Detecting Oriented Text in Natural Images by Linking Segments

Baoguang Shi* Xiang Bai* Serge Belongie†
*School of EIC, Huazhong University of Science and Technology
†Dept. of Computer Science, Cornell Tech
shibaoguang@gmail.com xbai@hust.edu.cn sjb344@cornell.edu

Abstract

Most state-of-the-art text detection methods are specific to horizontal text in Latin scripts and are not fast enough for real-time applications. We introduce Segment Linking (SegLink), an oriented text detection method. The main idea is to decompose text into two locally detectable elements, namely segments and links. A segment is an oriented bounding box that covers a part of a word or text line; A link connects two adjacent segments, indicating that they belong to the same word or line. Both elements are detected densely at multiple scales by an end-to-end trained, fully-convolutional neural network. Final detections are the combinations of segments that are connected by links. Compared with previous methods, our method improves along the dimensions of accuracy, speed and ease of training. It achieves an f-measure of 75.0% on the standard ICDAR 2015 Incidental *(Challenge 4) benchmark, outperforming the previous best by a large margin. It runs at over 20 FPS on 512×512 input images. In addition, our method is able to detect non-Latin text in long lines.*

Figure 1. **Illustration of SegLink**. The first row shows an image with two words of different scales and orientations. (a) Segments (yellow oriented boxes) detected on the image. (b) Links (green lines) detected between pairs of adjacent segments. (c) Segments connected by links are combined into whole words. (d-f) The SegLink strategy is able to detect text in arbitrary orientation and long lines of text in non-Latin scripts.

1. Introduction

Reading text in natural images is an active and challenging task driven by many real-world applications, such as photo OCR [2], geo-location and image retrieval [9]. In a text reading system, text detection, *i.e.* localizing text regions at the level of word/text line with bounding boxes, is usually the first step of great significance. In a sense, text detection is a specific instance of object detection, for which words/characters/text lines are taken as the targets to detect. Recently, a new trend has merged that the state-of-the-art text detection methods [9, 6, 22, 30] are designed based on the advanced detection/segmentation techniques for general objects [4, 5, 15].

Despite the great success of the previous work, we argue that the detection methods for general objects are not well fit for localizing text regions in several aspects. First, the bounding box of a word/text line has much larger aspect ratio than common objects. An (fast/faster) R-CNN [5, 4, 19]- or YOLO [18]-style detector may suffer from the difficulty of producing such boxes. Second, some non-Latin scripts do not have a blank space between adjacent words, *e.g.* Chinese Hanzi. The methods that directly hit words (*e.g.* [9, 6]) may not be applied to detecting text in such scripts, since there is no visual cue to determine the boundaries of each word. Last, text can be arbitrarily oriented in the wild [25], while most methods [9, 22] are specifically designed for horizontal text.

To overcome the above challenges, we tackle scene text detection in a new perspective. In this paper, we propose to represent scene text with two basic elements, namely *segment* and *link*. As illustrated in Fig. 1, a segment is a bounding box that covers a part of a word (for clarity we use "word" here and later, but segments also can be applied to a "text line" that contains multiple words). A link con-

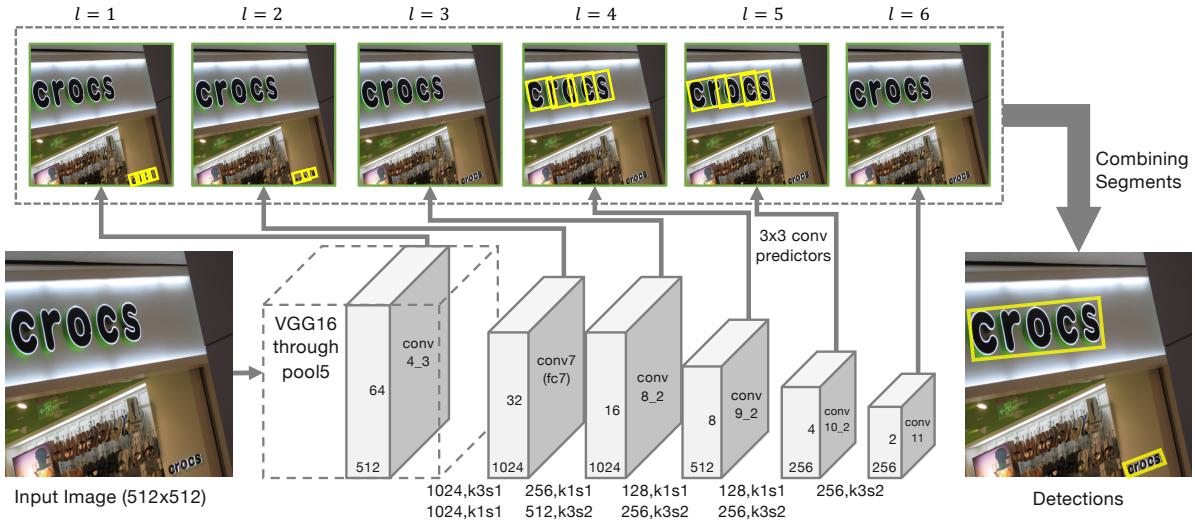

Figure 2. The architecture of the proposed detection network. The network consists of convolutional feature layers (shown as gray blocks) and convolutional predictors (thin gray arrows). Convolutional filters between the feature layers (some have one more convolutional layer between them) are represented in the format of "(#filters),k(kernel size)s(stride)". Segments (yellow boxes) and links (not displayed) are detected by the predictors on multiple feature layers (indexed by l), then combined into whole words by a combining algorithm.

nects a pair of adjacent segments, indicating that they belong to the same word. Under the definitions of segment and link, each word is composed of a number of segments with the links between all the adjacent pairs. In our method, segments and links are densely predicted on images with a convolutional neural network firstly, then the segments from the same word or text line are grouped together according to the link connections.

The key advantage of such a representation with segments and links is that text regions can be described locally with bounding boxes, even for the case of deformation or non-horizontal orientation. In addition, it allows us to detect text of arbitrary length, as the segment combining procedure only relies on the predicted links. Benefiting from the simplicity and flexibility of the proposed representation, our method is applicable to detecting text in long lines, particularly text in non-Latin scripts. Note that segments can be set in arbitrarily orientations in order to cover oriented or curve shaped text.

Specifically, we propose a convolutional neural network (CNN) model to predict both segments and links simultaneously, in a *fully-convolutional* manner. The architecture of the network is based on SSD [14], a recent development for general object detection. Convolutional predictors are added to multiple feature layers to detect segments and links at different scales. In particular, two types of links are detected, namely *within-layer links* and *cross-layer links*. A within-layer link connects a segment to its neighbor segments on one layer. A cross-layer link, on the other hand, connect segments on two different layers. In this way, segments of adjacent locations and scales can be connected. Finally, we combine the segments that are connected by links into whole words as the final outputs.

The main contribution of this paper is the proposed novel detection framework with segments and links, which possesses distinctive advantages over the state-of-the-art methods in three aspects: 1) *Robustness*: It well captures the characteristics of oriented text against complex backgrounds, achieving highly competitive results on standard benchmarks. In particular, it outperforms the previous best results by a large margin (75.0% vs 63.5%) in terms of F-score on the ICDAR 2015 Incidental (Challenge 4) benchmark [12]; 2) *Efficiency*: our method is highly efficient, being able to process more than 20 images with the size of 512x512 per second, as both segments and links are obtained by a single forward pass in a fully-convolutional CNN model, without resizing or rotating input images; 3) *Generality*: In addition to Latin scripts, our method is able to detect text of non-Latin scripts in long lines, which is demonstrated by the evaluation on a multi-lingual dataset.

2. Related Work

Text Detection In the past few years, the community has attracted a large amount of research efforts on text detection in natural images [24, 23, 17, 17, 25, 7, 8, 30, 29, 2, 9, 6, 22, 26]. According to the basic detection element, these methods can be roughly divided into three categories: *character-based*, *word-based* and *line-based*. Character-based methods [17, 23, 24, 10, 7, 8] assume that characters consist of one or several connected components and utilize this property to seek possible candidates of characters or strokes. Such methods often require the post processing of false pos-

itive filtering or component grouping. Word-based methods [9, 6] directly hit the bounding boxes of words, similar to recent popular pipelines of general object detection with convolutional neural networks. Though achieving excellent detection accuracies, these methods may not be applied to non-Latin scripts as mentioned before. Line-based methods [29, 30, 26] segment the regions of text lines in a holistic manner, which also require extra post-processing operations of word partitioning and/or false positive removal.

Compared to previous approaches, most detection procedures of our method focus on predicting segments and links with a single convolutional neural network, resulting in a much simpler and more clean detection pipeline. The proposed SegLink strategy is similar to a recent work [22] that detects text lines by analyzing and grouping a sequence of *fine-scale text proposals* through a combination of a covolutional neural network and recurrent neural layers. However, in our method, segments can be arbitrarily oriented, which are much more appropriate for covering oriented text than the horizontal boxes used in [22].

Object Detection Text detection can be seen as a particular instance of general object detection, which is a fundamental problem in computer vision. State-of-the-art detection system either classifies class-agnostic object proposals with a CNN model [5, 4, 19], or directly estimate object locations and bounding boxes [18, 14].

The architecture of our network is based on a recent object detection model SSD [14]. SSD proposes the idea of detecting objects on multiple feature layers with convolutional predictors. Our model also detects segments and links in a very similar way. Despite the similarity in model, our detection strategy is quite different: SSD directly outputs object bounding boxes, while the segments we detect are only pieces of a word or text line.

3. Segment Linking

Our method detects text with a feed-forward CNN model. Given an input image I of a particular size $w_I \times h_I$, the model outputs a fixed number of segments and links, which are then filtered by their confidence scores and combined into whole words. For both horizontal and oriented text, bounding boxes are represented by 5 parameters, $b = (x_b, y_b, w_b, h_b, \theta_b)$. x_b, y_b are the coordinates of bounding box center, w_b, h_b are the width and height, and θ_b is the angle to the horizontal in radians.

3.1. CNN Model

Fig. 2 illustrates the architecture of the CNN model. The configuration of the feature layers (conv*) is similar to that of SSD [14]. The early layers (conv_1 through pool5) come from a pretrained VGG-16 network [21], whose fully-connected layers are converted into convolutional ones (fc6-7 into conv6-7, respectively). These layers are followed by some extra convolutional layers (conv8-10), which extract even deeper features for detection. The last pooling layer of SSD (pool11) is replaced by a convolutional layer (conv11) in our network.

Segments and links are detected on six feature layers, namely conv4_3, conv7, conv8_2, conv9_2, conv10_2, and conv11. The feature layers produce feature maps of different sizes, extracting high-quality features at different scales. A convolutional predictor with 3×3 filters is added to each of the six layers to jointly detect segments and links. We index the feature layers and the predictors by $l = 1, \ldots, 6$.

Segment Detection Segments are also oriented bounding boxes, denoted by $s = (x_s, y_s, w_s, h_s, \theta_s)$. They are detected by estimating the confidence scores and geometric offsets to a set of *default boxes* [14] that are each associated with a feature map location. For simplicity, we only associate one default box with a map location.

Consider the l-th feature layer whose feature map size is $w_l \times h_l$. A location (x, y) on this map corresponds to a default box centered at (x_a, y_a) on the image, where

$$x_a = \frac{w_I}{w_l}(x + 0.5) \tag{1}$$

$$y_a = \frac{w_I}{w_l}(y + 0.5) \tag{2}$$

The width and height of the default box are both set to a constant a_l.

At the location of (x, y), the predictor predicts 2 numbers as the positive and negative segment scores and 5 numbers as the geometric offsets $(\Delta x_s, \Delta y_s, \Delta w_s, \Delta h_s, \Delta\theta_s)$. The geometric parameters of the segment are calculated by applying the estimated offsets to the default boxes:

$$x_s = a_l \Delta x_s + x_a \tag{3}$$

$$y_s = a_l \Delta y_s + y_a \tag{4}$$

$$w_s = a_l \exp(\Delta w_s) \tag{5}$$

$$h_s = a_l \exp(\Delta h_s) \tag{6}$$

$$\theta_s = \Delta\theta_s \tag{7}$$

The constant a_l controls the scale of the output segments. It is chosen with regard to the receptive field size of the l-th layer. We use the following empirical equation for choosing this size:

$$a_l = \gamma \frac{w_I}{w_l}, \quad \gamma = 1.5. \tag{8}$$

Within-Layer Link Detection The detected segments are parts of the whole words. We then explicitly estimate the connections between them by detecting links. A link

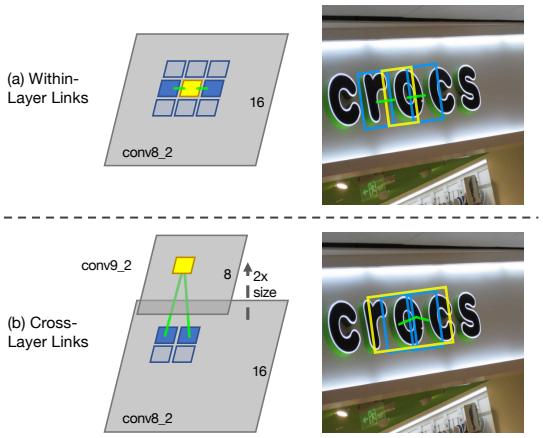

Figure 3. **Within-Layer and Cross-Layer Links.** (a) A location on conv8˙2 (yellow square), its 8-connected neighbors (blue squares). Two within-layer links (green lines) connect a segment (yellow box) and its two neighboring segments (blue boxes) on the same layer. (b) Two cross-layer links connect a segment on conv9_2 (yellow) and two segments on conv8_2.

connects a pair of adjacent segments, indicating that they belong to the same word. Here, adjacent segments are those detected at adjacent map locations. Links are not only necessary for combining segments into whole words, but also helpful for separating two adjacent words, between which the link should be predicted as negative.

Since we detect one segment at a map location, segments can be indexed by their map locations x, y and layer indexes l, *i.e.* $s^{(x,y,l)}$. As illustrated in Fig. 3.a, we define the *within-layer neighbors* of a segment as its 8-connected neighbors on the same feature map:

$$\mathcal{N}^{w}_{s^{(x,y,l)}} = \{s^{(x',y',l)}\}_{x-1\leq x'\leq x+1, y-1\leq y'\leq y+1} \setminus s^{(x,y,l)} \tag{9}$$

As segments are detected locally, a pair of neighboring segments are also adjacent on input image.

Links are also convolutionally detected by the predictor. At a location, the predictor estimates the positive and negative scores of the links between $s^{(x,y,l)}$ and all its 8 neighbors $\mathcal{N}^{w}_{s^{(x,y,l)}}$, producing 16 numbers.

Cross-Layer Link Detection Since segments are detected on multiple feature layers of different scales, the segments of a word can be detected on more than one layer, producing redundancies. Inspired by the linking strategy, we further propose another type of links, called *cross-layer links*, to address the redundancy problem. A cross-layer link connects segments on two consecutive layers, *e.g.* conv4_3 and conv5, conv7 and conv8_2. An important property of these pairs of layers is that the first one always has twice the size as the second one, because there is either a max-pooling layer or a convolutional layer with a stride of 2 between them. Note that this property only holds when all feature maps have even-number sizes, which are ensured by having the width and height of the input image both dividable by 128.

As illustrated in Fig. 3.b, we define the *cross-layer neighbors* of a segment as

$$\mathcal{N}^{c}_{s^{(x,y,l)}} = \{s^{(x',y',l-1)}\}_{2x\leq x'\leq 2x+1, 2y\leq y'\leq 2y+1} \tag{10}$$

The cross-layer neighbors are on the layer below the current one. They correspond to segments detected at a finer scale. Again, cross-layer links are detected convolutionally by the predictor, together with segments and within-layer links. At each map location, the predictor produces 8 numbers, which are the positive and negative scores for the 4 cross-layer links. The cross-layer links are detected on all feature layers except conv4_3, the downmost feature layer. They connect every pairs of consecutive layers. Hence, segments of two different scales can be connected and combined.

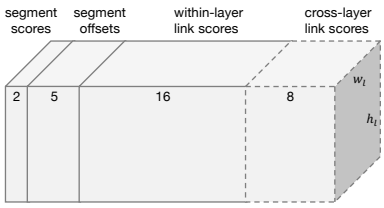

Figure 4. Outputs of a convolutional predictor. Cross-layer link scores are not predicted on conv4_3.

Outputs of Convolutional Predictor Putting things together, Fig. 4 shows the output channels of a convolutional predictor. The predictor is implemented as a convolutional layer that output these channels, with extra softmax layers to normalize the segment and link scores respectively. Therefore, our model detects segments and links in a fully-convolutional manner, yielding fast speed.

3.2. Combining Segments with Links

The last step is to combine the detected segments with the detected links. After feed-forwarding, the CNN model produces a fixed number of segments and links (the number depends on the image size), which are then filtered by their confidence scores. We set different filtering thresholds for segment and link, respectively α and β.

Taking the filtered segments as nodes and the filtered links as edges, we construct a graph over them. Then, a depth-first search (DFS) is performed over the graph to find its connected components. Each component contains a set of segments, denoted by $\mathcal{B}$, that are connected by links. Segments within a set are combined into a whole word by the procedures depicted in Alg. 1.

Algorithm 1 Combining Segments

Input: $\mathcal{B} = \{s^{(i)}\}_{i=1}^{|\mathcal{B}|}$, a set of segments connected by links, where $s^{(i)} = (x_s^{(i)}, y_s^{(i)}, w_s^{(i)}, h_s^{(i)}, \theta_s^{(i)})$
$\theta_b := \frac{1}{|\mathcal{B}|}\sum_{\mathcal{B}} \theta_s^{(i)}$
Find b of the straight line $(\tan\theta_b)x + b$ that minimizes the sum of distance to the centers $(x_s^{(i)}, y_s^{(i)})$ of the segments in $\mathcal{B}$
Find the two endpoints of the straight line as (x_p, y_p) and (x_q, y_q)
$x_b := \frac{1}{2}(x_p + x_q)$
$y_b := \frac{1}{2}(y_p + y_q)$
$w_b := \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} + \frac{1}{2}(w_p + w_q)$
$h_b := \frac{1}{|\mathcal{B}|}\sum_{\mathcal{B}} h_s^{(i)}$
$b := (x_b, y_b, w_b, h_b, \theta_b)$
Output b, the combined bounding box

4. Training

4.1. Groundtruths of Segments and Links

Training the detection network requires groundtruths of segments and links. The groundtruths include the labels of all default boxes, the offsets, and the labels of all within-layer and cross-layer links. They are calculated from the annotated word bounding boxes of a training dataset. We design the following rules to calculate the groundtruths.

Assuming that only one word is on an image, a default box is labeled as positive and matched to that word, if 1) the center of the box is inside the word; 2) the ratio between the box size a_l and the word height h satisfies:

$$\max(\frac{a_l}{h}, \frac{h}{a_l}) \leq 1.5 \tag{11}$$

Otherwise, it is labeled as negative.

When an image contains multiple words, a default box is labeled as negative if none of the word meets the aforementioned criteria. Otherwise, it is labeled as positive and matched to the word that has the closest size, *i.e.* the one with the minimal value at the left hand side of Eq. 11.

Offsets are calculated only on positive default boxes. Following the steps illustrated in Fig. 5 we first calculate the geometric parameters of the *groundtruth segment*. Then, we solve Eq. 3 to Eq. 7 to get the groundtruth offsets for that default box.

Finally, a link (within-layer or cross-layer) is labeled as positive if 1) both of the default boxes connected to it are labeled as positive; 2) the two default boxes are matched to the same word.

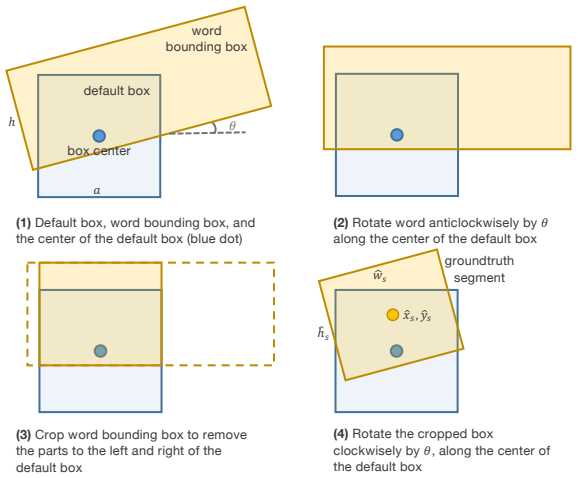

Figure 5. The steps of calculating a groundtruth segment given a default box and a word bounding box.

4.2. Optimization

Objective The CNN model is trained to simultaneously minimize the losses on segment classification, offsets regression, and link classification. Overall, the loss function is a weighted sum of the three losses:

$$L(\mathbf{y}_s, \mathbf{c}_s, \mathbf{y}_l, \mathbf{c}_l, \hat{\mathbf{s}}, \mathbf{s}) = \frac{1}{N_s} L_{\text{conf}}(\mathbf{y}_s, \mathbf{c}_s) + \lambda_1 \frac{1}{N_s} L_{\text{loc}}(\hat{\mathbf{s}}, \mathbf{s}) + \lambda_2 \frac{1}{N_l} L_{\text{conf}}(\mathbf{y}_l, \mathbf{c}_l) \tag{12}$$

Here, $\mathbf{y}_s$ is the labels of all segments. $\mathbf{y}_s^{(i)} = 1$ if the i-th default box is labeled as positive, and 0 otherwise. Likewise, $\mathbf{y}_l$ is the labels of the links. L_{conf} is the softmax loss over the predicted segment and link scores, respectively $\mathbf{c}_s$ and $\mathbf{c}_l$. L_{loc} is the *Smooth L1* regression loss [4] over the predicted segment geometries $\mathbf{s}$ and the groundtruth $\hat{\mathbf{s}}$. The losses on segment classification and regression are normalized by N_s, which is the number of positive default boxes. The loss on link classification is normalized by the number of positive links N_l. The weight constants λ_1 and λ_2 are both set to 1 in practice.

Online Hard Negative Mining For both segments and links, negatives take up most of the training samples. Therefore, a hard negative mining strategy is necessary to balance the positive and negative samples. We follow the online hard negative mining strategy proposed in [20] to keep the ratio between the negatives and positives 3:1 at most. Mining is performed separately for segments and links.

Data Augmentation We adopt an online augmentation pipeline that is similar to that of SSD [14] and YOLO [18]. Training images are randomly cropped to a patch that has

a minimum jaccard overlap of o with groundtruth word bounding boxes, before they are resized to the same size and loaded into a batch. For oriented text, the augmentation is performed over the minimal bounding rectangles of the oriented word bounding boxes. The overlap o is randomly chosen from 0 (no constraint), 0.1, 0.3, 0.5, 0.7, and 0.9 for each sample. The size of the patch is $[0.1, 1]$ of the original image size. Training images are *not* horizontally flipped.

5. Experiments

In this section, we evaluate the proposed algorithm on three benchmark datasets, namely *ICDAR 2015 Incidental Text (Challenge 4)*, *MSRA-TD500* and *ICDAR 2013*, following the standard evaluation protocols in this field.

5.1. Datasets

SynthText in the Wild (SynthText) [6] contains 800,000 *synthetic* training images. They are synthesized by blending natural images with artificial text rendered with random fonts, size, orientation, and color. Moreover, in order to make images more realistic, the authors proposed to seek image regions characterized by a uniform color and texture. Words are aligned to these estimated regions. The dataset provides very detailed annotations for characters, words, and text lines. In our experiments, the dataset is only adopted for pre-training of the proposed CNN model.

ICDAR 2015 Incidental Text (IC15) [12] is the Challenge 4 of the ICDAR 2015 Robust Reading Competition. This challenge features *incidental* scene text images that are captured by Google glasses without taking care of positioning, image quality and viewpoint. Consequently, text in this dataset may exhibit in any orientation and any location with small size or low resolution, making the challenge much more difficult than the previous ICDAR challenges. The dataset contains 1000 training images and 500 testing images. Annotations are provided as word bounding boxes.

MSRA-TD500 (TD500) [25] is the first popular dataset that focuses on oriented text. In addition, the dataset is multi-lingual, including both Chinese and English text. The dataset consists of 300 training images and 200 testing images. Different from IC15, the annotations of TD500 is provided as bounding boxes of text lines. The evaluation protocol for this dataset is slightly different from those for ICDAR challenges, which can be referred to [25].

ICDAR 2013 (IC13) [13] contains only horizontal text (some words are slightly oriented). The dataset has been widely adopted for evaluating text detection performance in previous work. The dataset consists of 229 training images and 233 testing images.

5.2. Implementation Details

Our detection network is pre-trained on the SynthText dataset and finetuned on real data. The detection network is optimized by the standard SGD algorithm with a momentum of 0.9. For both pretraining and finetuning, training images are resized to 384×384 after random cropping. As our model is fully-convolutional, we can train it on a certain size and apply it to images of other sizes in testing. Batch size is set to 32. In pretraining, the learning is set to 10^{-3} for the first 60k iterations, then decayed to 10^{-4} for the rest 30k iterations. For finetuning, the learning rate is fixed to 10^{-4} for 5-10k iterations. The number of finetuning iterations depends on the size of the dataset.

Due to the trade-off between precision and recall, we choose the best values for the thresholds α and β to achieve the best f-measure. In practice, they are chosen separately on different datasets by a grid search with 0.1 step on a hold-out set.

Our method is implemented under TensorFlow [1] (version r0.11), a dataflow-based, GPU-accelerated deep learning framework. The implementation is mostly based on Python, with some custom operators written in C++. All experiments are carried out on a workstation with an Intel Xeon 8-core CPU (2.8 GHz), 4 Titan X Graphics Card, and 64GB RAM. The training process runs on 4 GPUs in parallel, with each GPU processing 8 out of the 32 images in a batch at a time. Training a batch takes about 0.5s. The whole training process takes less than a day. Testing is performed on one GPU. Testing image size varies from dataset to dataset.

5.3. Detecting Oriented English Text

First, we evaluate SegLink on IC15, one of the most challenging datasets at present. The pretrained model is finetuned for 10k iterations on the training dataset of IC15. The original sizes of all testing images are 1280×800. These images are resized to 1280×768, so that their heights and widths are dividable by 128. We set the thresholds on segments and links to 0.9 and 0.7 respectively. Performance scores are calculated by the official central submission server (http://rrc.cvc.uab.es/?ch=4). In order to fit the required submission format, word bounding boxes are converted from oriented rectangles into polygons.

Table 1 lists and compares the results of the proposed method and other state-of-the-art methods, in terms of precision, recall and f-measure. Some results are obtained from the online competition leader-board. From the table, it is clear that our method outperforms the others by a large margin. In terms of f-measure, SegLink outperforms the second best by 10.2%. Considering that several methods have close or even higher precision value compared with SegLink, the improvement mainly comes from the recall. As shown in Fig. 6, our method is able to distinguish text regions from

Figure 6. **Examples on IC15**. Yellow boxes contain zoom-in images. Green regions are correctly detected text regions. Red ones are either false positive or false negative. Gray regions are detected but neglected by the evaluation algorithm. Visualizations are generated by the central submission system.

Table 1. Results on ICDAR 2015 Incidental Text

Method	Precision	Recall	F-measure
HUST_MCLAB	47.5	34.8	40.2
NJU_Text	72.7	35.8	48.0
StradVision-2	**77.5**	36.7	49.8
MCLAB_FCN [30]	70.8	43.0	53.6
CTPN [22]	51.6	74.2	60.9
Megvii-Image++	72.4	57.0	63.8
Yao *et al.* [26]	72.3	58.7	64.8
SegLink	73.1	**76.8**	**75.0**

Table 2. Results on MSRA-TD500

Method	Precision	Recall	F-measure	FPS
Kang *et al.* [11]	71	62	66	-
Yao *et al.* [25]	63	63	60	0.14
Yin *et al.* [27]	81	63	74	0.71
Yin *et al.* [28]	71	61	65	1.25
Zhang *et al.* [30]	83	67	74	0.48
Yao *et al.* [26]	77	75	76	~1.61
SegLink	**86**	**70**	**77**	**8.9**

the very cluttered background. In addition, owing to the explicit link prediction, our method correctly separates words that are very close to each other.

5.4. Detecting Multi-Lingual Text in Long Lines

To show the generality of SegLink, we further demonstrate its ability to detect non-Latin text in long lines. We adopt TD500 here, as it has both arbitrarily-oriented and multi-lingual text. The training set of TD500 only contains 300 images, which are not enough for finetuning our model even with data augmentation. To overcome the insufficiency of training data, we mix the training set of TD500 with the training set of IC15, in the way that every batch has half of its images coming from each dataset. Our pretrained model is finetuned for 8k iterations. The testing images are resized to 768×768. The thresholds α and β are set to 0.9 and 0.5 respectively. Performance scores are calculated by the official development toolkit.

According to the results summarized in Table 2, SegLink achieves the highest scores in terms of precision, recall and f-measure. Moreover, benefiting from its simplicity and well-designed architecture, SegLink processes approximately 9 images per second, being much faster than the competitors. SegLink also enjoys simplicity. The inference process of SegLink is a single forward pass in the detection network, while the previous methods [25, 28, 30] involve sophisticated rule-based grouping or filtering steps.

TD500 contains many long lines of text in both English and Chinese. Fig. 7 shows how SegLink handles such text. As can be seen, the long lines of text are well localized, covered by densely detected segments and links all over the images. In particular, our model suppresses the links between two adjacent text lines, so that the lines are separated rather than combined into one.

5.5. Detecting Horizontal Text

We also test SegLink on IC13 to evaluate its performance on detecting horizontal text. For training, our model is finetuned with 5k iterations with the combination of training sets of IC13 and IC15. As most text in IC13 is in relatively large sizes, the testing images are resized to 512×512 in our implementation. The thresholds α and β are set to 0.6 and 0.3, respectively. Since the submission format of IC13

Figure 7. **Examples on TD500**. The first row displays the detected segments and links. The within-layer and cross-layer links are displayed as red and green lines, respectively. Segments are displayed as rectangles in different colors, which denote different connected components. The second row displays the combined boxes.

is rectangular bounding boxes, we convert the detected oriented boxes into rectangular ones, by finding the minimal bounding rectangle for each of them.

Table 3 compares SegLink with other state-of-the-art methods. The scores are calculated by the central submission system under the "Deteval" evaluation protocol. SegLink achieves very competitive results in terms of f-measure, and is the fastest method which processes more than 20 images per second. Only one approach [22] outperforms SegLink in terms of f-measure. However, [22] is mainly designed for detecting horizontal text. It is also much slower than SegLink. In this sense, SegLink is still a top text detector on horizontal text localization.

Table 3. **Results on IC13**. P, R, F stand for precision, recall and f-measure respectively. *These methods are only evaluated under the "ICDAR 2013" evaluation protocol, the rest under "Deteval". The two protocols mostly yield very close scores.

Method	P	R	F	FPS
Neumann *et al.* [16]*	81.8	72.4	77.1	3
Neumann *et al.* [17]*	82.1	71.3	76.3	3
Busta *et al.* [3]*	84.0	69.3	76.8	6
Zhang *et al.* [29]	88	74	80	<0.1
Zhang *et al.* [30]	88	78	83	<1
Jaderberg *et al.* [9]	88.5	67.8	76.8	<1
Gupta *et al.* [6]	92.0	75.5	83.0	15
Tian *et al.* [22]	**93.0**	**83.0**	**87.7**	7.1
SegLink	87.7	**83.0**	85.3	**20.6**

5.6. Limitations

A major limitation of SegLink is that two thresholds, α and β, need to be set manually. as the recall, precision and f-measure are non-linear functions of the two parameters. In practice, the optimal values of the thresholds are found by a grid search.

Figure 8. **Failure cases on TD500**. Red boxes are false positives. (a)(b) SegLink fails to link the characters with large character spacing. (c) SegLink fails to detect curved text.

Another weakness of our method is that it fails to detect text that has very large character spacing. Fig. 8.a,b show two examples of such cases. The detected links connect adjacent segments, but fail on distant segments.

SegLink may fail to detect text of curved shape, such as the example in Fig. 8.c. However, we believe that this issue can be solved by a modification to the segment combining algorithm, which should be enabled to output deformed bounding boxes besides oriented bounding boxes.

6. Conclusion

We have presented SegLink, a novel text detection algorithm as well as a CNN model for effectively predicting segments and links. The superior performance over other competing methods in the literature on both horizontal and multi-oriented text detection benchmarks demonstrates that our method effectively captures the substructures of scene text, being flexible and general to overcome the large variations of them. As discussed, SegLink works well on both Latin and non-Latin scripts with arbitrary lengths and orientations, and has the potential to detect deformed text such as curve shaped text. Another merit of our method is its

high efficiency, which satisfies the requirement of practical applications. In the future, we are interested in extending SegLink to an end-to-end text recognition system.

Acknowledgment

This work was supported in part by National Natural Science Foundation of China (61222308 and 61573160), a Google Focused Research Award, AWS Cloud Credits for Research, a Microsoft Research Award and a Facebook equipment donation.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 6

[2] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *ICCV*, 2013. 1, 2

[3] M. Busta, L. Neumann, and J. Matas. Fastext: Efficient unconstrained scene text detector. In *ICCV*, 2015. 8

[4] R. B. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 3, 5

[5] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 3

[6] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *CVPR*, 2016. 1, 2, 3, 6, 8

[7] W. Huang, Z. Lin, J. Yang, and J. Wang. Text localization in natural images using stroke feature transform and text covariance descriptors. In *ICCV*, 2013. 2

[8] W. Huang, Y. Qiao, and X. Tang. Robust scene text detection with convolution neural network induced MSER trees. In *ECCV*, 2014. 2

[9] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *IJCV*, 116(1):1–20, 2016. 1, 2, 3, 8

[10] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *ECCV*, 2014. 2

[11] L. Kang, Y. Li, and D. S. Doermann. Orientation robust text line detection in natural images. In *CVPR*, 2014. 7

[12] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. K. Ghosh, A. D. 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 *ICDAR 2015*, 2015. 2, 6

[13] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. Almazán, and L. de las Heras. ICDAR 2013 robust reading competition. In *ICDAR 2013*, 2013. 6

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *ECCV*, pages 21–37, 2016. 2, 3, 5

[15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[16] L. Neumann and J. Matas. Efficient scene text localization and recognition with local character refinement. In *ICDAR*, 2015. 8

[17] L. Neumann and J. Matas. Real-time lexicon-free scene text localization and recognition. *PAMI*, 38(9):1872–1885, 2016. 2, 8

[18] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 1, 3, 5

[19] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 3

[20] A. Shrivastava, A. Gupta, and R. B. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016. 5

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 3

[22] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *ECCV*, 2016. 1, 2, 3, 7, 8

[23] K. Wang and S. J. Belongie. Word spotting in the wild. In *ECCV*, 2010. 2

[24] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *ICPR*, 2012. 2

[25] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *CVPR*, 2012. 1, 2, 6, 7

[26] C. Yao, X. Bai, N. Sang, X. Zhou, S. Zhou, and Z. Cao. Scene text detection via holistic, multi-channel prediction. *CoRR*, abs/1606.09002, 2016. 2, 3, 7

[27] X. Yin, W. Pei, J. Zhang, and H. Hao. Multi-orientation scene text detection with adaptive clustering. *PAMI*, 37(9):1930–1937, 2015. 7

[28] X. Yin, X. Yin, K. Huang, and H. Hao. Robust text detection in natural scene images. *PAMI*, 36(5):970–983, 2014. 7

[29] Z. Zhang, W. Shen, C. Yao, and X. Bai. Symmetry-based text line detection in natural scenes. In *CVPR*, 2015. 2, 3, 8

[30] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. In *CVPR*, 2016. 1, 2, 3, 7, 8