

TextDragon: An End-to-End Framework for Arbitrary Shaped Text Spotting

Wei Feng^{1,2} Wenhao He^{1,2} Fei Yin^{1,2} Xu-Yao Zhang^{1,2} Cheng-Lin Liu^{1,2,3}

¹ National Laboratory of Pattern Recognition (NLPR),

Institute of Automation of Chinese Academy of Sciences, Beijing 100190, China

² School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China

³ CAS Center for Excellence of Brain Science and Intelligence Technology, Beijing 100190, China

Email: {wei.feng, wenhao.he, fyi.n, xyz, liliucl}@nlpr.ia.ac.cn

Abstract

Most existing text spotting methods either focus on horizontal/oriented texts or perform arbitrary shaped text spotting with character-level annotations. In this paper, we propose a novel text spotting framework to detect and recognize text of arbitrary shapes in an end-to-end manner, using only word/line-level annotations for training. Motivated from the name of TextSnake [32], which is only a detection model, we call the proposed text spotting framework TextDragon. In TextDragon, a text detector is designed to describe the shape of text with a series of quadrangles, which can handle text of arbitrary shapes. To extract arbitrary text regions from feature maps, we propose a new differentiable operator named RoISlide, which is the key to connect arbitrary shaped text detection and recognition. Based on the extracted features through RoISlide, a CNN and CTC based text recognizer is introduced to make the framework free from labeling the location of characters. The proposed method achieves state-of-the-art performance on two curved text benchmarks CTW1500 and Total-Text, and competitive results on the ICDAR 2015 Dataset.

1. Introduction

Scene text spotting aims to detect and recognize text in an image. It has attracted increasing attention in recent years, due to its wide applications in document analysis and scene understanding. Although previous methods [29, 33, 25, 3] have made significant progress on datasets where text boundary is labeled by quadrangles or rectangles, text spotting for arbitrary shapes is still challenging for both detection and recognition.

Most existing methods perform text spotting through two independent steps: a detector is firstly employed to detect all texts in the image, and then text recognition is conducted on the detected regions. The disadvantages of these

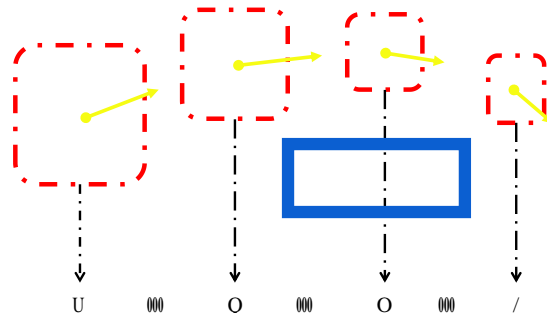
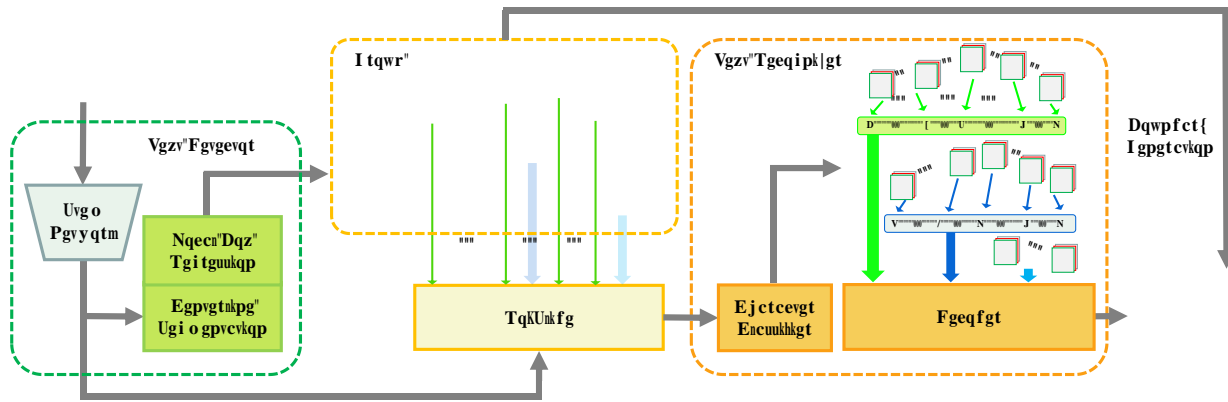


Figure 1. The reading mechanism of humans. Red boxes show the areas of fixations, and yellow arrows indicate the directions of eye movement. Black arrows indicate the recognition results, in which “-” means blank. Green and blue lines show text boundaries.

methods lie in the heavy time cost and correlation ignorance between text detection and recognition. Thus, several methods [29, 25, 3] are proposed recently to unify horizontal/oriented text detection and recognition in an end-to-end manner. However, scene texts in the real-world often appear in arbitrary shapes. Instead of describing text with quadrangles or rectangles, TextSnake [32] describes text with a series of local units, which behaves like a snake. However, this work mainly focuses on curved text detection. Lyu *et al.* [33] proposed Mask Textspotter to detect and recognize text of arbitrary shapes, in which a character-segmentation based text recognizer is used, and thus character-level annotations are required in the training process. Nevertheless, most datasets do not have character-level annotations, which require much more human labeling efforts.

To achieve the text spotting of arbitrary shapes, we can follow the reading mechanism of human [1] as shown in Fig 1. First, a local area of the text is detected. After that, the contents of the local area are recognized. Finally, the eyes move along the centerline of the text and repeat the



words. Deep learning based methods extract features from the whole image by CNN [12], and then employ Recurrent Neural Network (RNN) to generate sequential labels [42]. However, these methods regard texts as one-dimensional sequences, which are not suitable for curved text recognition.

To handle curved text, Shi *et al.* [39] and Liu *et al.* [28] introduced the spatial attention mechanism to transform the curved text into a suitable pose for recognition. Cheng *et al.* [5] proposed the arbitrary orientation network to deal with irregular texts, in which the features are combined into an attention-based decoder.

2.3. Scene Text Spotting

Most existing methods treat scene text spotting [19, 26] as two separate steps: the first step is to detect text lines and the second step is to recognize them. However, the multitude of steps may require exhaustive tuning, leading to sub-optimal performance and time consumption.

Recently, Li *et al.* [25] proposed an end-to-end text spotter which focuses on horizontal texts. Liu *et al.* [29] introduced a differentiable operator RoIRotate, which extracts oriented text regions from feature maps. Patel *et al.* [35] proposed an end-to-end method for multi-language text spotting and script identification. However, these methods can only deal with horizontal or oriented texts. On the basis of Mask-RCNN [11], Lyu *et al.* [33] detect and recognize text instances of arbitrary shapes by segmenting the text regions and character regions. Unlike [33], our method does not need character-level annotations. Moreover, the anchor mechanism in [33] may not be able to generate suitable text proposals as the shape of curved text is highly variable.

3. Methodology

Our proposed end-to-end scene text spotting framework is shown in Fig 2. First, a stem network is used to extract visual features on the input image. After feature extraction, the text detector is applied to describe each text with a series of quadrangles, which locate along the centerline. Then, the newly proposed RoISlide extracts features along each text centerline from feature maps, in which a local transformer network converts features in each quadrangle into rectified ones. Finally, the CNN based text recognizer predicts the category of each quadrangle, and decodes the sequenced result using a CTC decoder. The modules of text detection and recognition are trained jointly on images of texts without character-level annotations. In the following, we will introduce the details of the detector, RoISlide, recognizer and inference procedure.

3.1. Text Detection

To detect text with arbitrary shapes, we adopt similar idea in TextSnake [32] by predicting local geometry at-

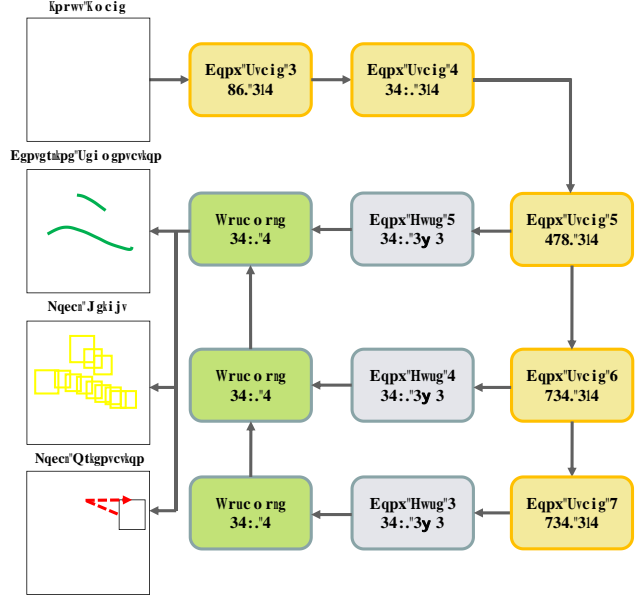


Figure 3. The architecture of text detector. “Conv Stages” 1-5 are from VGG-16, and “Upsample” represents a deconvolution layer of 128 channels with stride 2. We only show part of the bounding boxes in the Local Height branch for better visualization, but actually the bounding boxes are much more dense.

tributes of text shown in Fig 3. However, TextSnake uses disks to represent local geometric attributes, which is difficult for the subsequent feature extraction, thus text is represented as a series of quadrangles here. To detect texts of a wide scale range, we merge feature maps from different levels and upscale the fused feature map to 1/4 size of the input image. The output module consists of two tasks: Centerline Segmentation and Local Box Regression. The output maps of the two tasks are also 1/4 size of the input image. The rest of this subsection will introduce details of the two tasks.

Centerline Segmentation. This task is to extract text regions from images and can be deemed as down-sampled pixel-level classification between text (positive category) and non-text (negative category). Instead of segmenting all pixels within the text region out, here we only regard the centerline region which is a shrunk version of the original text area as positive. The eroded text region is to relieve the touching effect when texts are close to each other and brings better local units grouping results in the inference stage. To alleviate the negative influence of class imbalance between text and non-text, online hard example mining (OHEM) introduced in [40] is adopted during the training stage.

The loss function L_{seg} for Centerline Segmentation task is derived from the cross entropy loss. Denote the set of selected elements by OHEM as S , L_{seg} is formulated as:

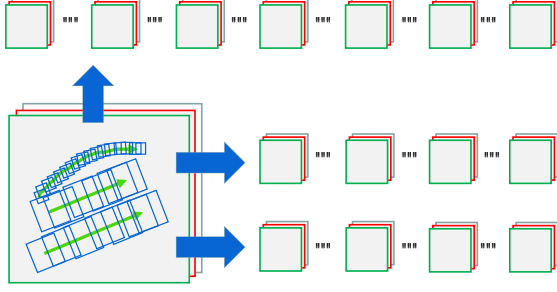


Figure 4. The illustration of RoISlide. The green arrows indicate the direction of sliding, and the blue arrows indicate the results of each quadrangle through RoISlide. We show the process of RoISlide on the input image for better visualization, but actually the operation is on the feature maps.

$$\begin{aligned} L_{\text{seg}} &= \frac{1}{|S|_s} L(p_s, p_s) \\ &= \frac{1}{|S|_s} (-p_s \log p_s - (1 - p_s) \log (1 - p_s)), \end{aligned} \quad (1)$$

where $|\cdot|$ is the number of element in a set, and $L(p_s, p_s)$ refers to the cross entropy loss between the predicted pixel-level text score p_s and the corresponding ground truth p_s ($p_s \in \{0, 1\}$).

Local Box Regression. This task is to depict the local quadrangles of each text to facilitate the further curved boundary generation and text recognition. Each local quadrangle is represented by two geometry attributes in this task. The first attribute is the local text height, which is represented by a squared box as shown in Fig 3, where the box side length is equal to the local text height. The second attribute is the local text orientation, which is the tangent angle of the curved text as shown in Fig 3.

Given a pixel i located in the positive area P in Centerline Segmentation task, the loss function L_{reg} for Local Box Regression task is formulated as:

$$\frac{L_B}{L} = \frac{1}{|P|_i} \text{Smooth}_{L_1} \frac{B_i - B_i}{i - i}, \quad (2)$$

$$L_{\text{reg}} = L_B + L, \quad (3)$$

where B_i and i are predicted local squared box and angle, respectively, while B_i and i are the corresponding ground truth. L is a hyper-parameter which is set to 10 in our experiments. We choose Smooth_{L_1} loss [36] here for its robustness to variation in object shape.

3.2. RoISlide

Text shapes in the real-world vary tremendously, therefore, it is difficult to directly transform the whole text fea-

tures into axis-aligned features. As the text detection branch has transformed the shape of text into a series of quadrangles, RoISlide is proposed to transform the whole text features into axis-aligned features indirectly by transforming each local quadrangle sequentially, which is the key to make the framework end-to-end trainable. Specifically, the RoISlide has two steps: firstly, we arrange quadrangles which are distributed along the text centerline in order. Then a Local Transformer Network (LTN) is proposed to transform feature maps cropped from each quadrangle into rectified ones in a sliding manner. After the above two steps, arbitrary shaped text features are converted into sequenced squared feature maps of identical dimension as shown in Fig 4.

Denote the sequenced quadrangles after the first step as $R = \{R_1, R_2, \dots, R_N\}$. For each quadrangle R_n , the LTN converts features in R_n into a unified spatial dimension which is $H \times H$, and we set H to 8 in the experiment. To achieve this goal, the LTN first computes a series of affine transformation matrices $M = \{M_1, M_2, \dots, M_N\}$ using features in R , which contain 6-dimensional parameters. The LTN consists of two convolution + max pooling layers, followed by two fully-connected layers for the regression of the transformation parameters. It should be noticed that the LTN is trained jointly with other modules without the need of position supervision.

After that, a grid generator generates a sampling grid to perform a warping of the input features. The point-wise affine transformation can be written as:

$$\begin{pmatrix} x_c^s \\ y_c^s \end{pmatrix} = M_n \begin{pmatrix} x_c^t \\ y_c^t \end{pmatrix} = \begin{pmatrix} 11 & 12 & 13 \\ n & n & n \\ 21 & 22 & 23 \\ n & n & n \end{pmatrix} \begin{pmatrix} x_c^t \\ y_c^t \end{pmatrix}, \quad (4)$$

where (x_c^s, y_c^s) and (x_c^t, y_c^t) represent the coordinate of a point on the shared feature maps and transformed feature maps respectively.

Finally, the RoI features are extracted from the shared feature maps with the set of sampling points, in which the interpolation method is bilinear interpolation.

Spatial transformer network (STN) [20] also uses affine transformation, which is mainly focused on transforming whole images. Different from STN, the proposed LTN takes local feature maps as input, and the transformed local feature maps make up the whole text features, which is more domain-specific for the arbitrary shaped text spotting.

3.3. Text Recognition

Although the shapes of texts are arbitrary, humans' eyes always read along the centerline. Meanwhile, their eyes do not move continuously, but make short rapid movements. Inspired by [45, 48], we adopt the sliding convolution character models rather than traditional LSTM [16] to recognize texts of arbitrary shapes for fast recognition. Unlike [45, 48] which extract features from the original images, we predict

Table 1. The network architecture of the text recognizer. Each convolutional layer is followed by a batch normalization layer and a ReLU layer. W is the number of character classes.

Type	Configurations
Input	$N \times 8 \times 8$
Conv.bn.relu	3×3 , 128, stride 1×1
Conv.bn.relu	3×3 , 128, stride 1×1
Max Pooling	2×2 , stride 2×2
Conv.bn.relu	3×3 , 256, stride 1×1
Conv.bn.relu	3×3 , 256, stride 1×1
Max Pooling	2×2 , stride 2×2
Conv.bn.relu	3×3 , 512, stride 1×1
Conv.bn.relu	3×3 , 512, stride 1×1
Fully Connection	256, drop: 0.5
Fully Connection	W

text labels with the features transformed by RoISlide. The text recognition branch consists of two parts: a character classifier and a transcription layer. The character classifier predicts a label distribution from the input squared features, and the transcription layer decodes each square's predictions into the final sequence.

As the input features are transformed from the shared feature maps, which contain rich semantic information, we replace the network in [48] with a simpler one as shown in Table 1. Each convolutional layer is followed by a batch normalization [17] (BN) layer for fast convergence. After every two convolutional layers, max-pooling layer is used to halve the size of feature maps. The final convolutional output is flattened into a vector of length 2048, and then fed into the following two fully-connected layers. To avoid overfitting, we insert one dropout layer after the first fully-connected layer. Finally, we use the softmax layer to get each square's label distribution.

In the transcription layer, we adopt the CTC decoder [9] and assume that each squared features after RoISlide represents a time step. The CTC decoder aims to transform each square's probability distribution to the label sequence. Denote the probability distribution at step n as $P(k|n)$ and \mathbf{X} is a CTC path, whose sequence length is equal to the number of squares N . The probability $P(\mathbf{X}|\mathbf{X})$ can be written as:

$$P(\mathbf{X}|\mathbf{X}) = \prod_{n=1}^N P(n|\mathbf{X}). \quad (5)$$

Then, a CTC mapping function \mathbf{B} is used to remove repetitions and delete the blank. The conditional probability of the ground truth y is the sum of the probabilities of all the paths by \mathbf{B} :

$$P(y|\mathbf{X}) = \sum_{\mathbf{B}^{-1}(y)} P(\mathbf{X}), \quad (6)$$

and the objective is to maximize the log likelihood of con-

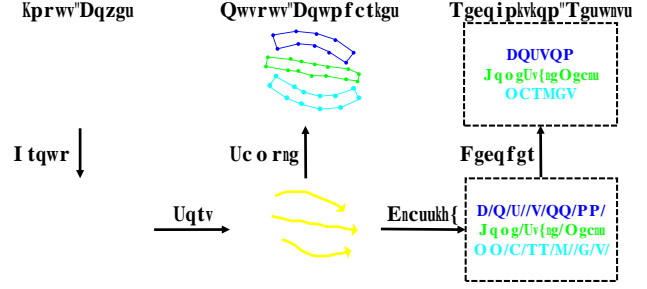


Figure 5. The procedure of inference. Red quadrangles show the input boxes of the inference stage. After grouping, quadrangles with the same color represent the same group. Yellow arrows indicate the directions of sorting. In the output boundaries, dots show the vertices after sampling. In the classification result, “-” means blank.

ditional probability of ground truth. Denote the number of words in the input image as M . The loss for text recognition L_{rec} can be written as:

$$L_{rec} = -\frac{1}{M} \sum_{m=1}^M \log p(y|\mathbf{X}). \quad (7)$$

For end-to-end training of text detection and recognition, the whole loss function can be formulated as:

$$L = L_{seg} + \alpha_{reg} L_{reg} + \alpha_{rec} L_{rec}, \quad (8)$$

where α_{reg} and α_{rec} are the hyper-parameters to control the balance among each task.

3.4 Inference

In the inference stage, our goal is to provide the text boundary through detector, as well as the text content together with recognizer. We first apply thresholding to the text centerline segmentation map, and then NMS is applied to the local bounding boxes to reduce redundancy. Finally, four steps are conducted based on the quadrangles produced by NMS to get the final results as shown in Fig 5.

Group. Given the input boxes, instead of using the disjoint-set as introduced in TextSnake [32], we group bounding boxes according to their geometric relation. In order to make the connected component complete, full resolution and low threshold are used in TextSnake, which will lead to more noise and heavy time consumption, but we can use a higher threshold and quarter resolution output to avoid these problems. Each box in a group should satisfy two heuristic conditions with at least other one box in the same group: (1) Their IoU should be higher than 0.5; (2) their absolute angle difference should be less than $\pi/4$.

Sort. After grouping bounding boxes, we sort the boxes for boundary generation and text recognition. First, we judge whether the overall direction is horizontal or vertical according to the average angle of all boxes within the same

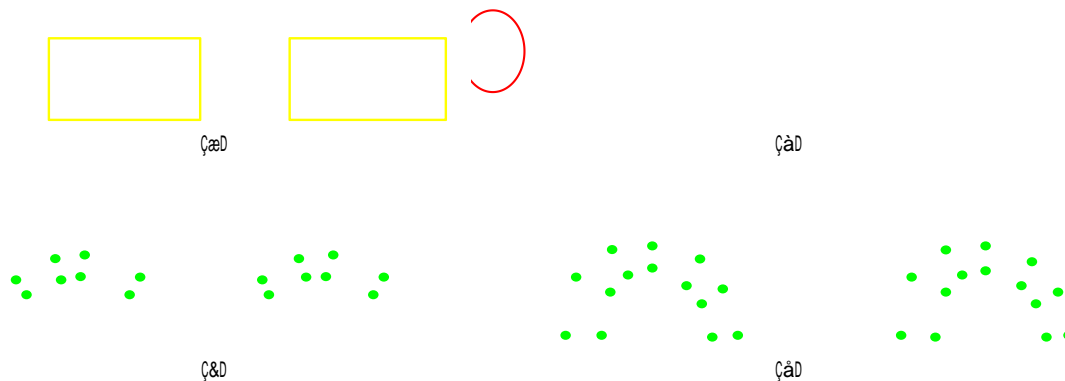


Figure 6. (a, b) showing that end-to-end training helps text detection. (c, d) showing that RoIRotate leads to wrong recognition results for curved text. In the first row, from left to right: detection without guidance from recognition and TextDragon. In the second row, from left to right: with RoIRotate and with RoISlide. Green dots show the vertexes of polygons.

group. Then boxes are sorted from left to right (horizontal) or from top to bottom (vertical).

Sample. For boundary generation, we just evenly sample the ordered boxes to form the vertexes of polygons. Then the boundary of text is generated by linking the vertexes sequentially.

Recognize. For text recognition, we first perform RoISlide on the shared feature maps with the ordered boxes. Then each transformed features are classified by the character classifier. Finally, the CTC decoder is used to predict the final recognition result.

4. Experiments

We evaluate both text spotting and detection performance of the proposed method on several standard benchmarks. Moreover, analysis of each module and comparisons with previous methods are also given to demonstrate the superiority and reasonableness of TextDragon.

4.1. Datasets

CTW1500. The CTW1500 dataset contains 1000 training images and 500 test images. Each image has at least one curved text. Horizontal and multi-oriented texts are also contained in this dataset. Each text is labeled as a polygon with 14 vertexes in line-level. The evaluation protocol of end-to-end recognition is similar to ICDAR 2015, where quadrangles are changed to arbitrary polygons. We report the end-to-end recognition results over two lexicons: “None” and “Full”. “None” means that no lexicons are provided, and “Full” lexicon provides all words in the test set. **Total-Text.** The Total-Text dataset has 1255 training images and 300 test images, which contains curved text, as well as horizontal and multi-oriented text. Each text is labeled as a polygon in word-level, and the evaluation proto-

col of end-to-end recognition follows that for CTW1500.

ICDAR 2015. The ICDAR 2015 dataset contains 1000 training images and 500 test images. Each text is labeled as a quadrangle with 4 vertexes in word-level. The text spotting task reports results over three lexicons: “Strong”, “Weak” and “Generic”. Strong lexicon provides 100 words that may appear in each image. Weak lexicon provides words in the whole test set, and generic lexicon provides a 90K vocabulary.

4.2. Implementation Details

Our implementation is based on Caffe framework [22]. The stem network VGG-16 [41] inherits parameters trained on ImageNet dataset [24], and then we pre-train the model on SynthText [10] for 600k iterations, and fine-tune on other datasets for 120k iterations. The input images of 512×512 are cropped from images after random scaling and rotation. We first set the loss weight $_{reg}$ and $_{rec}$ to 0.01, and then we raise them to 0.1 and 0.05 respectively after the segmentation task is well optimized. In the pre-training stage, the learning rate is 0.01. During the fine-tuning stage, we train the model with a learning rate of 0.001. Experiments are implemented on a workstation with 2.9GHz 12-core CPU, 256G RAM, GTX Titan X and Ubuntu 64-bit OS.

4.3. Ablation Studies

For better understanding the strengths of the proposed method, we first provide the ablation studies from three aspects. First, we demonstrate the benefits of end-to-end training. Second, we compare RoISlide and RoIRotate on recognition performance under different text shapes. Third, we compare the proposed CNN based recognizer with the more popularized LSTM based one.

Table 2. Results on CTW1500 test set.

Method	Detection			End-to-End	
	P	R	F	None	Full
SegLink [37]	42.3	40.0	40.8	-	-
EAST [49]	78.7	49.1	60.4	-	-
DMPNet [30]	69.9	56.0	62.2	-	-
FOTS [29]	79.5	52.0	62.8	21.1	39.7
CTD [31]	74.3	65.2	69.5	-	-
CTD+TLOC [31]	77.4	69.8	73.4	-	-
TextSnake [32]	67.9	85.3	75.6	-	-
Our Two-Stage	79.5	81.0	80.2	37.2	69.9
With RoIRotate	80.7	83.4	82.3	38.6	70.9
With LSTM	84.3	81.8	83.0	39.2	71.5
TextDragon	84.5	82.8	83.6	39.7	72.4

Table 3. Results on Total-Text test set.

Method	Detection			End-to-End	
	P	R	F	None	Full
SegLink [37]	30.3	23.8	26.7	-	-
Ch'ng <i>et al.</i> [6]	40.0	33.0	36.0	-	-
EAST [49]	50.0	36.2	42.0	-	-
FOTS [29]	52.3	38.0	44.0	32.2	35.9
Liao <i>et al.</i> [27]	62.1	45.5	52.5	36.3	48.9
Mask TextSpotter [33]	69.0	55.0	61.3	52.9	71.8
TextSnake [32]	82.7	74.5	78.4	-	-
Our Two-Stage	84.5	74.2	79.0	46.1	70.6
With RoIRotate	86.0	74.3	79.7	47.1	73.6
With LSTM	85.2	75.7	80.2	48.3	74.7
TextDragon	85.6	75.7	80.3	48.8	74.8

Table 4. Results on ICDAR 2015 test set. “P”, “R”, “F” represent “Precision”, “Recall”, “F-measure” respectively. “S”, “W”, “G” represent recognition with “Strong”, “Weak”, “Generic” lexicon respectively. “*” indicates that the method is designed for text of arbitrary shapes.

Method	Detection			Method	End-to-End			Word Spotting		
	P	R	F		S	W	G	S	W	G
SegLink [37]	74.74	76.50	75.61	Baseline OpenCV3.0 [23]	13.84	12.01	8.01	14.65	12.63	8.43
EAST [49]	83.27	78.33	80.72	Stradvision [23]	43.7	-	-	45.9	-	-
He <i>et al.</i> [15]	82.0	80.0	81.0	TextProposals [8, 18]	53.3	49.6	47.2	56.0	52.3	49.7
TextSnake [32]	84.9	80.4	82.6	HUST_MCLAB [37, 38]	67.9	-	-	70.6	-	-
PixelLink [7]	85.5	82.0	83.7	Deep text spotter [3]	54.0	51.0	47.0	58.0	53.0	51.0
Mask TextSpotter [33]	91.6	81.0	86.0	Mask TextSpotter [33]	79.3	73.0	62.4	79.3	74.5	64.2
He <i>et al.</i> [14]	87.0	86.0	87.0	He <i>et al.</i> [14]	82.0	77.0	63.0	85.0	80.0	65.0
FOTS [29]	91.85	87.92	89.84	FOTS [29]	83.55	79.11	65.33	87.01	82.39	67.97
Our Detection	84.82	81.82	83.05	Our Two-Stage	75.23	73.15	53.04	77.03	75.11	54.51
With RoIRotate	92.18	82.93	87.31	With RoIRotate	82.51	79.21	65.37	86.20	82.03	68.14
TextDragon	92.45	83.75	87.88	TextDragon	82.54	78.34	65.15	86.22	81.62	68.03

4.3.1 Spotting with vs. without End-to-End Training

The recognition supervision can provide more detailed text stroke features for text detection. Without end-to-end training, text detection may miss some text regions or misclassify text-like background. To demonstrate the importance of end-to-end training, we evaluate a variant of our method in which text detection and recognition are trained separately. As shown in Tables 2, 3 and 4, the end-to-end training based methods (including TextDragon and other configurations) outperform our two-stage method significantly in text detection and end-to-end recognition. Furthermore, using different text recognizers in end-to-end training shows similar performances in text detection. It demonstrates that the text recognition supervision can provide text stroke features for text detection no matter which kinds of text recognizers are used.

Some qualitative results are shown in Fig 6. In the Fig 6(a), with end-to-end training, text whose feature is not salient could also be detected. In the Fig 6(b), the flag which has repetitive structured stripes is well classified when adopting the recognition task.

4.3.2 RoISlide vs. RoIRotate

The RoIRotate operator [29] aims to transform features with affine transformation in a similar way, in which the transformation parameters are calculated from the detection results. However, the RoISlide operator gets transformation parameters by the proposed LTN. Through analyzing the recognition results, we argue that RoIRotate may be unsuitable for curved text, because the characters at the junction of two quadrangles will rotate with two different , causing difficulties in recognizing characters. Two examples are shown in Fig 6. The character “A” in the Fig 6(c) and the character “E” in the Fig 6(d) are mis-classified with RoIRotate, as they are located at the junction of two quadrangles.

Therefore, we use the transformation parameters predicted by the LTN rather than detection results. To further explore the impact of RoIRotate on text of different shapes, we evaluate a variant of our method with RoIRotate for curved and oriented text. Tables 2 and 3 show that using RoIRotate for curved text will reduce the end-to-end recognition performance, indicating that getting transformation parameters from detection results is not suitable for curved

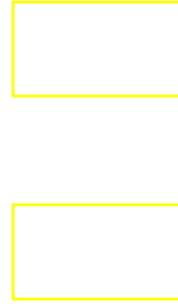


Figure 7. Examples of word spotting results. First column: CTW1500; second column: Total-Text; third column: ICDAR 2015.

text recognition. Although the orientation of each character in oriented text is consistent, Table 4 shows that RoISlide and RoIRotate achieve similar performance on oriented text, which verifies the generalization ability of RoISlide.

4.3.3 Spotting with vs. without LSTM

Most previous text recognition methods are based on LSTM, which predict classification results sequentially. But our text recognition model can classify each local box in parallel instead of sequential prediction like LSTM, and therefore our method is faster than those based on LSTM. To verify the effectiveness of CNN based recognizer, we evaluate a variant of our method which adds a bi-directional LSTM before the fully-connected layer in the text recognition branch, with $D = 256$ output channels per direction. As shown in Tables 2 and 3, adopting LSTM has little influence on end-to-end recognition performance. However, the speed of the CNN based text recognizer is **four times** faster than the LSTM based one (3 ms vs 12 ms), which demonstrates the advantages of CNN based recognizer.

4.4 Comparison with the State-of-the-art

In this subsection, we compare with previous methods on several benchmarks to verify the superiority of our work.

4.4.1 Experiments on Curved Text

As shown in Tables 2 and 3, the proposed method can achieve state-of-the-art performance on both CTW1500 and Total-Text. With the help of end-to-end training, TextDragon outperforms TextSnake on both datasets in the text detection task by a larger gap. For the end-to-end recognition task, TextDragon achieves state-of-the-art performance over full lexicon on Total-Text. Although the performance is inferior to Mask TextSpotter without lexicon, it is worth noting that our method does not need any character-level annotations, which owns much more practical value.

4.4.2 Experiments on Oriented Text

The results for oriented text are shown in Table 4. Compared with other methods designed for curved text, our method performs better on both text detection and end-to-end recognition. Meanwhile, our method achieves competitive results to the state-of-the-art approaches which are particularly designed and suitable for horizontal or oriented text. We also reimplement FOTS [29] on CTW1500 and Total-Text as shown in Tables 2 and 3. The results indicate that our method outperforms FOTS significantly on curved text, which demonstrates that TextDragon is suitable for both curved and oriented texts.

Some word spotting results in Fig 7 show that the proposed method can handle texts of arbitrary shapes .

5. Conclusion

In this paper, we propose a novel end-to-end scene text spotter to detect and recognize the text of arbitrary shapes. The text detector describes text with a series of quadrangles, which can adapt to complex shapes. A differentiable operator RoISlide is introduced to extract arbitrary text regions from feature maps. This is the key to unify text detection and recognition into an end-to-end pipeline. A text recognizer based on CNN classifier and CTC decoder is proposed to make the framework efficient and free from character-level annotations. Experiments on standard benchmarks have demonstrated the effectiveness of our method.

Acknowledgement

This work has been supported by the National Natural Science Foundation of China (NSFC) Grants 61733007, 61721004, 61633021, 61836014, Beijing Science and Technology Program Grant Z181100008918010, and NVIDIA NVAIL program..

References

- [1] Wikipedia. eye movement in reading — wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Eye_movement_in_reading.
- [2] Alessandro Bissacco, Mark Cummins, Yuval Netzer, and Hartmut Neven. Photoocr: Reading text in uncontrolled conditions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 785–792. IEEE, 2013.
- [3] Michal Bušta, Lukáš Neumann, and Jiri Matas. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2231. IEEE, 2017.
- [4] Huizhong Chen, Sam S Tsai, Georg Schroth, David M Chen, Radek Grzeszczuk, and Bernd Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 2609–2612. IEEE, 2011.
- [5] Zhanzhan Cheng, Xuyang Liu, Fan Bai, Yi Niu, Shiliang Pu, and Shuigeng Zhou. Arbitrarily-oriented text recognition. In *arXiv preprint arXiv:1711.04226*, 2017.
- [6] Chee Kheng Ch'ng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 935–942. IEEE, 2017.
- [7] Dan Deng, Haifeng Liu, Xuelong Li, and Deng Cai. Pixellink: Detecting scene text via instance segmentation. In *AAAI Conference on Artificial Intelligence*, 2018.
- [8] Lluís Gómez and Dimosthenis Karatzas. Textproposals: a text-specific selective search algorithm for word spotting in the wild. *Pattern recognition*, 70:60–74, 2017.
- [9] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 369–376. ACM, 2006.
- [10] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2315–2324, 2016.
- [11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE, 2017.
- [12] Pan He, Weilin Huang, Yu Qiao, Chen Change Loy, and Xiaoou Tang. Reading scene text in deep convolutional sequences. In *AAAI Conference on Artificial Intelligence*, volume 16, pages 3501–3508, 2016.
- [13] Tong He, Weilin Huang, Yu Qiao, and Jian Yao. Text-attentional convolutional neural network for scene text detection. *IEEE Transactions on Image Processing (TIP)*, 25(6):2529–2541, 2016.
- [14] Tong He, Zhi Tian, Weilin Huang, Chunhua Shen, Yu Qiao, and Changming Sun. An end-to-end textspotter with explicit alignment and attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5020–5029, 2018.
- [15] Wenhao He, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Deep direct regression for multi-oriented scene text detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 745–753, 2017.
- [16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *arXiv preprint arXiv:1502.03167*, 2015.
- [18] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *arXiv preprint arXiv:1406.2227*, 2014.
- [19] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision (IJCV)*, 116(1):1–20, 2016.
- [20] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, page 2008–2016, 2015.
- [21] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Deep features for text spotting. In *European Conference on Computer Vision (ECCV)*, pages 512–528. Springer, 2014.
- [22] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.
- [23] Dimosthenis Karatzas, Lluís Gomez-Bigorda, Angelos 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 *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015.
- [24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.
- [25] Hui Li, Peng Wang, and Chunhua Shen. Towards end-to-end text spotting with convolutional recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5238–5246, 2017.
- [26] Minghui Liao, Baoguang Shi, and Xiang Bai. Textboxes++: A single-shot oriented scene text detector. *IEEE Transactions on Image Processing*, 27(8):3676–3690, 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 *AAAI Conference on Artificial Intelligence*, pages 4161–4167, 2017.
- [28] Wei Liu, Chaofeng Chen, Kwan-Yee K Wong, Zhizhong Su, and Junyu Han. Star-net: A spatial attention residue network for scene text recognition. In *BMVC*, volume 2, page 7, 2016.

- [29] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. Fots: Fast oriented text spotting with a unified network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5676–5685, 2018.
- [30] Yuliang Liu and Lianwen Jin. Deep matching prior network: Toward tighter multi-oriented text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3454–3461, 2017.
- [31] Yuliang Liu, Lianwen Jin, Shuaitao Zhang, and Sheng Zhang. Detecting curve text in the wild: New dataset and new solution. In *arXiv preprint arXiv:1712.02170*, 2017.
- [32] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *European Conference on Computer Vision (ECCV)*, pages 19–35. Springer, 2018.
- [33] 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 *European Conference on Computer Vision (ECCV)*, September 2018.
- [34] Anand Mishra, Karteek Alahari, and CV Jawahar. Top-down and bottom-up cues for scene text recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012.
- [35] Yash Patel, Michal Buřta, and Jiri Matas. E2e-mlt - an unconstrained end-to-end method for multi-language scene text. In *arXiv preprint arXiv:1801.09919*, 2018.
- [36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.
- [37] Baoguang Shi, Xiang Bai, and Serge Belongie. Detecting oriented text in natural images by linking segments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2550–2558, 2017.
- [38] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304, 2017.
- [39] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4168–4176, 2016.
- [40] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 761–769, 2016.
- [41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [42] Bolan Su and Shijian Lu. Accurate scene text recognition based on recurrent neural network. In *Proceedings of the IEEE Asian Conference on Computer Vision (ACCV)*, pages 35–48. Springer, 2014.
- [43] Shangxuan Tian, Yifeng Pan, Chang Huang, Shijian Lu, Kai Yu, and Chew Lim Tan. Text flow: A unified text detection system in natural scene images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4651–4659, 2015.
- [44] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1457–1464. IEEE, 2011.
- [45] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 3304–3308. IEEE, 2012.
- [46] Xiaobing Wang, Yingying Jiang, Zhenbo Luo, Cheng-Lin Liu, Hyunsoo Choi, and Sungjin Kim. Arbitrary shape scene text detection with adaptive text region representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [47] Qixiang Ye and David S. Doermann. Text detection and recognition in imagery: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(7):1480–1500, 2015.
- [48] Fei Yin, Yi-Chao Wu, Xu-Yao Zhang, and Cheng-Lin Liu. Scene text recognition with sliding convolutional character models. In *arXiv preprint arXiv:1709.01727*, 2017.
- [49] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: An efficient and accurate scene text detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5551–5560, 2017.
- [50] Yingying Zhu, Cong Yao, and Xiang Bai. Scene text detection and recognition: Recent advances and future trends. *Frontiers of Computer Science*, 10(1):19–36, 2016.