TextBoxes++: A Single-Shot Oriented Scene Text Detector

Minghui Liao, Baoguang Shi, Xiang Bai, *Senior Member, IEEE*

***Abstract*—Scene text detection is an important step of scene text recognition system and also a challenging problem. Different from general object detection, the main challenges of scene text detection lie on arbitrary orientations, small sizes, and significantly variant aspect ratios of text in natural images. In this paper, we present an end-to-end trainable fast scene text detector, named TextBoxes++, which detects arbitrary-oriented scene text with both high accuracy and efficiency in a single network forward pass. No post-processing other than an efficient non-maximum suppression is involved. We have evaluated the proposed TextBoxes++ on four public datasets. In all experiments, TextBoxes++ outperforms competing methods in terms of text localization accuracy and runtime. More specifically, TextBoxes++ achieves an f-measure of 0.817 at 11.6fps for 1024×1024 ICDAR 2015 Incidental text images, and an f-measure of 0.5591 at 19.8fps for 768×768 COCO-Text images. Furthermore, combined with a text recognizer, TextBoxes++ significantly outperforms the state-of-the-art approaches for word spotting and end-to-end text recognition tasks on popular benchmarks.**

***Index Terms*—Scene text detection, multi-oriented text, word spotting, scene text recognition, convolutional neural networks.**

I. Introduction

Scene text is one of the most general visual objects in natural scenes, which frequently appears on road signs, license plates, product packages, *etc*. Reading scene text facilitates a lot of useful applications, such as image-based geolocation. Some applications of scene text detection and recognition are [1]–[4]. Despite the similarity to traditional OCR, scene text reading is much more challenging due to large variations in both foreground text and background objects, arbitrary orientations, aspect ratios, as well as uncontrollable lighting conditions, *etc*., as summarized in [5]. Owing to these inevitable challenges and complexities, traditional text detection methods tend to involve multiple processing steps, *e.g.* character/word candidate generation [6]–[8], candidate filtering [8], and grouping [9]. They often end up struggling to get each module work properly, requiring much effort in tuning parameters and designing heuristic rules, also slowing down detection speed.

Inspired by the recent developments in object detection [10], [11], we propose to detect text by *directly* predicting word bounding boxes with arbitrary orientation via a single neural network that is end-to-end trainable. We call it TextBoxes++. Concretely, we replace the rectangular box representation in conventional object detector by a quadrilateral or oriented rectangle representation. Furthermore, to achieve better receptive field that covers text regions which are usually long, we design some "long" convolutional kernels to predict the bounding boxes. TextBoxes++ directly outputs word bounding boxes at multiple layers by jointly predicting text presence and coordinate offsets to *anchor boxes* [11], also known as *default boxes* [10]. The final outputs are the non-maximum suppression outputs on all boxes. A single forward pass in the network densely detects multi-scale text boxes all over the image. As a result, the detector has a great advantage in speed. We will show by experiments that TextBoxes++ achieves both high accuracy and high speed with a single forward pass on single-scale inputs, and even higher accuracy when performing multiple passes on multi-scale inputs. Some text detection examples on several challenging images are depicted in Fig. 1.

We further combine TextBoxes++ with CRNN [12], an open-source text recognition module. The recognizer not only produces extra recognition outputs but also regularizes text detection with its semantic-level awareness, thus further boosts the accuracy of word spotting considerably. The combination of TextBoxes++ and CRNN yields state-of-the-art performance on both word spotting and end-to-end text recognition tasks, which appears to be a simple yet effective solution to robust text reading in the wild.

A preliminary version of this study called TextBoxes was exposed in [13]. The current paper extends TextBoxes with four main improvements: 1) We extend TextBoxes, a horizontal text detector, to a detector that can handle arbitrary-oriented text; 2) We revisit and improve the network structure and the training process, which leads to a further performance boost; 3) More comparative experiments have been conducted to further demonstrate the efficiency of TextBoxes++ in detecting arbitrary-oriented text in natural images; 4) We refine the combination of detection and recognition by proposing a novel score which elegantly utilizes both the information of detection and recognition.

The main contributions of this paper are three folds: 1) The proposed arbitrary-oriented text detector TextBoxes++ is accurate, fast, and end-to-end trainable. As compared to closely related methods, TextBoxes++ has a much simpler yet effective pipeline. 2) This paper also offers some comparative studies on some important design choices and other issues, including bounding box representations, model configurations, and the effect of different evaluation methods. The conclusions of these studies may generalize to other text reading algorithms and give insights on some important issues. 3) We also introduce the idea of using recognition results to further refine the detection results thanks to the semantic-level awareness of

Minghui Liao, Baoguang Shi, Xiang Bai are with the School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan, 430074, China. Email: {mhliao, xbai}@hust.edu.cn shibaoguang@gmail.com.

Fig. 1: Detection results on some challenging images.

recognized text. To the best of our knowledge, this intuitive yet effective combination has not been exploited before.

The rest of the paper is organized as follows. Section II briefly reviews some related works on object detection and scene text detection. The proposed method is detailed in Section III. We present in Section IV some experimental results. A detailed comparison with some closely related methods is given in Section V. Finally, conclusions are drawn in Section VI.

II. Related Works

A. Object detection

Object detection is a fundamental problem in computer vision, which aims to detect general objects in images. Recently, there are two mainstream CNN-based methods on this topic: R-CNN based methods [11], [14], [15] and YOLO-based methods [10], [16].

a) R-CNN based object detection: R-CNN [14] views a detection problem as a classification problem leveraging the development of classification using convolutional neural networks(CNN). It first generates proposals by selective search [17] and then feeds the proposals into a CNN to extract deep features based on which an SVM [18] is applied for classification. Fast R-CNN [15] improves R-CNN by extracting deep features of the proposals from the feature maps via RoI pooling [15] instead of cropping from the origin image. This significantly simplifies the pipeline of R-CNN. Furthermore, a regression branch is also used in fast R-CNN to get more accurate bounding boxes. Faster R-CNN [11] further improves the speed of fast R-CNN [15] by introducing an end-to-end trainable region proposal network based on anchor boxes to generate proposals instead of using selective search. The generated proposals are then fed into a Fast R-CNN network for detection.

b) YOLO-based object detection: The original YOLO [16] directly regresses the bounding boxes on the feature maps of the whole image. A convolutional layer is used to predict the bounding boxes on different areas instead of the RPN and RoI pooling used in [11]. This results in a significantly reduced runtime. Another YOLO-based method is SSD [10] which predicts object bounding boxes using default boxes of different aspect ratios on different stages. The concept of default boxes [10] is similar to anchor boxes [11], which are fixed as the reference systems of the corresponding ground truths. The translation and scale invariance for regression is achieved by using default boxes of different aspect ratios and scales at each location, which eases the regression problem. SSD further improves the performance of original YOLO while maintaining its runtime.

B. Text detection

A scene text reading system is usually composed of two main components: text detection and text recognition. The former component localizes text in images mostly in the form of word bounding boxes. The latter one transcripts cropped word images into machine-interpretable character sequences. In this paper, we cover both aspects, but more attention is paid to detection. In general, most text detectors can be roughly classified into several categories following two classification strategies based on primitive detection targets and shape of target bounding boxes, respectively.

a) Classification strategy based on primitive detection targets: Most text detection methods can be roughly categorized into three categories:
1) Character-based: Individual characters or parts of the text are first detected and then grouped into words [7]. A representative example is the method proposed in [7] which locates characters by classifying Extremal Regions and then groups the detected characters by an exhaustive search method. Other popular examples of this type are the works in [19]–[23];
2) Word-based: Words are directly extracted in the similar manner as general object detection [8], [24]–[26]. In the representative work [8], the authors propose an R-CNN-based [14] framework, where word candidates are first generated with class-agnostic proposal generators followed by a random forest classifier. Then a convolutional neural network for bounding box regression is adopted to refine the bounding boxes. YOLO network [16] is used in [27] which also relies on a classifier and a regression step to remove some false positives;
3) Text-line-based: Text lines are detected and then broken into words. The works in [21], [28]–[30] are such examples. In [28], the authors propose to detect text lines making use of the symmetric characteristics of text. This idea is further exploited in [29] by using a fully convolutional network to localize text lines.

b) Classification strategy based on shape of target bounding boxes: Following this classification strategy, the text detection methods can be categorized into two categories:
1) Horizontal or nearly horizontal: These methods focus on detecting horizontal or nearly horizontal text in images. An algorithm based on AdaBoost is proposed in [31]. Then, Yi et al. [32] propose a 3-stage framework which consists of boundary clustering, stroke segmentation, and string fragment classification. Some examples which are inspired by object detection methods are [8], [27]. They all use horizontal rectangle bounding boxes as predict targets, which is very similar to general object detection methods. Another popular method of this type is the work in [33] which detects nearly horizontal text parts and then links them together to form word candidates. Besides, Cao et al. [34] try to use deblurring techniques for more robust detection results.
2) Multi-oriented: As compared to horizontal or nearly horizontal detection, multi-oriented text detection is more robust because scene text can be in arbitrary orientations in images. There exist several works which attempt to detect multi-oriented text in images. In [19], the authors propose two sets of rotation-invariant features for detecting multi-oriented text. The first set is component level features such as estimated center, scale, direction before feature computation. The second one is chain level features such as size variation, color self-similarity, and structure self-similarity. Kang et al. [35] build a graph of MSERs [36] followed by a higher-order correlation clustering to generate multi-oriented candidates. A unified framework for multi-oriented text detection and recognition is proposed in [37]. They use the same features for text detection and recognition. Finally, a texton-based texture classifier is used to discriminate text and no-text candidates. In [30], [38], multi-oriented text bounding boxes are generated from text saliency maps given by a dedicated segmentation network. Recently, Shi et al. [39] propose to detect text with segments and links. More precisely, they first detect a number of text parts named segments and meanwhile predict the linking relationships among neighboring segments. Then related segments are linked together to form text bounding boxes. A U-shape fully convolutional network is used in [40] for detecting multi-oriented text. In this work, the authors also introduce a PVANet [41] for efficiency. Quadrilateral sliding windows, a Monte-Carlo method, and a smooth Ln loss are proposed in [42] to detect oriented text, which is effective while complicated.

C. TextBoxes++ Versus some related works

TextBoxes++ is a *word-based* and *multi-oriented* text detector. In contrast to [8], which consists of three detection steps and each step includes more than one algorithm, TextBoxes++ has a much simpler pipeline. Only an end-to-end training of one network is required. As described in Section II-B, Tian et al. [33] and Shi et al. [39] propose to detect text parts and then link them together. Both of them achieve impressive results for long words. However, the proposed method in [33] has limited adaptability for oriented text due to the single orientation of the Bidirectional Long Short-Term Memory (BLSTM) [43], and the work in [39] has two super parameters for linking the segments, which are determined by grid search and difficult to adjust. The method in [40] is considered as the current state-of-the-art approach. It relies on a U-shape network to simultaneously generate a score map for text segmentation and the bounding boxes. Yet, an accurate text region segmentation is challenging in itself. Besides, The extra pyramid-like deconvolutional layers involved in text region segmentation require additional computation. Whereas, TextBoxes++ directly classifies and regresses the default boxes on the convolutional feature maps, where richer information is reserved as compared to the segmentation score map. TextBoxes++ is thus much simpler, avoiding the time consuming on pyramid-like deconvolution.

One of the most related work to TextBoxes++ is SSD [10], a recent development in object detection. Indeed, TextBoxes++ is inspired by SSD [10]. The original SSD aims to detect general objects in images but fails on words having extreme aspect ratios. TextBoxes++ relies on specifically designed textbox layers to efficiently solve this problem. This results in a significant performance improvement. Furthermore, SSD can only generate bounding boxes in terms of horizontal rectangles, while TextBoxes++ can generate arbitrarily oriented bounding boxes in terms of oriented rectangles or general quadrilaterals to deal with oriented text.

Another closely related work to TextBoxes++ is the method in [42], which proposes quadrilateral sliding windows and a Monte-Carlo method for detecting oriented text. In TextBoxes++, we use horizontal rectangles as default boxes, and hence have much less default boxes in every region. Benefiting from the horizontal rectangle default boxes, TextBoxes++ also enjoys a much simpler strategy for matching default boxes. Moreover, TextBoxes++ simultaneously regresses the maximum horizontal rectangles of the bounding boxes and the quadrilateral bounding boxes, which makes the training more stable. Furthermore, we propose a novel score by combining the detection and recognition scores to further refine the detection results.

The ultimate goal of text detection is to spot words or recognize text in images. On the other hand, the semantic-level awareness of spotted words or recognized words can also help to further regularize text detection results. Following this idea, we propose to combine a text recognizer with TextBoxes++ for word spotting and end-to-end recognition, and use the confidence scores of recognized words to regularize the detection outputs of TextBoxes++. For that, we adopt a state-of-the-art text recognizer called CRNN [12], which directly outputs character sequences given input images and is also end-to-end trainable. Other text recognizers such as [8] are also applicable. Such simple pipeline for word spotting and end-to-end recognition is very different from the classical pipelines.

III. Detecting oriented text with TextBoxes++

A. Overview

TextBoxes++ relies on an end-to-end trainable fully convolutional neural network to detect arbitrary-oriented text. The basic idea is inspired by an object detection algorithm SSD

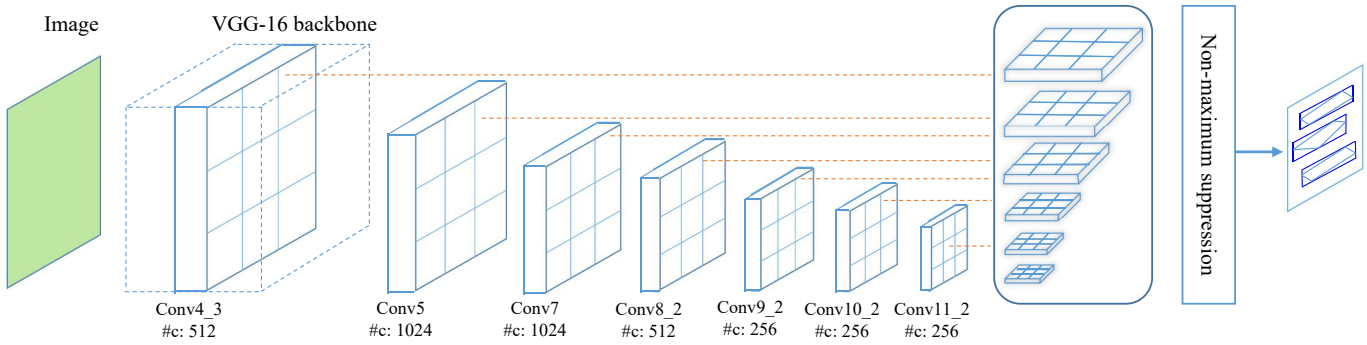


Fig. 2: The architecture of TextBoxes++, a 29-layer fully convolutional network including 13 layers from VGG-16 followed by 10 extra convolutional layers, and 6 Text-box layers connected to 6 intermediate convolutional layers. Each location of a text-box layer predicts an n dimensional vector for each default box consisting of the text presence scores (2 dimensions), horizontal bounding rectangles offsets (4 dimensions), and rotated rectangle bounding box offsets (5 dimensions) or quadrilateral bounding box offsets (8 dimensions). A non-maximum suppression is applied during test phase to merge the results of all 6 text-box layers. Note that "#c" stands for the number of channels.

proposed in [10] using default boxes. We propose some special designs for adapting SSD network to efficiently detect oriented text in natural images. More specifically, we propose to represent arbitrary-oriented text by quadrilaterals [40] or oriented rectangles [44]. Then we adapt the network to predict the regressions from default boxes specially designed for oriented text to oriented text represented by quadrilaterals or oriented rectangles. To better cover the text which could be dense in some area, we propose to densify default boxes with vertical offsets. Furthermore, we adapt the convolution kernels to better handle text lines which are usually long objects as compared to general object detection. These network adaptions are detailed in Section III-B. Some specific training adaptions for arbitrary-oriented text detection are presented in Section III-C, including on-line hard negative mining and data augmentation by a novel random cropping strategy specifically designed for text which is usually small. TextBoxes++ detects oriented text at 6 different scales in 6 stages. During the test phase, the multi-stage detection results are merged together by an efficient cascaded non-maximum suppression based on IOU threshold of quadrilaterals or oriented rectangles (see Section III-D). Finally, we also propose an intuitive yet effective idea of using text recognition to further refine detection results thanks to the semantic-level awareness of recognized text. This is discussed in Section III-E.

B. Proposed network

1) Network architecture: The architecture of TextBoxes++ is depicted in Fig. 2. It inherits the popular VGG-16 architecture [45], keeping the layers from `conv1_1` through `conv5_3`, and converting the last two fully-connected layers of VGG-16 into convolutional layers (`conv6` and `conv7`) by parameters down-sampling [10]. Another eight convolutional layers divided into four stages (`conv8` to `conv11`) with different resolutions by max-pooling are appended after `conv7`. Multiple output layers, which we call *text-box layers*, are inserted after the last and some intermediate convolutional layers. They are also convolutional layers to predict outputs for aggregation and then undergo an efficient non-maximum suppression (NMS) process. Putting all above together, TextBoxes++ is a fully-convolutional structure consisting of only convolutional and pooling layers. As a result, TextBoxes++ can adapt to images of arbitrary size in both training and testing phases. Compared with a preliminary study in [13] of this paper, TextBoxes++ replaces the last global average pooling layer with a convolutional layer, which is furthermore beneficial for multi-scale training and testing.

2) Default boxes with vertical offsets: Text-box layers are the key component of TextBoxes++. A text-box layer simultaneously predicts text presence and bounding boxes, conditioned on its input feature maps. The output bounding boxes of TextBoxes++ include oriented bounding boxes $\{\mathbf{q}\}$ or $\{\mathbf{r}\}$, and minimum horizontal bounding rectangles $\{\mathbf{b}\}$ containing the corresponding oriented bounding boxes. This is achieved by predicting the regression of offsets from a number of pre-designed horizontal default boxes at each location (see Fig. 3 for an example). More precisely, let $\mathbf{b}_0 = (x_0, y_0, w_0, h_0)$ denote a horizontal default box, which can also be written as $\mathbf{q}_0 = (x^q_{01}, y^q_{01}, x^q_{02}, y^q_{02}, x^q_{03}, y^q_{03}, x^q_{04}, y^q_{04})$ or $\mathbf{r}_0 = (x^r_{01}, y^r_{01}, x^r_{02}, y^r_{02}, h^r_0)$, where (x_0, y_0) means the center point of a default box and w_0 and h_0 are the width and height of a default box respectively. The relationships among $\mathbf{q}_0$, $\mathbf{r}_0$ and $\mathbf{b}_0$ are as following:

$$\begin{aligned}
x^q_{01} &= x_0 - w_0/2, y^q_{01} = y_0 - h_0/2, \\
x^q_{02} &= x_0 + w_0/2, y^q_{02} = y_0 - h_0/2, \\
x^q_{03} &= x_0 + w_0/2, y^q_{03} = y_0 + h_0/2, \\
x^q_{04} &= x_0 - w_0/2, y^q_{04} = y_0 + h_0/2, \\
x^r_{01} &= x_0 - w_0/2, y^r_{01} = y_0 - h_0/2, \\
x^r_{02} &= x_0 + w_0/2, y^r_{02} = y_0 - h_0/2, \\
h^r_0 &= h_0.
\end{aligned} \tag{1}$$

At each map location, it outputs the classification score and offsets to each associated default box denoted as q_0 or r_0 in a convolutional manner. For the quadrilateral representation of oriented text, the text-box layers predict the values of

$(\Delta x, \Delta y, \Delta w, \Delta h, \Delta x_1, \Delta y_1, \Delta x_2, \Delta y_2, \Delta x_3, \Delta y_3, \Delta x_4, c)$, indicating that a horizontal rectangle $\mathbf{b} = (x, y, w, h)$ and a quadrilateral $\mathbf{q} = (x_1^q, y_1^q, x_2^q, y_2^q, x_3^q, y_3^q, x_4^q, y_4^q)$ given in the following are detected with confidence c:

$$\begin{aligned} x &= x_0 + w_0 \Delta x, \\ y &= y_0 + h_0 \Delta y, \\ w &= w_0 \exp(\Delta w), \\ h &= h_0 \exp(\Delta h), \\ x_n^q &= x_{0n}^q + w_0 \Delta x_n^q, n = 1, 2, 3, 4 \\ y_n^q &= y_{0n}^q + h_0 \Delta y_n^q, n = 1, 2, 3, 4. \end{aligned} \tag{2}$$

When the representation by rotated rectangles is used, the text-box layers predict the value of $(\Delta x, \Delta y, \Delta w, \Delta h, \Delta x_1, \Delta y_1, \Delta x_2, \Delta y_2, \Delta h^r, c)$, and the rotated rectangle $\mathbf{r} = (x_1^r, y_1^r, x_2^r, y_2^r, h^r)$ is calculated as following:

$$\begin{aligned} x_n^r &= x_{0n}^r + w_0 \Delta x_n^r, n = 1, 2 \\ y_n^r &= y_{0n}^r + h_0 \Delta y_n^r, n = 1, 2 \\ h^r &= h_0^r \exp(\Delta h_r). \end{aligned} \tag{3}$$

In the training phase, ground-truth word boxes are matched to default boxes according to box overlap following the matching scheme in [10]. As shown in Fig. 3, the minimum bounding horizontal rectangle of a rotated rectangle or a quadrilateral is used to match the default boxes for efficiency. Note that there are a number of default boxes with different aspect ratios at each location. In this way, we effectively divide words by their aspect ratios, allowing TextBoxes++ to learn specific regression and classification weights that handle words of similar aspect ratio. Therefore, the design of default boxes is highly task-specific.

Different from general objects, words tend to have large aspect ratios. Therefore, the preliminary study TextBoxes in [13] include "long" default boxes that have large aspect ratios. Specifically, for the horizontal text dataset, we defined 6 aspect ratios for default boxes, including 1,2,3,5,7, and 10. However, TextBoxes++ aims to detect arbitrary-oriented text. Consequently, we set the aspect ratios of default boxes to $1, 2, 3, 5, 1/2, 1/3, 1/5$. Furthermore, text is usually dense on a certain area, so each default box is set with a vertical offset to better cover all text, which makes the default boxes dense on the vertical orientation. An example is shown in Fig. 4. In Fig. 4(a), the normal default box (black dashed) in the middle can not handle the two words close to it at the same time. In this case, one word would be missed for detection if no vertical offset is applied. In Fig. 4(b), the normal default boxes does not cover the bottom word at all, which also demonstrates the necessity of using default boxes with vertical offsets.

3) convolutional layers: In the preliminary study of this paper for horizontal text detection in [13], we have adopted irregular 1×5 convolutional filters instead of the standard 3×3 ones in the text-box layers. This is because that words or text lines in natural images are usually long objects. However, these long convolutional filters are not appropriate for oriented text. Instead, we use 3×5 convolutional filters for oriented text. These inception-style [46] filters use rectangular receptive fields, which better fit words with larger aspect ratios. The noise signals that a square-shaped receptive field would bring in are also avoided thanks to these inceptions.

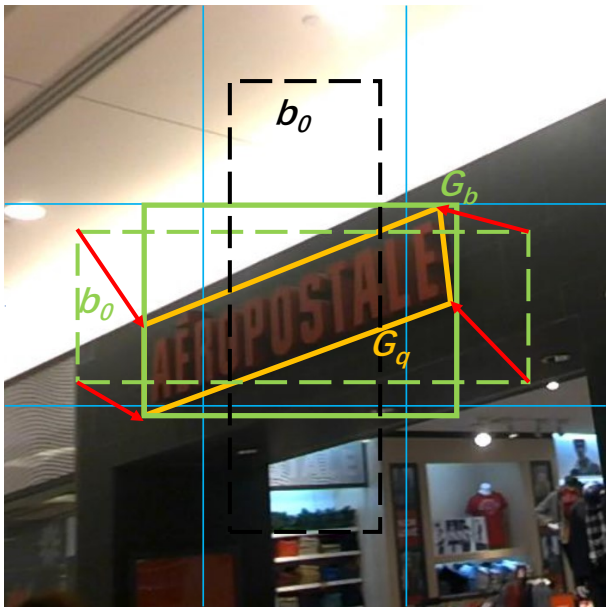
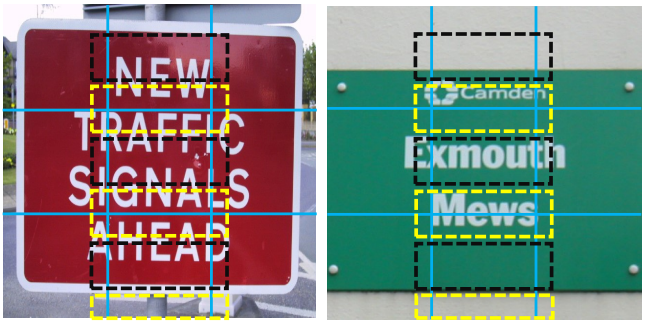


Fig. 3: Illustration of the regression (red arrows) from a matched default box (green dashed) to a ground truth target quadrilateral (yellow) on a 3×3 grid. Note that the black dashed default box is not matched to the ground truth. The regression from the matched default box to the minimum horizontal rectangle (green solid) containing the ground truth quadrilateral is not shown for a better visualization.



(a) (b)

Fig. 4: Vertical offsets of default boxes on a 3×3 grid. Black (*resp.* yellow) dashed bounding boxes are normal default boxes (*resp.* default boxes with vertical offsets). Note that only the default boxes of appropriate aspect ratio are shown for better visualization.

C. Adapted training for arbitrary-oriented text detection

1) Ground Truth representation: For the two target box representations described in Section III-B, we adapt the ground truth representation of oriented text as following:

a) Quadrilateral: : For each oriented text ground truth T, let $\mathbf{G}_b = (\tilde{x}_0^b, \tilde{y}_0^b, \tilde{w}_0^b, \tilde{h}_0^b)$ be its horizontal rectangle ground

truth (*i.e.,* the minimum horizontal rectangle enclosing T), where $(\tilde{x}_0^b, \tilde{y}_0^b)$ is the center of $\mathbf{G}_b$, $\tilde{w}_0^b$ and $\tilde{h}_0^b$ are the width and the height of $\mathbf{G}_b$ respectively. This rectangle ground truth can also be denoted as $\mathbf{G}_b = (b_1, b_2, b_3, b_4)$ following Eq. (1), where (b_1, b_2, b_3, b_4) are the four vertices in clockwise order of $\mathbf{G}_b$ with b_1 the top-left one. We use the four vertices of the oriented text ground truth T to represent T in terms of a general quadrilateral denoted by $\mathbf{G}_q = (q_1, q_2, q_3, q_4) = (\tilde{x}_1^q, \tilde{y}_1^q, \tilde{x}_2^q, \tilde{y}_2^q, \tilde{x}_3^q, \tilde{y}_3^q, \tilde{x}_4^q, \tilde{y}_4^q)$. The four vertices (q_1, q_2, q_3, q_4) are also organized in clockwise order such that the sum of Euclidean distances between four point pairs $(b_i, q_i), i = 1, 2, 3, 4$ is minimum. More precisely, let (q_1', q_2', q_3', q_4') in clockwise order represent the same quadrilateral $\mathbf{G}_q$ with q_1' being the top point (top-left point in case of $\mathbf{G}_q$ being a rectangle). Then the relationship between q and q' is given by:

$$\begin{aligned} d_{i,\Delta} &= d_E(b_i, q'_{(i+\Delta-1)\%4+1}), \Delta = 0, 1, 2, 3 \\ \Delta_m &= \arg\min_{\Delta} (d_{1,\Delta} + d_{2,\Delta} + d_{3,\Delta} + d_{4,\Delta}), \\ q_i &= q'_{(i+\Delta_m-1)\%4+1}, \end{aligned} \tag{4}$$

where $d_E(b, q')$ is the Euclidean distance between two points, and Δ_m is the shift of points that gives the minimal sum of distance of four pair of corresponding points between $\mathbf{G}_b$ and $\mathbf{G}_q$.

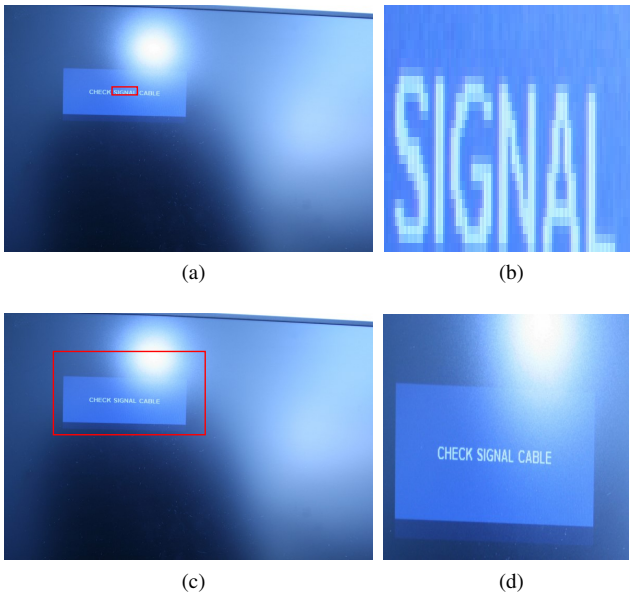
b) Rotated rectangle: : There are several different representations for rotated rectangles. A popular one is given by a horizontal rectangle and a rotated angle, which could be written as (x, y, w, h, θ). However, due to the bias of the dataset, there is usually an uneven distribution on θ, which may make the model dataset-dependent. To ease this problem, we propose to use another representation proposed in [44] for a ground truth rotated rectangle $\mathbf{G}_r$: $\mathbf{G}_r = (\tilde{x}_1^r, \tilde{y}_1^r, \tilde{x}_2^r, \tilde{y}_2^r, \tilde{h}^r)$, where $(\tilde{x}_1^r, \tilde{y}_1^r)$ and $(\tilde{x}_2^r, \tilde{y}_2^r)$ are the first and second vertex of the ground truth (*i.e.,* the first and second vertex of $\mathbf{G}_q$), $\tilde{h}^r$ is the height of the rotated rectangle.

2) Loss function: We adopt a loss function similar to the one used in [10]. More specifically, let x be the match indication matrix. For the i-th default box and the j-th ground truth, $x_{ij} = 1$ means a match following the box overlap between them, otherwise $x_{ij} = 0$. Let c be the confidence, l be the predicted location, and g be the ground-truth location. The loss function is defined as:

$$L(x, c, l, g) = \frac{1}{N}(L_{\text{conf}}(x, c) + \alpha L_{\text{loc}}(x, l, g)), \tag{5}$$

where N is the number of default boxes that match ground-truth boxes, and α is set to 0.2 for quick convergence. We adopt the smooth L1 loss [15] for L_{loc} and a 2-class soft-max loss for L_{conf}.

3) On-line hard negative mining: Some textures and signs are very similar to text, which are hard for the network to distinguish. We follow the hard negative mining strategy used in [10] to suppress them. More precisely, the training on the corresponding dataset is divided into two stages. The ratio between the negatives and positives for the first stage is set to 3:1, and then changed to 6:1 for the second stage.



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

Fig. 5: Data augmentation by random cropping based on Jaccard overlap (a-b) and object coverage constraints (c-d). Images in (b) and (d) are the corresponding resized crops.

4) Data Augmentation: Similar to many CNN-based vision problems, data augmentation is a classical and necessary way to increase the limited size of a training set. For example, a random crop augmentation based on the minimum Jaccard overlap between crops and ground truths is applied in [10]. However, this strategy is not appropriate for text which is usually small. This is because that the Jaccard overlap constraint is difficult to satisfy for small objects. As depicted by an example in Fig. 5(a-b), for a small object, even if the Jaccard overlap is satisfied, the object in the resized image after data augmentation is extremely large that almost covers the whole image. This is not the usual case for text in natural images. To ease this problem, we propose to add a new overlap constraint called object coverage in addition to the Jaccard overlap. For a cropped bounding box $\mathbf{B}$ and a ground truth bounding box $\mathbf{G}$, The Jaccard overlap $\mathbf{J}$ and object coverage $\mathbf{C}$ are defined as follows:

$$\begin{aligned} \mathbf{J} &= |\mathbf{B} \cap \mathbf{G}| / |\mathbf{B} \cup \mathbf{G}|, \\ \mathbf{C} &= |\mathbf{B} \cap \mathbf{G}| / |\mathbf{G}|, \end{aligned} \tag{6}$$

where $|\cdot|$ denotes the cardinality (*i.e.* area). The random crop strategy based on object coverage $\mathbf{C}$ is more appropriate for small objects such as most text in natural images. An example is given in Fig. 5(c-d). In this paper, we use both random crop strategies with minimum overlap or coverage thresholds randomly set to 0, 0.1, 0.3, 0.5, 0.7 and 0.9. Note that a threshold set to 0 implies that neither minimum Jaccard overlap nor object coverage constraint is used. Each cropped region is then resized to a fixed size image that feeds into the network.

5) Multi-scale training: For the sake of training speed, the randomly cropped regions are resized to images of a relatively small size. However, the input images of the pro-

posed TextBoxes++ can have arbitrary size thanks to its fully convolutional architecture. To better handle multi-scale text, we also use larger scale input size for the last several thousand iterations in training phase. The training details are discussed in Section IV-B.

D. Testing with efficient cascaded non-maximum suppression

Similar to classical methods for object detection, we apply a Non-Maximum Suppression (NMS) during the test period to extract predicted boxes of arbitrary-oriented text. Firstly, we resize the six multi-scale prediction results to the original size of an input image, and fusion these resized results into one dense confidence map. Then the NMS operation is applied on this merged confidence map. Since the NMS operation on quadrilaterals or rotated rectangles is more time-consuming than that on horizontal rectangles, we divide this NMS operation into two steps to accelerate the speed. First, we apply the NMS with a relatively high IOU threshold (*e.g.* 0.5) on the minimum horizontal rectangles containing the predicted quadrilaterals or rotated rectangles. This operation on horizontal rectangles is much less time-consuming and removes many candidate boxes. Then the time-consuming NMS on quadrilaterals or rotated rectangles is applied to a few remaining candidate boxes with a lower IOU threshold (*e.g.* 0.2). The remaining boxes after this second NMS operation are considered as the final detected text boxes. This cascaded non-maximum suppression is much faster than directly applying NMS on the quadrilaterals or rotated rectangles.

E. Word spotting, end-to-end recognition, and detection refining

Intuitively, a text recognizer would help to eliminate some false-positive detection results that are unlikely to be meaningful words, *e.g.* repetitive patterns. Particularly, when a lexicon is present, a text recognizer could effectively remove the detected bounding boxes that do not match any of the given words. Following this intuitive idea, we propose to improve the detection results of TextBoxes++ with word spotting and end-to-end recognition.

1) Word spotting and end-to-end recognition: Word spotting is to localize specific words that are given in a lexicon. End-to-end recognition concerns both detection and recognition. Both tasks can be achieved by simply connecting TextBoxes++ with a text recognizer. We adopt the CRNN model [12] as our text recognizer. CRNN uses CTC [47] as its output layer, which estimates a sequence probability conditioned on the input image I denoted as $p(\mathbf{w}|I)$, where $\mathbf{w}$ represents a character sequence output. If no lexicon is given, $\mathbf{w}$ is considered as the recognized word, and the probability $p(\mathbf{w}|I)$ measures the compatibility of an image to that particular word $\mathbf{w}$. CRNN also supports the use of lexicon. For a given lexicon $\mathcal{W}$, CRNN outputs the probability that measures how the input image I matches each word $w \in \mathcal{W}$. We define the recognition score s_r in the following:

$$s_r = \begin{cases} p(\mathbf{w}|I), & \text{Without lexicon} \\ \max\limits_{\mathbf{w}\in\mathcal{W}} p(\mathbf{w}|I), & \text{With lexicon } \mathcal{W} \end{cases} \tag{7}$$

Note that the use of lexicon is not a necessary in the proposed method. We only use lexicons for fair comparisons with other methods.

2) Refining detection with recognition: We propose to refine detection with recognition by integrating the recognition score s_r to the original detection s_d score. In practice, the value of recognition score is generally not comparable to the value of detection score. For example, the threshold of the detection score s_d is usually set to 0.6, and the threshold of recognition score s_r is usually set to 0.005. A trivial combination of these two scores would lead to a severe bias of detection score. In this paper, we propose to define the novel score S as following:

$$S = \frac{2 \times e^{(s_d + s_r)}}{e^{s_d} + e^{s_r}}. \tag{8}$$

There are two motivations in Eq. (8). First, we use the exponential function to make the two score values comparable. Then, a harmonic mean is adopted to get the final combined score. This combined score S is more convenient than applying a grid search on two scores, respectively.

IV. Experiments

Inherited from object detection, all existing scene text detection benchmarks rely on an IOU threshold to evaluate the performance of text detectors. However, text detection is quite different from object detection because the ultimate purpose of detecting text is text recognition. A text recognizer may yield totally different results with the same IOU. For example, the three detection results in Fig. 6(a-c) have the same IOU. However, the detection results in Fig. 6(a) and Fig. 6(b) fail to correctly recognize the underlying text due to the lack of text parts. The detection results in Fig. 6(c) tends to give an accurate recognition result thanks to the full text coverage. Thus, in addition to classical text detection benchmarks, we also conduct word spotting and end-to-end recognition experiments using the text detection results to further demonstrate the performance of TextBoxes++.

(a) (b) (c)

Fig. 6: An example of recognition results (yellow text) with different text detection results having the same IOU. The red (*resp.* green) bounding box is the ground truth (*resp.* detection result).

A. Datasets & evaluation protocol

The proposed TextBoxes++ detects arbitrary-oriented text. We have tested its performance on two datasets including oriented text: ICDAR 2015 Incidental Text (IC15) dataset [48] and COCO-Text dataset [49]. To further demonstrate the

versatility of TextBoxes++, we have conducted experiments on two popular horizontal text datasets: ICDAR 2013 (IC13) dataset [50] and Street View Text (SVT) dataset [51]. Besides these benchmark datasets, the SynthText dataset [27] is also used to pre-train our model. A short description of all these concerned datasets is given in the following (see the corresponding references for more details).

SynthText: The SynthText dataset [27] contains 800k synthesized text images, created via blending rendered words with natural images. As the location and transform of text are carefully chosen with a learning algorithm, the synthesized images look realistic. This dataset is used for pre-training our model.

ICDAR 2015 Incidental Text (IC15): The ICDAR 2015 Incidental Text dataset [48] issues from the Challenge 4 of the ICDAR 2015 Robust Reading Competition. The dataset is composed of 1000 training images and 500 testing images, which are captured by Google glasses with relatively low resolutions. Each image may contain multi-oriented text. Annotations are provided in terms of word bounding boxes. This dataset also provides 3 lexicons of different sizes for word spotting and end-to-end recognition challenge: 1) strong lexicon which gives 100 words as an individual lexicon for each test image; 2) weakly lexicon containing hundreds of words for the whole test set; 3) generic lexicon with 90k words.

COCO-Text: The COCO-Text dataset [49] is currently the largest dataset for scene text detection and recognition. It contains 43686 training images and 20000 images for validation/testing. The COCO-Text dataset is very challenging since text in this dataset are in arbitrary orientations. This difficulty also holds for annotations which are not as accurate as the other tested datasets in this paper. Therefore, even though this dataset provides oriented annotations, its standard evaluation protocol still relies on horizontal bounding rectangles. For TextBoxes++, we make use of both the annotations in terms of horizontal bounding rectangles and the quadrilateral annotations to train our model. For evaluation, we follow the standard protocol based on horizontal bounding rectangles.

ICDAR 2013 (IC13): The ICDAR 2013 dataset [50] consists of 229 training images and 233 testing images in different resolutions. This dataset contains only horizontal or nearly horizontal text. The lexicon setting for this dataset is the same as the IC15 dataset described before.

Street View Text (SVT): The SVT dataset [51] is more challenging than previous ICDAR 2013 dataset due to lower resolutions of images. There are 100 training images and 250 testing images in the SVT dataset. The images have only horizontal or nearly horizontal text. A lexicon containing 50 words is also provided for each image. Note that not all the text in the dataset are labeled. As a result, this dataset is only used for word spotting evaluation.

Evaluation Protocols: The classical evaluation protocols for text detection, word spotting, and end-to-end recognition all rely on $precision$ (P), $recall$ (R), and $f\text{-}measure$ (F). They are given by:

$$\begin{aligned} P &= \frac{TP}{TP+FP} \\ R &= \frac{TP}{TP+FN} \\ F &= 2 \times \frac{P \times R}{P+R} \end{aligned} \tag{9}$$

where TP, FP, and FN is the number of hit boxes, incorrectly identified boxes, and missed boxes, respectively. For text detection, a detected box b is considered as a hit box if the IOU between b and a ground truth box is larger than a given threshold (generally set to 0.5). The hit boxes in word spotting and end-to-end recognition require not only the same IOU restriction but also correct recognition results. Since there is a trade-off between precision and recall, $f\text{-}measure$ is the most used measurement for performance assessment.

B. Implementation details

TABLE I: Implementation details. "lr" is short for learning rate. The size of input image is denoted as "size"."nr" refers to the negative ratio in hard negative mining. "#iter" stands for the number of training iterations.

Dataset	All datasets			IC15	COCO-Text	SVT	IC13
Settings	lr	size	nr	#iter	#iter	#iter	#iter
Pre-train	10^{-4}	384	3	60k	60k	60k	60k
Stage 1	10^{-4}	384	3	8k	20k	2k	2k
Stage 2	10^{-5}	768	6	4k	30k	8k	8k

TextBoxes++ is trained with Adam [52]. The whole training process is composed of three stages as shown in Table. I. Firstly, we pre-train TextBoxes++ on SynthText dataset for all tested datasets. Then the training process is continued on the corresponding training images of each dataset. Finally, we continue this training with a smaller learning rate and a larger negative ratio. Note also that at the last training stage, a larger input image size is used to achieve better detections of multi-scale text. The number of iterations for the pre-training step is fixed at 60k for all tested datasets. However, this number differs in the second and third training stage which are conducted on each dataset's own training images. This difference is decided by the different dataset size.

Text recognition is performed with a pre-trained CRNN model [12], which is implemented and released by the authors[1].

All the experiments presented in this paper are carried out on a PC equipped with a single Titan Xp GPU. The whole training process (including the pre-training time on SynthText dataset) takes about 2 days on ICDAR 2015 Incidental Text dataset, which is currently the most tested dataset.

C. Quadrilateral VS Rotated Rectangle

The rotated rectangle is an approximate simplification of the quadrilateral, which is more flexible in representing arbitrary-oriented text bounding box. Although both representations

[1]https://github.com/bgshih/crnn

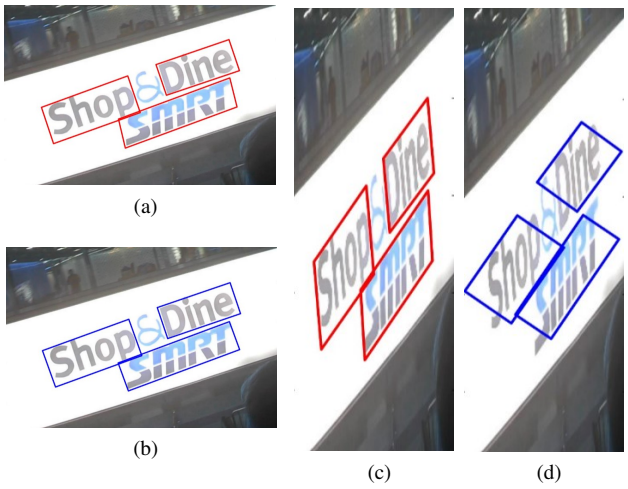


Fig. 7: Ground truth representations in terms of quadrilaterals in red and rotated rectangles in blue. The underlying image in (c) and (d) is resized from the original image shown both in (a) and (b).

TABLE II: Performance comparisons on ICDAR 2015 Incidental Text dataset with different IOU threshold settings between four variants of TextBoxes++, which use different bounding box representation and different input scales. "RR" stands for the rotated rectangle, and "Quad" represents the quadrilateral. "MS" is short for using multi-scale inputs.

Method	IOU_threshold=0.5			IOU_threshold=0.7		
	R	P	F	R	P	F
RR	0.764	0.822	0.792	0.613	0.574	0.593
Quad	0.767	0.872	0.817	0.577	0.676	0.623
RR_MS	0.766	0.875	0.817	0.569	0.623	0.594
Quad_MS	**0.785**	**0.878**	**0.829**	**0.617**	**0.690**	**0.651**

may equally suit for normal text fonts and styles, the quadrilateral representation may adapt better for resized images. An example is given in Fig. 7, where the bounding boxes represented by quadrilaterals and rotated rectangles in the original image (see Fig. 7(a-b)) are almost the same. However, as shown in Fig. 7(c-d), the rotated rectangles match less well the text than quadrilaterals in the resized image. This is because that a rotated rectangle generally become a parallelogram when resized directly, which leads to a small deviation when trying to keep it as a rotated rectangle. In this sense, TextBoxes++ with quadrilateral representation would be more accurate than its variant using rotated rectangles.

We have conducted experiments on the widely tested ICDAR 2015 Incidental Text dataset to compare these two variants of TextBoxes++; The quantitative comparison is given in Table II. Following the standard text detection evaluation, the TextBoxes++ using quadrilateral representation significantly outperforms the version using rotated rectangles with at least 2.5 percents improvements. Note also that using multi-scale inputs would improve both versions of TextBoxes++. The quadrilateral version still performs better especially when the IOU threshold for matching evaluation is set to 0.7. Besides, under such a high IOU threshold setting, the difference between quadrilateral version and rotated rectangle version with multi-scale inputs is more significant. This is because a much more accurate text detector is expected for a high IOU threshold setting. This confirms that quadrilateral representation is more accurate than the rotated rectangle for TextBoxes++. Consequently, we choose the TextBoxes++ using the quadrilateral representation for the rest experiments in this paper and denote it simply as TextBoxes++ when no ambiguity is present. TextBoxes++_MS stands for this version of TextBoxes++ with multi-scale inputs.

TABLE III: Text localization results on ICDAR 2015 Incidental Text dataset.

Methods	recall	precision	f-measure
CNN MSER [48]	0.34	0.35	0.35
AJOU [53]	0.47	0.47	0.47
NJU [48]	0.36	0.70	0.48
StradVision1 [48]	0.46	0.53	0.50
StradVision2 [48]	0.37	0.77	0.50
Zhang et al. [30]	0.43	0.71	0.54
Tian et al. [33]	0.52	0.74	0.61
Yao et al. [38]	0.59	0.72	0.65
Liu et al. [42]	0.682	0.732	0.706
Shi et al. [39]	0.768	0.731	0.750
EAST PVANET2x. RBOX [40]	0.735	0.836	0.782
EAST PVANET2x RBOX MS [40]	0.783	0.833	0.807
TextBoxes++	0.767	0.872	**0.817**
TextBoxes++_MS	**0.785**	**0.878**	**0.829**

TABLE IV: Text localization results on COCO-Text dataset.

Methods	recall	precision	f-measure
Baseline A [49]	0.233	0.8378	0.3648
Baseline B [49]	0.107	0.8973	0.1914
Baseline C [49]	0.047	0.1856	0.0747
Yao et al. [38]	0.271	0.4323	0.3331
Zhou et al. [40]	0.324	0.5039	0.3945
TextBoxes++	0.5600	0.5582	0.5591
TextBoxes++_MS	**0.5670**	**0.6087**	**0.5872**

D. Text localization

TABLE V: Text localization on ICDAR 2013 dataset. P, R, and F refer to precision, recall and f-measure, respectively.

Evaluation protocol	IC13 Eval			DetEval		
Methods	R	P	F	R	P	F
fasttext [54]	0.69	0.84	0.77	–	–	–
MMser [55]	0.70	0.86	0.77	–	–	–
Lu et al. [56]	0.70	0.89	0.78	–	–	–
TextFlow [57]	0.76	0.85	0.80	–	–	–
He et al. [58]	0.76	0.85	0.80	–	–	–
He et al. [59]	0.73	**0.93**	0.82	–	–	–
FCRNall+filts [27]	–	–	–	0.76	0.92	0.83
FCN [30]	0.78	0.88	0.83	–	–	–
Tian et al [60]	0.84	0.84	0.84	–	–	–
Qin et al. [61]	0.79	0.89	0.83	–	–	–
Shi et al. [39]	–	–	–	0.83	0.88	0.85
Tian et al. [33]	–	–	–	0.83	**0.93**	0.88
Tang et al. [62]	0.87	0.92	**0.90**	–	–	–
SSD [10]	0.60	0.80	0.68	0.60	0.80	0.69
TextBoxes [13]	0.74	0.86	0.80	0.74	0.88	0.81
TextBoxes MS [13]	0.83	0.88	0.85	0.83	0.89	0.86
TextBoxes++	0.74	0.86	0.80	0.74	0.88	0.81
TextBoxes++_MS	**0.84**	**0.91**	0.88	**0.86**	0.92	**0.89**

(a) Results given by Zhang et al. [30]

(b) Results using Shi et al. [39]

(c) TextBoxes++ results

Fig. 8: Qualitative comparisons of text detection results on some ICDAR 2015 Incidental text images. Green bounding boxes: correct detections; Red solid boxes: false detections; Red dashed boxes: missed ground truths.

1) Performance: We have first evaluated the proposed TextBoxes++ on two popular oriented text datasets to assess its ability of handling arbitrary-oriented text in natural images. To further validate the versatility of TextBoxes++, we have also tested on two widely used horizontal text datasets.

Oriented Text Dataset: The first tested oriented text dataset is the widely used ICDAR 2015 Incidental text dataset. Some qualitative comparisons are illustrated in Fig. 8. As depicted in this figure, TextBoxes++ is more robust than the competing methods in detecting oriented text and text of a variety of scales. Quantitative results following the standard evaluation protocol is given in Table. III. TextBoxes++ with single input scale outperforms all the state-of-the-art results. More specifically, TextBoxes++ improves the state-of-the-method [40] by 3.5 percents when single scale inputs are used in both methods. Furthermore, the single scale version of TextBoxes++ is still 1.0 percent than the multi-scale version of [40]. Note that a better performance is achieved for TextBoxes++ using multi-scale inputs.

TextBoxes++ also significantly exceeds the state-of-the-art methods on COCO-Text dataset with its latest annotations v1.4. As depicted in Table. IV, TextBoxes++ outperforms the competing methods by at least 16 percents with single scale input. Furthermore, the performance of TextBoxes++ is boosted by 2.81 percents when multi-scale inputs are used.

Horizontal Text Dataset: We have also evaluated TextBoxes++ on ICDAR 2013 dataset, one of the most popular horizontal text dataset. The comparison with some state-of-the-art methods is depicted in Table. V. Note that there are many methods evaluated on this dataset, but only some of the best results are shown. TextBoxes++ achieves at least 1.0 percent improvement over other methods except for [62] on this dataset. However, Tang et al. [62] use a cascaded architecture which contains two networks, taking 1.36 seconds per image. Moreover, it can only detect horizontal text. We have also compared TextBoxes++ with one state-of-the-art general object detector SSD [10], which is the most related method. The same training procedures of TextBoxes++ is used to train SSD for this comparison. As reported in Table. V, such a straightforward adaption of SSD for text detection does not perform as well as the state-of-the-art methods. In particular, we have observed that SSD fails to correctly detect the words with large aspect ratios. TextBoxes++ performs much better thanks to the proposed text-box layers which are specifically designed to overcome the length variation of words. Compared with TextBoxes [13], TextBoxes++ achieves almost the same performance with single scale, and better performance is achieved with multi-scales, owing to the multi-scale training strategy adopted in TextBoxes++. However, note that the experiment on this dataset is to verify that TextBoxes++, although dedicated to arbitrary-oriented text detection, has no performance loss compared with the preliminary study TextBoxes, which is specifically designed for horizontal text detection.

2) Runtime: TextBoxes++ is not only accurate but also efficient. We have compared its runtime with the state-of-the-art methods on ICDAR 2015 Incidental Text dataset. As shown in Table VI, TextBoxes++ achieves an F-measure of 0.817 with 11.6 fps, which has a better balance on runtime and performance than the other competing methods. Note that

(a) Some results on ICDAR 2013 images

(b) Some results on ICDAR 2015 Incidental text images

Fig. 9: Some examples of end-to-end recognition results represented by yellow words. Note that following the evaluation protocol, Words less than 3 letters are ignored. The box colors have the same meaning as Fig. 8.

TABLE VI: Runtime and performance comparison on ICDAR 2015 Incidental Text dataset. "F" is short for F-measure. See corresponding text for detailed discussions.

Method	Res.	Device	FPS	F
Zhang et al. [30]	MS*	Titan X	0.476	0.54
Tian et al. [33]	ss-600*	GPU	7.14	0.61
Yao et al. [38]	480p	K40m	1.61	0.65
EAST PVANET [40]	720p	Titan X	**16.8**	0.757
EAST PVANET2x [40]	720p	Titan X	13.2	0.782
EAST VGG16 [40]	720p	Titan X	6.52	0.764
Shi et al. el. [39]	768×768	Titan X	8.9	0.750
TextBoxes++	1024×1024	Titan X	11.6	**0.817**
TextBoxes++_MS	MS*	Titan X	2.3	**0.829**

ss-600 for the method proposed by Tian et al. [33] means the short side of images is resized to 600. The best result on ICDAR 2015 Incidental dataset for this method is given by using a short edge of 2000, which would lead to a much slower runtime for this method. For Zhang et al. [30], MS means that they used three scales (*i.e.,* 200, 500, 1000) on MSRA-TD500 dataset [19]. The method proposed in [40] performs at 16.8 fps with PVANet [41], a faster backbone compared to VGG-16. However, the performance is 6 percents lower than TextBoxes++. To improve the performance, the authors double the number of channels of PVANet, which results in a runtime at 13.2 fps. TextBoxes++ has a similar runtime but with a 3.5 percents performance improvement. Furthermore, when the same backbone (VGG-16) is applied, the method in [40] is much lower and still performs less well than TextBoxes++. For Shi et al. [39], the reported runtime is tested on 768×768 MSRA-TD 500 images, but the reported performance is achieved with 720×1280 ICDAR 2015 Incidental text images. Note that the runtime for TextBoxes++ on 768×768 COCO-Text images is 19.8 fps. TextBoxes++_MS achieves about 2.3 fps with four input scales (384×384, 768×768, 1024×1024, 1536×1536).

E. Word spotting and end-to-end recognition to refine text detection

1) Word spotting and end-to-end recognition: In the beginning of Section IV, we have discussed the limitations of the standard text detection evaluation protocols which rely on the classical IOU threshold setting. It is meaningless to only detect text without correct recognition. In this sense, an evaluation based on the ultimate purpose of text detection would further assess the quality of text detection. For that, we have also evaluated the proposed text detector TextBoxes++ combined with a recent text recognizer CRNN model [12] in the framework of word spotting and end-to-end recognition. Note that although word spotting is similar to end-to-end recognition, the evaluation of word spotting and end-to-end recognition is slightly different. For word spotting, only some specified words are required to be detected and recognized, which implies that word spotting is generally easier than end-to-end recognition. We have tested the pipeline of TextBoxes++ followed by CRNN model [12] on three popular word spotting or end-to-end recognition benchmark datasets: ICDAR 2015 Incidental Text dataset, SVT dataset, and ICDAR 2013 dataset.

Oriented text datasets: As TextBoxes++ can detect arbitrary-oriented text in natural images, we first evaluate it for word spotting and end-to-end recognition on ICDAR 2015 Incidental Text dataset. Some qualitative results are given in Fig. 9(a). In general, the proposed pipeline correctly recognize most oriented text. A quantitative comparison with other competing methods is depicted in Table. VII. Note that there are not yet published papers for the competing methods in this table. These results are public on the ICDAR 2017 competition website[2], and only some of the best results are shown. Our method significantly outperforms the other methods under all strong, weak, and generic lexicon for both word spotting and end-to-end recognition. More specifically, for strong lexicon, the proposed method outperforms the best competing method

[2]http://rrc.cvc.uab.es

TABLE VII: F-measures for word spotting and end-to-end results on ICDAR 2015 Incidental Text dataset. See the corresponding dataset description in Section IV-A for strong, weak, and generic lexicon settings. Note that the methods marked by "*" are published on the ICDAR 2017 Robust Reading Competition website: http://rrc.cvc.uab.es.

Methods	IC15 word spotting			IC15 end-to-end		
	strong	weak	generic	strong	weak	generic
Megvii-Image++ *	0.4995	0.4271	0.3457	0.4674	0.4	0.3286
Yunos_Robot1.0*	0.4947	0.4947	0.4947	0.4729	0.4729	0.4729
SRC-B-TextProcessingLab*	0.5408	0.5186	0.3712	0.526	0.5019	0.3579
TextProposals + DictNet*	0.56	0.5226	0.4973	0.533	0.4961	0.4718
Baidu IDL*	0.6578	0.6273	0.5165	0.64	0.6138	0.5071
HUST_MCLAB*	0.7057	–	–	0.6786	–	–
TextBoxes++	**0.7645**	**0.6904**	**0.5437**	**0.7334**	**0.6587**	**0.5190**

TABLE VIII: F-measures for word spotting and end-to-end results on ICDAR 2013 dataset. The lexicon settings are the same as for ICDAR 2015 Incidental Text dataset.

Methods	SVT spotting	SVT-50 spotting	IC13 spotting			IC13 end-to-end		
			strong	weak	generic	strong	weak	generic
Alsharif [63]	–	0.48	–	–	–	–	–	–
Jaderberg [8]	0.56	0.68	–	–	0.76	–	–	–
FCRNall+filts [27]	0.53	0.76	–	–	0.85	–	–	–
TextBoxes	**0.64**	**0.84**	0.94	0.92	**0.87**	0.91	0.89	0.84
TextBoxes++	**0.64**	**0.84**	**0.96**	**0.95**	**0.87**	**0.93**	**0.92**	**0.85**

TABLE IX: Refined detection results with recognition. The evaluation method for ICDAR 2013 dataset is the IC13 Eval. "Det": TextBoxes++_MS; "Rec": recognition without lexicon; "Rec-lex" : recognition with the given strong lexicon in each dataset.

Datasets	IC13			IC15		
	R	P	F	R	P	F
Det	0.844	0.912	0.876	0.785	0.878	0.829
Det+Rec	**0.847**	0.918	0.881	**0.804**	0.881	0.842
Det+Rec-lex	0.838	**0.957**	**0.894**	0.792	**0.912**	**0.848**

by 6 percent for both tasks. For weak lexicon, the proposed method improves the state-of-the-art results by 6 percents for word spotting and 4 percents for end-to-end recognition. The improvement is less significant (2.7 percents and 1.2 percents for the two tasks, respectively) when a generic lexicon is used.

Horizontal text datasets: We have also evaluated the proposed method for word spotting and end-to-end recognition on two horizontal text datasets: ICDAR 2013 dataset and SVT dataset. Some qualitative results on ICDAR 2013 dataset are depicted in Fig. 9. In general, TextBoxes++ achieves good results in various occasions, regardless of the sizes, aspect ratios, fonts, and complex backgrounds. Some quantitative results are given in Table. VIII. Our method outperforms the state-of-the-art methods. More specifically, on ICDAR 2013 dataset, our method outperforms the best competing method by at least 2 percents for all the evaluation protocols listed in Table. VIII. The performance improvement is even more significant on SVT dataset. TextBoxes++ outperforms the state-of-the-art method [27] by at least 8 percents on both SVT and SVT-50. This is mainly because that TextBoxes++ is more robust when dealing with low-resolution images in SVT thanks to its training on relatively low-resolution images. Note that Jaderberg [8] and FCRNall+filts [27] adopt a much smaller lexicon (50k words) than our method (90k words), yet the proposed method still performs better. Compared with TextBoxes [13], TextBoxes++ achieves better performance on ICDAR 2013 dataset.

2) Refining detection with recognition: We propose to use recognition to further refine detection results by integrating recognition score into detection score with Eq. (8). We have evaluated this idea on two datasets. As depicted in Tab. IX, the recognition without lexicon improves detection results of TextBoxes++ by 0.5 percent and 1.3 percents on ICDAR 2013 and ICDAR 2015 Incidental Text dataset, respectively. This improvement is further boosted using a specified lexicon, achieving 0.8 percent and 1.9 percents on ICDAR 2013 and ICDAR 2015 Incidental Text dataset, respectively. Note that the current text recognizer still has difficulties in dealing with vertical text and recognizing low-resolution text. A further performance improvement is expected with a better text recognizer.

F. Weaknesses

As demonstrated by previous experimental results, TextBoxes++ performs well in most situations. However, it still fails to handle some difficult cases, such as object occlusion and large character spacing. TextBoxes++ also fails to detect some vertical text due to the lack of enough vertical training data. Even with hard negative mining, some text-like areas are still falsely detected. Another failure case is curved text detection. Different from some part-based methods, *e.g.* [23], TextBoxes++ is hard to fit the accurate boundary for curved texts due to the limitation of quadrilateral representation. Note that all these difficulties also hold for the other state-of-the-art methods [39], [40]. Some failure cases are shown in Fig. 10.

Fig. 10: Some failure examples. Green bounding boxes: correct detections; Red solid boxes: false detections; Red dashed boxes: missed ground truths.

V. Comparisons with recent works

We compare in detail the proposed TextBoxes++ with EAST [40], one of the previous state-of-the-art method, and one of the most related method DMPNet [42] from two aspects: simplicity and performance.

A. TextBoxes++ vs. EAST

1) Simplicity: EAST generates a text region map, or named as score map, using a U-shape network [64]. It also regresses the oriented rectangles or quadrilaterals based on the same feature which generates the score map. It is a combination of segmentation and detection. In this way, it relies on pyramid-like deconvolutional layers for accurate segmentation. This extra pyramid-like deconvolutional layers require additional computation. However, TextBoxes++ directly classifies and regresses the default boxes on the convolutional feature maps, which is much simpler, avoiding the time consuming on pyramid-like deconvolution. This is evidenced by the speed comparison shown in Tab VI, where TextBoxes++ (with a VGG-16 backbone) has a speed of 11.6fps while the VGG16 RBOX version of EAST runs at 6.52fps.

2) Performance: EAST relies on an accurate segmentation score map as the score of the bounding boxes. Yet, the text region segmentation is challenging in itself. If the score map is not accurate enough, it is difficult to achieve correct results. For example, it is possible that the partition between two close words is predicted as text region in the segmentation score map, in this case, it is rather difficult to separate these two words in the detection. To alleviate this problem, EAST shrinks the text region in the ground truth of segmentation score map. TextBoxes++ does not suffer from such limitations. It relies on default boxes, and regresses the bounding boxes directly from the convolutional feature maps, where richer information is reserved as compared to the segmentation score map. Thus, TextBoxes++ achieves higher performance (see Table. III). Specifically, TextBoxes++ outperforms [42] by 3.5 percents (PVANET2x RBOX version) and 6 percents (VGG16 RBOX version) with a single scale.

B. TextBoxes++ vs. DMPNet

1) Simplicity: 1) TextBoxes++ uses horizontal rectangles as default boxes instead of quadrilaterals with different orientations used in [42]. In this way, we use much less default boxes in every region. Furthermore, we argue that the receptive fields of the convolutional feature map are all in terms of horizontal rectangles, so the target quadrilaterals can be well regressed if it matches the receptive field. The use of oriented rectangle default boxes adopted in [42] is not necessary for general scene text detection. 2) Benefiting from the horizontal rectangle default boxes, TextBoxes++ enjoys a much simpler strategy for matching default boxes and ground truth by using the maximum horizontal rectangles instead of quadrilaterals. In fact, computing the intersection area between two horizontal rectangles is much easier (just using subtract operation and multiply operation once) than computing the intersection area between two arbitrary quadrilaterals, even though a Monte-Carlo method is used in [42].

2) Performance: 1) TextBoxes++ simultaneously regresses the maximum horizontal rectangles of the bounding boxes and the quadrilateral bounding boxes, which makes the training more stable than the method in [42]. 2) As compared to the method in [42], TextBoxes++ goes further study on small texts in the images. We adopt a new scheme for data augmentation which is beneficial to small texts. 3) In this paper, we not only focus on scene text detection, but also concern the combination between detection and recognition. We proposed a novel score which effectively and efficiently combines the detection scores and the recognition scores.

It is regretful that DMPNet [42] did not report the runtime. However, we argue that it is slower than TextBoxes++ based on above analysis. Moreover, TextBoxes++ outperforms DMP-Net [42] by 11 percents (with a single scale setting) in terms of F-measure on ICDAR 2015 dataset (see Tab. III).

VI. Conclusion

We have presented TextBoxes++, an end-to-end fully convolutional network for arbitrary-oriented text detection, which is highly stable and efficient to generate word proposals against cluttered backgrounds. The proposed method directly predicts arbitrary-oriented word bounding boxes via a novel regression model by quadrilateral representation. The comprehensive evaluations and comparisons on some popular benchmark datasets for text detection, word spotting, and end-to-end scene text recognition, clearly validate the advantages of TextBoxes++. In all experiments, TextBoxes++ has achieved state-of-the-art performance with high efficiency for both horizontal text datasets and oriented text datasets. In the future, we plan to investigate the common failure cases (*e.g.*, large character spacing and vertical text) faced by almost all state-of-the-art text detectors.

REFERENCES

[1] C. Yi and Y. Tian, "Scene text recognition in mobile applications by character descriptor and structure configuration," *IEEE Trans. Image Processing*, vol. 23, no. 7, pp. 2972–2982, 2014.

[2] B. Xiong and K. Grauman, "Text detection in stores using a repetition prior," in *Proc. WACV*, 2016, pp. 1–9.

[3] C. Kang, G. Kim, and S. I. Yoo, "Detection and recognition of text embedded in online images via neural context models," in *Proc. AAAI*, 2017, pp. 4103–4110.

[4] X. Rong, C. Yi, and Y. Tian, "Recognizing text-based traffic guide panels with cascaded localization network," in *Proc. ECCV*, 2016, pp. 109–121.

[5] Q. Ye and D. Doermann, "Text detection and recognition in imagery: A survey," *IEEE TPAMI*, vol. 37, no. 7, pp. 1480–1500, 2015.

[6] Y.-F. Pan, X. Hou, and C.-L. Liu, "A hybrid approach to detect and localize texts in natural scene images," *IEEE T. Image Proc.*, vol. 20, no. 3, pp. 800–813, 2011.

[7] L. Neumann and J. Matas, "Real-time scene text localization and recognition," in *Proc. CVPR*, 2012, pp. 3538–3545.

[8] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *IJCV*, vol. 116, no. 1, pp. 1–20, 2016.

[9] B. Bai, F. Yin, and C. L. Liu, "Scene text localization using gradient local correlation," in *Proc. ICDAR*, 2013, pp. 1380–1384.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed, "SSD: single shot multibox detector," in *Proc. ECCV*, 2016.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015.

[12] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE TPAMI*, vol. 39, no. 11, pp. 2298–2304, 2017.

[13] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "Textboxes: A fast text detector with a single deep neural network," in *Proc. AAAI*, 2017, pp. 4161–4167.

[14] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014.

[15] R. B. Girshick, "Fast R-CNN," in *Proc. ICCV*, 2015.

[16] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. CVPR*, 2016.

[17] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *IJCV*, vol. 104, no. 2, pp. 154–171, 2013.

[18] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[19] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, "Detecting texts of arbitrary orientations in natural images," in *Proc. CVPR*, 2012, pp. 1083–1090.

[20] Y. Li, W. Jia, C. Shen, and A. van den Hengel, "Characterness: An indicator of text in the wild," *IEEE Trans. Image Processing*, vol. 23, no. 4, pp. 1666–1677, 2014.

[21] W. Huang, Y. Qiao, and X. Tang, "Robust scene text detection with convolution neural network induced mser trees," in *Proc. ECCV*, 2014.

[22] L. Gomez and D. Karatzas, "Multi-script text extraction from natural scenes," in *Proc. ICDAR*, 2013, pp. 467–471.

[23] Y. Guo, Y. Sun, P. Bauer, J. P. Allebach, and C. A. Bouman, "Text line detection based on cost optimized local text line direction estimation," in *Proc. SPIE 9395, Color Imaging XX: Displaying, Processing, Hardcopy, and Applications, 939507*, 2015.

[24] M. Zhao, S. Li, and J. T. Kwok, "Text detection in images using sparse representation with discriminative dictionaries," *Image Vision Comput.*, vol. 28, no. 12, pp. 1590–1599, 2010.

[25] Z. Zhong, L. Jin, S. Zhang, and Z. Feng, "Deeptext: A unified framework for text proposal generation and text detection in natural images," *CoRR*, vol. abs/1605.07314, 2016.

[26] L. Gomez-Bigorda and D. Karatzas, "Textproposals: a text-specific selective search algorithm for word spotting in the wild," *Pattern Recognition*, vol. 70, pp. 60–74, 2017.

[27] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proc. CVPR*, 2016.

[28] Z. Zhang, W. Shen, C. Yao, and X. Bai, "Symmetry-based text line detection in natural scenes," in *Proc. CVPR*, 2015, pp. 2558–2567.

[29] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, 2015.

[30] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai, "Multi-oriented text detection with fully convolutional networks," in *Proc. CVPR*, 2016.

[31] X. Chen and A. L. Yuille, "Detecting and reading text in natural scenes," in *Proc. CVPR*, 2004, pp. 366–373.

[32] C. Yi and Y. Tian, "Localizing text in scene images by boundary clustering, stroke segmentation, and string fragment classification," *IEEE Trans. Image Processing*, vol. 21, no. 9, pp. 4256–4268, 2012.

[33] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, "Detecting text in natural image with connectionist text proposal network," in *Proc. ECCV*, 2016.

[34] X. Cao, W. Ren, W. Zuo, X. Guo, and H. Foroosh, "Scene text deblurring using text-specific multiscale dictionaries," *IEEE Trans. Image Processing*, vol. 24, no. 4, pp. 1302–1314, 2015.

[35] L. Kang, Y. Li, and D. S. Doermann, "Orientation robust text line detection in natural images," in *Proc. CVPR*, 2014, pp. 4034–4041.

[36] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and vision computing*, vol. 22, no. 10, pp. 761–767, 2004.

[37] C. Yao, X. Bai, and W. Liu, "A unified framework for multioriented text detection and recognition," *IEEE Trans. Image Processing*, vol. 23, no. 11, pp. 4737–4749, 2014.

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

[39] B. Shi, X. Bai, and S. J. Belongie, "Detecting oriented text in natural images by linking segments," in *Proc. CVPR*, 2017, pp. 3482–3490.

[40] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: an efficient and accurate scene text detector," in *Proc. CVPR*, 2017, pp. 2642–2651.

[41] K. Kim, Y. Cheon, S. Hong, B. Roh, and M. Park, "PVANET: deep but lightweight neural networks for real-time object detection," *CoRR*, vol. abs/1608.08021, 2016.

[42] Y. Liu and L. Jin, "Deep matching prior network: Toward tighter multi-oriented text detection," in *Proc. CVPR*, 2017.

[43] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[44] Y. Jiang, X. Zhu, X. Wang, S. Yang, W. Li, H. Wang, P. Fu, and Z. Luo, "R2CNN: rotational region CNN for orientation robust scene text detection," *CoRR*, vol. abs/1706.09579, 2017.

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

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, 2015.

[47] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, pp. 369–376.

[48] 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 *Proc. ICDAR*, 2015, pp. 1156–1160.

[49] A. Veit, T. Matera, L. Neumann, J. Matas, and S. J. Belongie, "Coco-text: Dataset and benchmark for text detection and recognition in natural images," *CoRR*, vol. abs/1601.07140, 2016.

[50] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. de las Heras, "Icdar 2013 robust reading competition," in *ICDAR*, 2013, pp. 1484–1493.

[51] K. Wang and S. Belongie, "Word spotting in the wild," in *Proc. ECCV*, 2010, pp. 591–604.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[53] H. I. Koo and D. H. Kim, "Scene text detection via connected component clustering and nontext filtering," *IEEE Trans. Image Processing*, vol. 22, no. 6, pp. 2296–2305, 2013.

[54] M. Busta, L. Neumann, and J. Matas, "Fastext: Efficient unconstrained scene text detector," in *Proc. ICCV*, 2015, pp. 1206–1214.

[55] A. Zamberletti, L. Noce, and I. Gallo, "Text localization based on fast feature pyramids and multi-resolution maximally stable extremal regions," in *Proc. ACCV*, 2014, pp. 91–105.

[56] S. Lu, T. Chen, S. Tian, J.-H. Lim, and C.-L. Tan, "Scene text extraction based on edges and support vector regression," *IJDAR*, vol. 18, no. 2, pp. 125–135, 2015.

[57] S. Tian, Y. Pan, C. Huang, S. Lu, K. Yu, and C. Lim Tan, "Text flow: A unified text detection system in natural scene images," in *Proc. ICCV*, 2015.

[58] D. He, X. Yang, W. Huang, Z. Zhou, D. Kifer, and C. L. Giles, "Aggregating local context for accurate scene text detection," in *Proc. ACCV*, 2016, pp. 280–296.

[59] T. He, W. Huang, Y. Qiao, and J. Yao, "Text-attentional convolutional neural network for scene text detection," *IEEE Trans. Image Processing*, vol. 25, no. 6, pp. 2529–2541, 2016.
[60] C. Tian, Y. Xia, X. Zhang, and X. Gao, "Natural scene text detection with mc–mr candidate extraction and coarse-to-fine filtering," *Neurocomputing*, 2017.
[61] S. Qin and R. Manduchi, "A fast and robust text spotter," in *Proc. WACV*, 2016, pp. 1–8.
[62] Y. Tang and X. Wu, "Scene text detection and segmentation based on cascaded convolution neural networks," *IEEE Trans. Image Processing*, vol. 26, no. 3, pp. 1509–1520, 2017.
[63] O. Alsharif and J. Pineau, "End-to-end text recognition with hybrid HMM maxout models," *CoRR*, vol. abs/1310.1811, 2013.
[64] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*, 2015.