfwang@stu.xidian.edu.cn, zhangchengquan@baidu.com, fred.qi@ieee.org, {liushanshan07, zhangxiaoqiang01}@baidu.com, lvpyuan@gmail.com, {hanjunyu, liujingtuo, dingerrui}@baidu.com, gmshi@xidian.edu.cn

Abstract

The reading of arbitrarily-shaped text has received increasing research attention, but existing text spotters are mostly built on two-stage frameworks or character-based methods, which suffer from either Non-Maximum Suppression (NMS) and Region-of-Interest (RoI) operations or character-level annotations. In this paper, to address above problems, we propose a novel fully convolutional Point Gathering Network (PGNet) for reading arbitrarily-shaped text in real-time. PGNet is a single-shot text spotter, where the pixel-level character classification map is learned with proposed PG-CTC loss avoiding the usage of character-level annotations. With PG-CTC decoder, we gather high-level character classification vectors from two-dimensional space and decode them into text symbols without NMS and RoI operations involved, which guarantees high efficiency. Additionally, reasoning the relations between each character and its neighbors, a graph refinement module (GRM) is proposed to optimize the coarse recognition and further improve the end-to-end performance. Experiments demonstrate that the proposed method achieves state-of-the-art or competitive accuracy, meanwhile significantly improving the running speed. In particular, on Total-Text, it runs at 46.7 FPS, surpassing the previous spotters with a large margin.

1 Introduction

Recently, scene text reading has attracted extensive attention in both academia and industry for its numerous applications, such as scene understanding, image retrieval, augmented reality translation (Wu et al. 2019), and robot navigation. Thanks to the surge of deep neural networks, significant progress has been made in detection and recognition separable solutions (Wu and Natarajan 2017; Long et al. 2018; Wang et al. 2019a,b; Zhan and Lu 2019; Shi et al. 2018; Yu et al. 2020; Wan et al. 2019), as well as end-to-end text spotting methods. However, existing end-to-end models (Sun et al. 2018; Liu et al. 2018; Feng et al. 2019) are mostly built on two-stage frameworks or character-based methods (Xing et al. 2019; Lyu et al. 2018) with a complex pipeline, which are inefficient for real-time applications. In this paper, we try to investigate a real-time text spotter for arbitrarily-shaped text.

∗Equal contribution. Fei Qi is the corresponding author.
Copyright ⓒ 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
In this paper, we propose a novel framework for reading text in real-time speed with point gathering operation, namely PGNet. The PGNet is a single-shot text spotter based on multi-task learning. The architecture of PGNet is shown in Fig. 3. We employ a FCN (Milletari, Navab, and Ahmadi 2016) model to learn various information of text regions simultaneously, including text center line (TCL), text border offset (TBO), text direction offset (TDO), and text character classification map (TCC). The pixel-level character classification map is trained with a proposed Point Gathering CTC (PG-CTC) loss, making it free from character-level annotations. In the post-processing, we extract the center point sequence in reading order of each text instance with TCL and TDO maps, and the detection results can be obtained with the corresponding boundary offset information from TBO map. Using the PG-CTC decoder, we serialize high-level two-dimensional TCC map to character classification probability vector sequences which can be further decoded to the recognition results. The details will be discussed in Sec.3.1. As depicted in Fig. 2 and Fig. 1, our pipeline is simple yet efficient, and experiments on public benchmarks prove that PGNet achieves better or competitive performance in end-to-end performance with excellent running speed.

Moreover, inspired by SRN (Yu et al. 2020) and GTC (Hu et al. 2020), we propose a graph refinement module (GRM) to make secondary reasoning to further improve the end-to-end performance. The points in a text sequence are formulated as nodes in a graph, where the representation of each node is enhanced with semantic context and visual context information from its neighbors, and the character classification result should be more accurate.

The contributions of this paper are three-fold:

- We propose a simple yet powerful arbitrarily-shaped text spotter (PGNet), which is free from character-level annotations, NMS and RoI operations, and it achieves better or competitive performance in end-to-end performance with excellent running speed;
- We introduce a mechanism to restore the reading order of characters in each text instance, making our method able to correctly recognize text in more challenging situations and non-traditional reading directions;
- We also propose an efficient graph refinement module (GRM) to improve the CTC recognition.

## 2 Related Work

In this section, we will review some representative scene text spotters, as well as some recent progress in graph neural networks. A comprehensive review of recent scene text spotters can be found in (Ye and Doermann 2015; Zhu, Yao, and Bai 2016; Baek et al. 2019).

**Scene Text Spotting.** Inspired by the generic object detection methods (Liu et al. 2016; Ren et al. 2015; Redmon et al. 2016) and segmentation methods (He et al. 2017; Milletari, Navab, and Ahmadi 2016), the text spotting methods are developed from spotting regular scene text to spotting arbitrarily-shaped scene text. Lee and Osindero (2016) proposed the first successful end-to-end text recognition model which only supports horizontal text and requires relatively complex training procedures. To address the multi-orientation problem of text, Bušta et al. (2017) utilize YOLO (Redmon et al. 2016) to generate rotational proposals, and train RoI sampled features with CTC loss. Inspired by the Faster RCNN (Ren et al. 2015), TextNet (Sun et al. 2018) generates text proposals in quadrangles, and encode the aligned RoI features into context information with a simple recurrent neural network to generate the text sequences, which contains some background information, thus it may suffer from reading curve texts.

For the spotting of arbitrarily-shaped scene text, Mask TextSpotter (Lyu et al. 2018) detects and recognizes text instances of arbitrary shapes by segmenting the text regions and character regions. However, the character-level annotations are required for training, which is too expensive to afford. Considering the arbitrarily-shaped region of text as a series of quadrangles, TextDragon (Feng et al. 2019) extracts the components of text feature through RoISlide and...
recognizes each cropped feature with CTC based text recognizer. ABCNet (Liu et al. 2020) attempts to adaptively fit arbitrarily-shaped text by a parameterized Bezier curve to reduce the computation overhead, and they propose a BezierAlign layer for extracting accurate convolution features of a text instance, making it suitable for some real-time scenarios. Mask Textspotter, TextDragon, and ABCNet are capable of spotting arbitrarily-shaped scene text, but they are all RoI-based methods and may involve NMS, RoI cropping, and pooling operations, which is time-consuming and may reduce the performance. CharNet is the first one-stage arbitrarily-shaped scene text spotting method, which requires character-level ground truth data for training and its backbone is too heavy to run in a real-time speed.

**Graph Neural Networks.** The unstructured data is common in many machine learning tasks and can be organized as graphs. Thanks to the previous effort, we can use the convolutional neural network for graph-structured data. The graph convolutional networks (GCNs) can be categorized into spectral methods and spatial methods. Spectral GCNs generalize convolution by Graph Fourier Transform, while spatial GCNs directly perform manually-defined convolution on graph nodes and their neighbors. For more details about GCNs, please refer to (Wu et al. 2020). In terms of applications, GCNs is proved to be efficient in many computer vision tasks, including scene graph generation, point clouds classification, and action recognition. For example, Zhang et al. (2020) propose a relational reasoning graph network for arbitrary shape text detection by predicting linkages of text components. In this paper, we adopt the Spatial GCN to reasoning the semantic information between point and its neighbors to improve the CTC recognition.

### 3 Methodology

The architecture of our proposed method is shown in Fig. 3. Firstly, the input image is fed into a stem backbone with FPN to produce feature $F_{\text{visual}}$. Then, the $F_{\text{visual}}$ is used to predict TCL, TBO, TDO, and pixel-level TCC map by multi-task learning in parallel at $1/4$ size of the input images. In the training period, the TCL, TBO, and TDO are supervised by the same scale label maps, while a PG-CTC loss is proposed to train the pixel-level TCC map to solve the lack of character-level annotations. In the inference period, we extract the center point sequence of each text instance from TCL, and sort them with TDO information to recover the right reading order, which makes our method able to correctly recognize text in non-traditional reading directions. With the aid of the corresponding boundary offset information from TBO, the detection of each text instance can be achieved in a single shot by polygon restoration. Simultaneously, the PG-CTC decoder can serialize high-level two-dimensional TCC map to character classification probability sequences and decode them into final text recognition results.

#### 3.1 Point Gathering CTC

The point gathering (PG) operation plays an important role in both the training and inference process of PGNet and helps to get rid of character-level annotations, NMS, and RoI operation. The TCC maps of PGNet are maps of 37 characters, including 26 letters, 10 Arabic numerals, and one background class. The point gathering operation is employed to gather the character classification probability sequence from TCC map according to the center points in the center of each
where \( \pi = \{ p_1, p_2, \ldots, p_N \} \) is a center point sequence of length \( N \), and \( p_i = (x_i, y_i) \). The output \( P_\pi \) is the character classification probability sequence with size \( N \times 37 \).

In the training process, the proposed PG-CTC loss makes the training of pixel-level TCC map free from character-level annotations. The typical CTC loss function tackles the training problem of the source and target sequences with inconsistent lengths with a background class. The CRNN (Shi, Bai, and Yao 2017) framework transforms the height of the feature map to 1, which may suffer from the background noise while recognizing the curved text. The 2DCTC (Wan et al. 2019) expands the search path of CTC to two-dimensional space, but it still cannot handle an image with multi-text instances. Here we address the problem by PG-CTC and formulate the classic CTC loss as

\[
L_{\text{PG-CTC}} = \sum_{i=1}^{M} \text{CTC loss}(P_{\pi_i}, L_i),
\]

where we can calculate the centerline of polygon word-level annotations and sample it densely to obtain the center point sequences \( \pi_i \) in the training process, instead of using character-level annotations. With the training of big data, the character classification information of each pixel in TCC could be learned.

In the inference process, the PG-CTC decoder helps to greatly simplify the overall pipeline of an end-to-end arbitrarily-shaped text spotter, and the NMS and RoI operation is not required in PGNet. For a text region in the TCL map, we extract a center point sequence and sort it in the right reading order, which can be denoted as \( \pi \). Specifically, we adopt a morphological method (Zhang and Suen 1984) to get the skeleton of a text region and treat it as the center point sequence. The text direction of each point can be extracted from TDO maps. We calculate an average direction of all points and sort them according to the length of projection along the direction to obtain the center point sequence \( \pi \). The character classification probability sequence \( P_\pi \) can be extracted with Eq. (1), and the PG-CTC decoder can be denoted as

\[
R_\pi = \text{CTC decoder}(P_\pi),
\]

where \( R_\pi \) represents the transcription of \( \pi \). For the polygon restoration, we obtain the corresponding border point pairs of \( \pi \) with TBO maps in the same position, and link all the border points clockwise to obtain a complete polygon representation. For more details about polygon restoration, please refer to SAST (Wang et al. 2019a). Compared with CTC-based CRNN framework, the PG-CTC could handle images with multi-text instances of arbitrary shape, where the application of CTC loss is expanded a lot.

---

**3.2 Network Architecture**

Considering the limitation of computing resources in different scenarios, we propose two versions of PGNet, that is, PGNet-Accuracy and PGNet-Efficient, denoted by PGNet-A and PGNet-E in the following sections. The only difference is stem network, and the PGNet-A adopts ResNet-50 as the backbone network, while PGNet-E employs EfficientNet-B0. With different levels of feature map from the stem network gradually merged three-times in the FPN manner, a fused feature map \( F_{\text{visual}} \) is produced at 1/4 size of the input images. The TCL and the other maps are predicted in parallel, where we adopt a 1 \times 1 convolution layer with the number of output channel set to \{1, 2, 4, 37\} for TCL, TDO, TBO, and TCC map respectively.

---

**3.3 Label Generation**

The label generation of arbitrarily-shaped text is shown in Fig. 4. The TCL map is the shrunk version segmentation of the text region. The TBO map indicates the offset between each pixel in TCL and the corresponding point pair in the upper and lower edge of its text region, which helps to determine the boundaries of text regions in the inference. We follow SAST (Wang et al. 2019a) to generate TCL and TBO map, where more details are introduced. Inspired by the reading mechanism of humans that the eye moves from one character to the next character along the centerline of text region while reading, TDO map is estimated to recover the reading order of scene text components, which benefits both detection and recognition tasks, especially for scene text in non-traditional reading directions. The TDO map indicates the offset vector of each pixel in the TCL map to the next reading position. For a quadrilateral region annotation, the direction of the offset vector is from the center point of the left edge to the center point of the right edge, and its magnitude is the length of text region normalized by the number of characters. Polygonal annotations of more than four vertices are treated as a series of quadrangles connected together, and TBO map and TDO map can be generated gradually from quadrangles as described before.
3.4 Training Objectives

The loss of multi-task learning can be formulated as

\[ L = \lambda_1 L_{tcl} + \lambda_2 L_{tbo} + \lambda_3 L_{tdo} + \lambda_4 L_{tcc}, \]

where \( L_{tcl}, L_{tbo}, L_{tdo} \) and \( L_{tcc} \) represent the loss of TCL, TBO, TDO and TCC maps. We train TCL branch by minimizing the Dice loss (Milletari, Navab, and Ahmadi 2016), and the Smooth \( L_1 \) loss (Girshick 2015) is adopted for TBO and TDO map, while TCC map is trained with PG-CTC loss as mentioned before. The loss weights \( \lambda_1, \lambda_2, \lambda_3, \) and \( \lambda_4 \) are a trade-off between four tasks in this work, and we determine a set of values \{1.0, 1.0, 1.0, 5.0\} empirically.

3.5 Graph Refinement Module

Inspired by GTC (Hu et al. 2020) and SRN (Yu et al. 2020), we propose a slight graph refinement module to perceive the word-level semantic context and visual context information to improve the end-to-end reading performance with GCNs further.

The coarse recognition results is refined in text instance level, and we construct a visual reasoning graph and a semantic reasoning graph for a point sequence \( \pi \), of which the points are considered as the nodes in a graph. We use the same structure of graph convolution layer as (Wang et al. 2019c) and (Zhang et al. 2020). Especially, the \( F_{\text{visual}} \) and TCC map are both required as inputs. \( F_{\text{visual}} \) is the output feature of FPN, as illustrated in Fig.3. For a point sequence \( \pi = \{p_1, p_2, \ldots, p_N\} \), the adjacency matrix is defined as

\[ A_{ij} = 1 - D(p_i, p_j)/\max(A), \]

where \( D(p_i, p_j) \) is a L2 distance between \( p_i \) and \( p_j \), and each node is self-connected.

The structure of GRM is depicted in Fig.5, and the two numbers in brackets indicate \( d_{in} \) and \( d_{out} \) of a neural layer. In the semantic reasoning graph, the features \( X_s \) is obtained by point gathering operation from TCC map, and further embedded to \( X_s \) of shape \( N \times 256 \). With three graph layers, the input \( X_s \) is transformed to produce \( Y_s \) of shape \( N \times 64 \); In the visual reasoning graph, the features \( F_v \) is gathered from \( F_{\text{visual}} \) and its channel is transformed from 128 to 256 following convolutional layers to get \( X_v \). With a similar network, we can get the visual reasoning output \( Y_v \) of shape \( N \times 64 \); Finally, we concatenate \( Y_s \) and \( Y_v \), and treat it as a classification problem with several FC layers to produce the refined probability sequence, where the GRM is also optimized with CTC loss. It is worth mentioning that we padding the coarse sequence recognition to the same length and batched together for efficient training, and the max length is set to 64.

4 Experiments

4.1 Datasets

The benchmark datasets used for the experiments in this paper are briefly introduced below.

ICDAR 2015. The ICDAR 2015 dataset (Karatzas et al. 2015) is collected for the ICDAR 2015 Robust Reading Competition, with 1,000 natural images for training and 500 for testing. The text instances are annotated in word-level.

Total-Text. The Total-Text (Ch’ng and Chan 2017) is another curved text benchmark, which consists of 1,255 training images and 300 testing images with more than three different text orientations: Horizontal, Multi-Oriented, and Curved. The annotations are labeled in word-level.

4.2 Implementation Details

Training. The stem network is initialized with pre-trained weight on ImageNet (Deng et al. 2009). The training process is mainly divided into the warming-up step, fine-tuning step, and training step of the GRM module. In the warming-up step, we apply Adam optimizer to train our model with learning rate 1e-3, and the learning rate decay factor is 0.94 on the SynthText (Gupta, Vedaldi, and Zisserman 2016); In the fine-tuning step, the learning rate is re-initiated to 1e-3 and the model is tuned on ICDAR2015 and Total-Text data. The GRM is an additional module in our pipeline, and in the last step, we only train the learnable parameters in GRM module to provide both the metrics on ICDAR 2015 and Total-Text with/without GRM for a fair comparison.

The experiments are performed on a workstation with the following configuration, CPU: Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz x16; GPU: NVIDIA TITAN Xp × 4; RAM: 64GB. During the training time, we set the batch size to 12/18 per GPU in parallel for PGNet-A and PGNet-E.

4.3 Ablation Study

In this section, comprehensive experiments are conducted to prove the strengths of the proposed method.

The Effectiveness of TDO. The TDO map is adopted to recover the sequence information of text in non-traditional reading directions, which benefits both detection and recognition tasks. To verify the efficiency of TDO, we conduct several experiments in ICDAR2015 and Total-Text for PGNet-A and PGNet-E. As shown in Tab. 1, the benefits of TDO in Total-Text are obvious, and the recognition accuracy in PGNet-E obtains 1.6% improvement, while it is less than...
Table 1: Evaluation the effectiveness of TDO to improve the recognition performance.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>ICDAR2015</th>
<th>Total-Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>PGNet-A</td>
<td>PGNet-E</td>
</tr>
<tr>
<td>with TDO</td>
<td>62.3</td>
<td>57.4</td>
</tr>
<tr>
<td>w/o TDO</td>
<td>61.9</td>
<td>56.9</td>
</tr>
<tr>
<td>Gain</td>
<td>+0.4</td>
<td>+0.5</td>
</tr>
</tbody>
</table>

Table 2: Evaluation the effectiveness of GRM to improve the recognition performance.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>ICDAR2015</th>
<th>Total-Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>PGNet-A</td>
<td>PGNet-E</td>
</tr>
<tr>
<td>with GRM</td>
<td>62.5</td>
<td>58.7</td>
</tr>
<tr>
<td>w/o GRM</td>
<td>62.3</td>
<td>57.4</td>
</tr>
<tr>
<td>Gain</td>
<td>+1.2</td>
<td>+1.3</td>
</tr>
</tbody>
</table>

0.5% in ICDAR2015. We claim that the non-traditional texts are more likely to appear in the curved Total-Text, while the texts in ICDAR2015 are mostly quadrangles, and the predefined rules may cover most situations.

The Effectiveness of GRM. The GRM is proposed to model word-level semantic context implicitly and improve the end-to-end reading performance. To verify its efficiency, we conduct several experiments in ICDAR2015 and Total-Text on both PGNet-A and PGNet-E respectively. As depicted in Tab. 2, the efficiency of GRM is proved, and the end-to-end reading performance is improved significantly. Especially, it gains 2.2% for end-to-end recognition tasks for PGNet-E in Total-Text.

4.4 Evaluation for Curved Text

On Total-Text, we evaluate the performance of PGNet for spotting scene text of arbitrary shapes. We fine-tune our model with Total-Text training set and partial ICDAR2019 ArT (Chng et al. 2019), where we remove the images from the Total-Text test set by comparing the MD5 value of each image. The ratio of these two datasets is set to 3:2 during training. The GRM is further trained with SynthText, partial ICDAR2019 ArT, and Total-Text, of which the data ratio is set to 6:2:2, while the learnable parameters outside GRM are frozen. In the inference phase, the longer side of images is resized to 640 with the aspect ratio kept.

The results are shown in Tab. 3 and Fig. 6. Our method achieves 0.882 F-Measure on text detection tasks and is better than most of the previous methods. Besides, on the end-to-end recognition task, the PGNet-A with strong and weak lexicon achieves state-of-the-art accuracy. Particularly, our method exceeds the previous best one-stage text spotter CharNet R-50 by 3.2%, 3.8%, and 1.2% when evaluated with strong, weak, and general lexicon respectively.

4.5 Evaluation for Multi-oriented Text

We also conduct experiments on ICDAR2015 to confirm the superiority of the proposed method on the multi-oriented scene text. Following the same training strategy as Total-Text, The ICDAR2017-MLT-Latin and ICDAR2015 are used in fine-tune stage, of which the data ratio is 3:7, while SynthText, ICDAR2017-MLT-Latin (Nayef et al. 2017) and ICDAR2015 with a ratio of 6:2:2 are adopted for training GRM. In the test phase, the longer side of images is resized to 1536 with the aspect ratio kept.

The results are shown in Tab. 4 and Fig. 6. Our method achieves 0.882 F-Measure on text detection tasks and is better than most of the previous methods. Besides, on the end-to-end recognition task, the PGNet-A with strong and weak lexicon achieves state-of-the-art accuracy. Particularly, our method exceeds the previous best one-stage text spotter CharNet R-50 by 3.2%, 3.8%, and 1.2% when evaluated with strong, weak, and general lexicon respectively.

4.6 Runtime

In this paper, we propose a simple yet powerful arbitrarily-shaped text spotter, and considering the limitation of computing resources in different application scenarios, we propose PGNet-A and PGNet-E with different stem networks. The runtime of our method can be roughly divided into three parts: network inference stage, post-processing stage, and graph refinement stage. The runtime of PGNet on Total-Text is evaluated with NVIDIA Tesla V100-SXM2-16GB, which is the same as ABCNet. The longer side of the test image is resized to 640 and the batch size is set to 1 on a single GPU. It takes 15.1 ms, 6.4 ms, and 3.2 ms in the three stages respectively. It is worth noting that, the post-processing stage which accounted for a large proportion in time cost is executed with Python code and can be further optimized. In Total-Text, PGNet-E runs at 46.7 FPS with 84.8% and 58.4% F-Measure respectively on the detection and end-to-end recognition without lexicon, surpassing the previous arbitrarily-shaped text spotters in efficiency significantly, as depicted in Tab. 3.

5 Conclusion and Future Work

In this paper, we propose a real-time arbitrarily-shaped text spotter PGNet together with a novel graph refinement module to relieve the inefficient problem of previous methods. Optimizing TCC maps with proposed PG-CTC loss, our PGNet is free from character-level annotations and gets rid of NMS and RoI operations in the inference stage which is single-shot, simple, and efficient. Moreover, a novel graph refinement module is proposed to further improve the end-to-end performance by reasoning the word-level semantic information via a spatial graph neural network. The contributions of our paper are confirmed by comprehensive experiments and qualitative visualization in public benchmarks. In the future, we are interested in promoting the deployment of our novel real-time arbitrarily-shaped text spotter on edge smart devices.
Table 3: Evaluation on Total-Text for detecting text lines of arbitrary shapes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>Detection</th>
<th>Recognition</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOTS (Liu et al. 2018)</td>
<td>ResNet-50</td>
<td>38.0</td>
<td>52.3</td>
<td>44.0</td>
</tr>
<tr>
<td>Textboxes (Liao et al. 2017)</td>
<td>ResNet-50-FPN</td>
<td>45.5</td>
<td>62.1</td>
<td>52.5</td>
</tr>
<tr>
<td>Mask TextSpotter (Lyu et al. 2018)</td>
<td>ResNet-50-FPN</td>
<td>55.0</td>
<td>69.0</td>
<td>61.3</td>
</tr>
<tr>
<td>TextNet (Sun et al. 2018)</td>
<td>ResNet-50-SAM</td>
<td>59.5</td>
<td>68.2</td>
<td>63.5</td>
</tr>
<tr>
<td>TextSnake (Long et al. 2018)</td>
<td>VGG16-FPN</td>
<td>74.5</td>
<td>82.7</td>
<td>78.4</td>
</tr>
<tr>
<td>TextDragon (Feng et al. 2019)</td>
<td>VGG16-FPN</td>
<td>75.7</td>
<td>85.6</td>
<td>80.3</td>
</tr>
<tr>
<td>CharNet H-57 (Xing et al. 2019)</td>
<td>Hourglass-57</td>
<td>81.0</td>
<td>88.6</td>
<td>84.6</td>
</tr>
<tr>
<td>Boundary (Wang et al. 2020)</td>
<td>ResNet-50-FPN</td>
<td>85.0</td>
<td><strong>88.9</strong></td>
<td><strong>87.0</strong></td>
</tr>
<tr>
<td>ABCNet-F (Liu et al. 2020)</td>
<td>ResNet-50-FPN</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>Detection</th>
<th>Recognition</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGNet-A</td>
<td>ResNet-50-FPN</td>
<td><strong>86.8</strong></td>
<td>85.3</td>
<td>86.1</td>
</tr>
<tr>
<td>PGNet-A with GRM</td>
<td>ResNet-50-FPN</td>
<td><strong>86.8</strong></td>
<td>85.5</td>
<td>86.1</td>
</tr>
<tr>
<td>PGNet-E</td>
<td>EfficientNet-B0-FPN</td>
<td>85.0</td>
<td>84.6</td>
<td>84.8</td>
</tr>
<tr>
<td>PGNet-E with GRM</td>
<td>EfficientNet-B0-FPN</td>
<td>84.9</td>
<td>84.7</td>
<td>84.8</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Method</th>
<th>Detection</th>
<th>E2E Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST (Zhou et al. 2017)</td>
<td>78.3</td>
<td>13.8</td>
</tr>
<tr>
<td>LOMO (Zhang et al. 2019)</td>
<td>81.0</td>
<td>54.0</td>
</tr>
<tr>
<td>Mask TextSpotter (Lyu et al. 2018)</td>
<td>81.0</td>
<td>79.3</td>
</tr>
<tr>
<td>TextNet (Sun et al. 2018)</td>
<td>85.4</td>
<td>78.7</td>
</tr>
<tr>
<td>FOTS-R-50 (Liu et al. 2018)</td>
<td>85.2</td>
<td>81.1</td>
</tr>
<tr>
<td>TextDragon (Feng et al. 2019)</td>
<td>83.8</td>
<td>82.5</td>
</tr>
<tr>
<td>CharNet R-50 (Xing et al. 2019)</td>
<td><strong>88.3</strong></td>
<td><strong>82.5</strong></td>
</tr>
<tr>
<td>Boundary (Wang et al. 2020)</td>
<td>87.5</td>
<td>79.7</td>
</tr>
<tr>
<td>TextPerceptron (Qiao et al. 2020)</td>
<td>82.5</td>
<td>80.5</td>
</tr>
<tr>
<td>PGNet-A</td>
<td>84.3</td>
<td>82.9</td>
</tr>
<tr>
<td>PGNet-A with GRM</td>
<td>84.8</td>
<td>83.3</td>
</tr>
<tr>
<td>PGNet-E</td>
<td>83.6</td>
<td>82.0</td>
</tr>
<tr>
<td>PGNet-E with GRM</td>
<td>83.6</td>
<td>80.5</td>
</tr>
</tbody>
</table>

Figure 6: Qualitative results of our method on Total-Text (left two columns) and ICDAR15 (right two columns) datasets.
6 Acknowledgments

This work is done when Pengfei Wang is an intern at Baidu Inc., and is also supported in part by the National Natural Science Foundation of China (NSFC) under Grant Nos. 61572387, 61632019, 61871304, and the Foundation for Innovative Research Groups of the NSFC under Grant 61621005.

References


