

Self-organized Text Detection with Minimal Post-processing via Border Learning

Yue Wu and Prem Natarajan
Information Sciences Institute
4676 Admiralty Way, #1001, Marina Del Rey, CA, USA
{yue_wu, pnatarajan}@isi.edu

Abstract

In this paper we propose a new solution to the text detection problem via border learning. Specifically, we make four major contributions: 1) We analyze the insufficiencies of the classic non-text and text settings for text detection. 2) We introduce the border class to the text detection problem for the first time, and validate that the decoding process is largely simplified with the help of text border. 3) We collect and release a new text detection PPT dataset containing 10,692 images with non-text, border, and text annotations. 4) We develop a lightweight (only 0.28M parameters), fully convolutional network (FCN) to effectively learn borders in text images. The results of our extensive experiments show that the proposed solution achieves comparable performance, and often outperforms state-of-the-art approaches on standard benchmarks—even though our solution only requires minimal post-processing to parse a bounding box from a detected text map, while others often require heavy post-processing.

1. Introduction

As one of the most well-known and oldest problems in computer vision [20, 27, 7, 43], text detection has attracted increased attention in recent years [21, 13, 16, 33, 36, 22, 11, 39, 3, 35, 28, 1, 41, 12, 24, 9, 40, 38, 4, 42, 5, 8, 31, 25, 29, 19, 34, 15], bringing benefits to many real-world applications, including but not limited to image search, document analysis, automatic driving, and human-computer interactions. A good review of the text detection problem can be found in [37].

Although the text detection problem seems to be self-explanatory, namely detecting texts in a given image, it can be interpreted in many different ways according to the used detection unit, *e.g.*, component detection [2, 33, 17, 22, 39], character detection [19, 40, 5, 38, 24, 28, 29], word detection [10], line detection [41], and region/block detection [42, 34]. From the perspective of detection speed, the component detection is the most handy because components can

be efficiently extracted from low-level features like edges, and some heuristic rules are also effective to distinguish text from non-text. From the perspective of rigorous problem definition, character detection is highly preferred because a character is naturally a unit in any language, and there are always a small number of characters. From the perspective of human readable results, the most preferred is either the word- or line-level detection. From the perspective of reducing false alarms, the region/block-level detection is preferred because it contains a larger image context region to differentiate text from non-text. However, text detection is not an isolated task, but is closely related to text recognition. From the perspective of a down-streaming recognition system, the most preferred is the line- or word-level detection depending on whether a recognition system is trained by text lines or isolated words. Consequently, there are three common strategies: 1) bottom-up [1, 24, 5, 38, 29, 19]: first detect text at the component- or character-level, and convert initial detection results to a desired word- or line-level; 2) top-down [42]: first detect text on the block level, and break a block to a word- or line-level if necessary; and 3) hybrid [34]: take advantage of both bottom-up and top-down strategies. Actually, these are the mainstream frameworks used in text detection until now, although people may use different component/character detectors, rules to connect character bounding boxes to a word/line, rules to divide a region into words/lines, etc.

Early efforts under the mainstream frameworks focus on various heuristics that are helpful for detecting characters or character components. The two most well-known ones are: 1) maximally stable extremal regions (MSER) [2, 18, 22] and 2) stroke-width transform (SWT) [6, 17, 23]. MSER assumes that text components have similar homogeneous background and foreground and thus are stable with respect to a range of thresholds. SWT assumes that text components have comparable stroke width, and thus finding components with comparable stroke width finds text. Both MSER and SWT are very useful heuristics, but they have to be combined with additional post-processing to produce reasonable text candidates.

More recent efforts focus on reducing the amount of handcrafted features or man-made rules in text detection, especially after the big success of deep convolutional neural networks (CNN) in the ImageNet Challenge [14]. These recent approaches show very exciting performance scores on both document images and scene text images. Tian *et al.* [24] proposed the *Text Flow* method that sequences character CNN candidate detection, false character removal, text line extraction and text line verification using minimum cost flow networks. Cho *et al.* [5] suggested the *Canny Text Detector* that proposes character candidates using maximum stable regions and edge similarities, text line tracking and grouping using heuristic rules. Zhang *et al.* [42] suggests a fully convolutional neural network (FCN) to obtain text block candidates, a character-centroid FCN to assist text line generation from a text block, and a set of heuristic rules based on intensity and geometry consistencies to reject false candidates. Yao *et al.* [34] constructs a similarity graph based on block-level text detection results, character-level text detection results and estimated text orientations, and proposes line-level predictions via region splitting according to graph similarities.

There is nothing wrong with those mainstream frameworks, but they are **not** desired for neither training nor decoding. The multi-step nature of the framework implies that a text detection system of this type is not fully optimized even if each individual step is optimized. In other words, it is very difficult to tune a text detection system in an end-to-end manner. Furthermore, obtaining line-level text candidates from candidates of other levels is not an easy task due to variations in both image foreground and background. It is not rare to see that authors [33, 42, 5, 24, 38] may spend 1/4 to 2/3 of all paper pages discussing how to obtain good line-level detection results via all kinds of post-processing—ranging from very simple operations like finding connected components to complicated region operations [33, 19] like grouping, division, chaining, linkage, etc., and even more complicated operations like coarse-to-fine analysis [38, 9], data clustering [38], additional classifiers [40, 29], conditional random fields [28], etc. At the end of the day, one may spend more time on tuning post-processing than on the core “text detection” algorithm.

Instead of considering only non-text and text classes [33, 22, 11, 39, 3, 35, 28, 41, 24, 9, 40, 38, 4, 42, 5, 31, 25, 29, 19, 34], we consider three classes, namely *non-text*, *border* and *text* classes, and propose a new text detection framework that self-organizes predicted text groups as text line candidates. As far as we know, this is the first attempt to tackle the text detection problem using the border class. The introduction of the new border class makes a huge difference in both training and decoding. In training, all kinds of heuristics about text lines and semantic segmentation are now implicitly encoded into the border and text

classes. Thus, we not only train a text detection system in an end-to-end manner, but also avoid learning and tuning explicit post-processing related rules, classifiers, and other parameters. In testing, we can directly obtain meaningful text candidates from a predicted probability map, because borders help isolate each individual text region from the others (see examples in Fig. 2). In this way, we achieve the self-organized text detection. In addition to this main contribution, we also make an effort to 1) collect and release a text detection *PPT* dataset with single-column and single-line annotations, and 2) develop a lightweight FCN supporting multi-resolution analysis for self-organized text detection. Our experimental results on the public benchmarks of ICDAR 2015 Robust Reading Competition Task 1.1 (born-digital images) and 2.1 (focused scene text), and the MSRA-TD500 dataset support that the proposed new framework outperforms the state-of-the-art approaches with heavy post-processing and parameter tuning.

The remainder of this paper is organized as follows: Sec. 2 discusses our motivations; Sec. 3. proposes our newly created *PPT* dataset and baseline FCN; Sec. 4 shows our experimental results on public benchmarks; and we conclude our paper in Sec. 5.

2. Methodology

2.1. Motivation

When we analyze why so much post-processing is involved under the mainstream text detection frameworks, we notice that it is mainly to close the gap between text candidates at a desired level (line-level) and initial candidates at undesired levels (character-level, component-level or block-level). However, why do we have to format text candidates at a desired level from initial candidates at other levels? Why can’t we directly predict text candidates at a desired line-level?

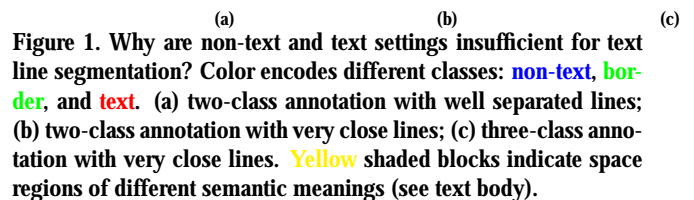


Figure 1. Why are non-text and text settings insufficient for text line segmentation? Color encodes different classes: **non-text**, **border**, and **text**. (a) two-class annotation with well separated lines; (b) two-class annotation with very close lines; (c) three-class annotation with very close lines. Yellow shaded blocks indicate space regions of different semantic meanings (see text body).

Although we failed to find any answer from existing literature, we found that [42, 34] use line-level annotations during training. Intuitively, if they train a system using line-level annotations, then the resulting system should also give line-level text candidates. Unfortunately, this is not the case: as Yao *et al.* stated in [34], “when several text

Figure 2. Sample results of our self-organized text detection. Upper row: original images; middle row: predicted three-class probability map (non-text, border, and text); and lower row: rectangular bounding boxes drawn for each text line.

lines are very close, the estimated text regions may stick together, making it difficult to identify each individual text line.” These lessons teach us that pure line-level annotations are **not** sufficient to train a text detection system to predict text line candidates (see examples in Fig. 1 of [42] and Fig. 1 and 2 of [34]).

However, two important but unanswered questions remain: **why are line-level annotations not sufficient?** and **what can be done to resolve this problem?** After several rounds of failure analysis we determined the problem was an inherent limitation of the two-class (*i.e.* text and non-text) settings in text detection. As seen in Fig. 1-(a), when lines are separated by sufficient space, two “text” lines on an annotated group truth map can be easily identified. When the space between two lines becomes small enough, it becomes very hard to distinguish the two lines, and two “text” lines merge to one on a corresponding annotated group as shown in Fig. 1-(b). Hence, it is not surprising to see that adjacent text lines eventually stick together and are predicted as a single one in testing.

2.2. The Border Class for Text Detection

Instead of seeking additional post-processing to separate multiple text lines, we propose a new solution by introducing a third class, namely the **border class**, to the text detection problem. As shown in Fig. 1-(c), two text lines will never be indistinguishable on the ground truth map after the border class is used. Moreover, visually similar blocks, *i.e.*, those yellow blocks in Fig. 1-(c), are now labeled differently according to physical meanings, *i.e.* #1: space in background (non-text), #2: space between text lines (border), and #3: space between words (text).

Now we are ready to reformulate the text detection prob-

lem: **Given a text image, our aim is to find all text candidates according to detected borders.** Fig. 2 shows sample detection results from our system, which is a fully convolutional network trained with the three-class annotations (see Sec. 3). It is clear that with the help of the border class: 1) we are able to distinguish one text candidate from the other, and 2) a text candidate is no longer restricted to only a rectangular shape, but could be an arbitrary shape.

3. Text Detection via Border Learning

3.1. Data Collection and Border Annotation

To initiate self-organized text detection, we first need enough training data. Specifically, we expect to have a dataset that is big enough (*e.g.*, > 10000 sample images) and is annotated in the format of non-text, border, and text classes. Although there are many public datasets for text detection, *e.g.*, MSRA-TD500 [33], HUST-TR400 [32], COCO-Text [26], and so on, none of them meet our expectations. We therefore create our own dataset.

In total, our *PPT* dataset contains 10,692 images (72 dpi) with 93,228 text regions mixed of English and Arabic. As its name implies, all source images are converted from PowerPoint slides, which are downloaded from the Internet through the Google search API. Text regions are first automatically retrieved by using the Office 365 API for each image, and saved for later human inspection to ensure each text region is single-line and single-column.

A typical slide of aspect ratio 4:3 is of size 720×540 pixels after converting to an image. Table 1 summarizes the statistics of the text line height at different percentiles, where “Abs. LH” computes the actual line height of each text region in pixels; “Rel. LH” computes the relative line

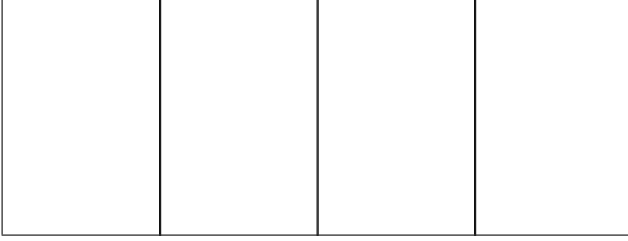


Figure 3. Samples images (upper row) and annotations (lower row) of the *PPT* dataset, where target colors encode the probability map of classes **non-text**, **border**, and **text**.

Table 1. Line height (LH) statistics of *PPT* dataset.

Name \ Perc.	1	10	20	30	40	50	60	70	80	90	99
Abs. LH (pixel)	85	57	49	44	40	37	34	32	29	24	9
Rel. LH (%)	11.55	7.78	6.56	6.00	5.48	5.11	4.78	4.11	3.96	3.37	1.56
{max/min}(LH)	27.69	3.46	2.24	1.85	1.63	1.49	1.37	1.28	1.20	1.13	1.00

height of each text region normalized by corresponding image height; and “{max/min}(LH)” computes the ratio of the largest line height to the smallest line height for all text regions within an image. Each element in the table tells the value of the top p percentile of a given variable. For example, the element 85 at the row “Abs. LH” and column Perc = 1 indicates that 1% of the entire data are of “Abs. LH”—above or equal to 85 pixels.

To effectively learn borders, we also annotate border pixels for each text region by 1) estimating the original text region line height \hat{h} , and 2) marking the outer ribbon-like region of width $c \cdot \hat{h}$ as text borders. For each slide image in our dataset, we then generate a “target” of the three classes, *i.e.*, non-text, border, and text. This completes our *PPT* dataset. Fig. 3 shows samples of the *PPT* dataset and automatically generated target images (border width coefficient $c = 15\%$); this entire dataset can be downloaded from <https://gitlab.com/rex-yue-wu/ISI-PPT-Dataset>.

3.2. Baseline Training

We partition the *PPT* dataset into training (90%), validation (5%), and testing (5%) sets. Instead of choosing a popular architecture inspired by a different problem, *e.g.* ImageNet object classification, we develop our own lightweight architecture as shown in Fig. 4, which is a simple feed-forward network with targets defined as the probability map of the three classes, *i.e.*, non-text, border, and text. The reason why a lightweight architecture is still capable of text detection is that text is, in general a simpler class than various physical object classes like cat and dog, in the sense that it often has a homogeneous foreground and surroundings. Both single and multi-resolution baselines are implemented in *Theano* and trained w.r.t. the cross-entropy loss.

With regard to data argumentation, we use 1) random resize for a ratio in $[0.25, 1.25]$; 2) random color shift in $[-64, 64]$; 3) random negation; and 4) random rotation for an angle within $[-25^\circ, +25^\circ]$. We use the *adadel ta* optimizer and batch size 16 in training, and the single resolution baseline model gets converged after about 150 epochs. Once we obtain the single reso-

lution baseline model, we plug it into our multi-resolution baseline model and continue training until convergence. This multi-resolution baseline predicts non-text, border, and text probability maps on four scales and fuses all maps together. In this way, we handle the font size variations in the *PPT* dataset.

Algorithm 1: Rectangular text region decoder.

```

1 Function RTRD( $Y$ ):
   Input :  $Y$ , a predicted probability map of non-text, border and text.
   Output:  $L$ , a list of decoded text bounding boxes
2   for each pixel in  $Y$ , decide its membership;
3   obtain a mask  $M$  by assigning 1s to pixels belong to the text class and
   0s otherwise;
4   initialize  $L$  to a empty list;
5   for  $r$  connected component of  $M$  do
6     find out text coordinates  $\{(x, y) | (x, y) \in r\}$ ;
7     top = min $\{y \in r\}$ , bot = max $\{y \in r\}$ ;
8     left = min $\{x \in r\}$ , right = max $\{x \in r\}$ ;
9     height = bot-top, width = right-left;
10    relax this rectangular box according to the border coefficient  $c$ 
    used in training data;
11    append this relaxed rectangular box to  $L$ 
12  end

```

3.3. Self-Organized Text Detector

One salient advantage of the proposed method is that we predict non-text, border and text. The knowledge of borders is very important for two reasons: 1) border separates close-by regions, and 2) the concurrence of both text class and border class could help further improve text region prediction. As one can see from Fig. 2, a predicted non-text, border, and text probability map has already clearly identified text regions and separated near-by regions via borders. Therefore, we could obtain reasonably well text candidates with minimal postprocessing, namely simply analyzing connected components. Alg. 1 shows an example of how to obtain horizontal rectangular bounding boxes from a predicted probability map, but one can easily modify it to predict text bounding boxes in other forms, *e.g.* an oriented bounding box. Our pretrained models are available at <https://gitlab.com/rex-yue-wu/ISI-PPT-Text-Detector>.

3.4. The Effect of Border Class in Text Detection

To understand the effect of the proposed border class in text detection, we evaluate models trained with and without using the border class. Specifically, besides the two proposed models shown in Fig. 4, we train two additional models: 1) a single-resolution FCN variant without border, which adopts the exact same architecture as the single resolution FCN, except for changing the last layer from predicting three channels with **softmax** activation to predicting only one channel with **sigmoid** activation to determine text or non-text; and 2) a state-of-the-art text-block FCN [42] which does not use the border class. All four models share the same training, validation and testing dataset. Model performance is evaluated by the widely used Wolf’s object detection evaluation method [30] with default parameters¹ as shown in Table 2. Decoding speed is estimated based on the NVidia TitanX GPU and the Intel Xeon CPU E5-2695 v2 @2.4GHz. From Table 2,

¹<http://iris.cnrs.fr/christian.wolf/software/deteval/>

Figure 4. Fully convolutional networks used in the proposed method. Left: single resolution FCN. Right: multi-resolution FCN.

it is clear that 1) using a lightweight architecture (2% of [42]’s #param.) greatly improves decoding speed by 2x while not trading off detection quality (see row 1-2); 2) the border class helps better localize a text region and thus largely improve text detection by **0.27** in F-score (see row 2-3); and 3) multi-resolution analysis further boosts detection performance to **0.96** in F-score (see row 3-4). Qualitative results can be found in Fig. 5.

Table 2. Performance comparisons on *PPT* testing Dataset

Models	Prec.	Recall	F-score	#Param.(M)	sec/img (GPU/CPU)
Text-Block FCN[42]	0.63	0.67	0.65	14.71	0.55 / 14.7
single-res. w/o border	0.61	0.69	0.65	0.27	0.24 / 7.2
single-res. w/ border	0.91	0.94	0.92	0.27	0.25 / 7.4
multi-res. w/ border	0.94	0.97	0.96	0.28	0.32 / 9.8

Figure 5. Effectiveness of the border class for text detection. (a), (b), and (c) are decoding results on the testing *PPT* dataset using models of single-res. w/o border, single-res. w/ border, and multi-res. w/ border, respectively. See differences in arrowed regions.

4. Experiment Results

4.1. Datasets

To validate the performance of the proposed new framework, we evaluate our method on the latest ICDAR 2015 Robust Reading Competition dataset,² namely Task 1.1 (born-digital images), and Task 2.1 (focused scene text), and the oriented scene text dataset MSRA-TD400 [33].

ICDAR 2015 Task 1.1 dataset contains images of born-digital *e.g.* online advertisement images. It was first introduced in the ICDAR 2011 Robust Reading Competition [21]. In total, it contains 551 images, where 410 images belong to the training set. ICDAR 2015 Task 2.1 dataset (also known as ICDAR 2013 dataset [13]) is composed of “focused scene text” images, whose text regions are placed nearly frontal and horizontal. The total number of images in this dataset is 462, and 229 belong to the training set. MSRA-TD500 was introduced by [33] and contains 500 images with scene-text (English and Chinese) of different orientations.

Although the proposed border learning method is also applicable to **word-level** text detection, which is the ground truth files’ format of the used ICDAR datasets, we use synthesized **line-level** annotation instead for two reasons: 1) the provided horizontal rectangular boxes for rotated texts may mix non-text regions and thus are inaccurate for border learning; 2) our pretrained models use on the line-level annotation, and it is better to finetune it to a dataset with the same annotation type than a different one. Specifically, we synthesize a new version of line-level annotation for each training image by simply merging word-level annotations if they are on the same horizontal line. Human inspection is involved to verify and correct the automatically generated line-level annotations. It is worth noting that an oriented text region in this dataset is originally treated as a horizontal one (see Fig. 6-left), but we manually correct it to a polygon covering the oriented text region. Fig. 6 shows the difference between the original ICDAR 2015 Task 1.1 (born-digital images) annotation and the variant we used for training. This line-level annotation can be provided upon request. Finally, we apply the same trick as we did in the baseline training to obtain

²<http://rrc.cvc.uab.es/>

annotations of the border class.

Figure 6. Original ICDAR 2015 Task 1.1 (born-digital images) word-level annotation (left) and our modified line-level annotation (right).

4.2. Fine-Tuning Settings

Fine-tuning is performed with respect to each individual dataset. We reuse all settings in the multi-resolution baseline training, except that we resize a training image when its size is too large. Specifically, we keep all images whose dimensions are below 540×720 unchanged, and only resize images whose dimensions are larger than 540×720 (720×540), while keeping an image's original aspect ratio. The reasons why we apply resizing to those very large images are 1) in most cases 540×720 is large enough for text detection; and 2) a large size image may crash the entire training process because the required GPU memory is linearly dependent on the size of a training sample.

4.3. Evaluation Protocol and Decoding Settings

Both ICDAR 2015 Task 1.1 (born-digital images) and 2.1 (focused scene text) are evaluated through the online system of the ICDAR 2015 Robust Reading Competition, which uses the Wolf's object detection evaluation method [30]. Precision and recall are computed with respect to the best matches between one-to-one, one-to-many, and many-to-one matches. More details about this online evaluation system can be found in [12]. For the MSRA-TD500 dataset, we follow the evaluation protocol employed by [33], which considers both the area overlapping ratios and the orientation differences between hypothesis and reference.

It is worth noting that ICDAR and MSRA-TD500 datasets expect a hypothesis text box in the format of horizontal rectangular and oriented rectangular, respectively. We therefore parse a detected text bounding box from a predicted probability map by finding a smallest horizontal/oriented rectangular (see Fig. 7).

4.4. Results

Fig. 7 shows detection results of the proposed self-organized text detection method after finetuning our pretrained models on all three datasets. As one can see, the proposed self-organized text detector not only works well on those images with a near-homogeneous text background, but also those images with complicated background and foreground. Even for very difficult cases, e.g., images with a large perspective change (see CHINA POST in Fig. 7-(c)), and images with curved texts (see VALUT and Chocome in Fig. 7-(b)), and bing and COSTA COFFEE in Fig. 7-(c)), the proposed detector works reasonably well.

Table 3 compares the performance of the proposed self-organized text detector on the three testing datasets with the state-of-the-art methods, where * denotes methods relying on no extra external training data or pretrained weights. As one can see, the proposed self-organized text detector achieves better or comparable performance on all three testing datasets compared to recent peer algorithms. It is worth noting that all peer methods listed in the table have their own post-processing to group small regions, split a big region, or reject a text line etc., but we do not apply any post-processing except for finding the smallest rectangle for a given self-organized text region predicted on a probability map.

Table 3. Performance comparisons between the proposed approach and state-of-the-art methods.

Algorithm	Precision	Recall	F-Score
<i>ICDAR 2015 RRT 1.1 (born-digital images)</i>			
[35]	0.96	0.91	0.93
*[3]	0.92	0.86	0.89
[5]	0.95	0.91	0.93
*[39]	0.94	0.87	0.90
*[4]	0.92	0.89	0.90
Ours	0.91	0.95	0.93
<i>ICDAR 2015 RRT 2.1 (focused scene text)</i>			
[38]	0.84	0.65	0.73
*[40]	0.84	0.65	0.73
*[19]	0.82	0.71	0.76
*[1]	0.84	0.69	0.77
[41]	0.88	0.74	0.80
*[29]	0.84	0.77	0.80
*[24]	0.85	0.76	0.80
[5]	0.86	0.79	0.82
[42]	0.88	0.78	0.83
[34]	0.89	0.80	0.84
Ours	0.91	0.78	0.84
<i>MSRA-TD500</i>			
[33]	0.63	0.63	0.63
[11]	0.71	0.62	0.61
*[39]	0.71	0.61	0.66
[38]	0.81	0.63	0.71
[42]	0.83	0.67	0.74
[34]	0.76	0.75	0.76
Ours	0.77	0.78	0.77

4.5. Discussions

4.5.1 Imperfect Matching Metrics

We find that the ICDAR online evaluation metric [30] unfairly penalizes our system, primarily due to the fact that our system outputs line-level bounding boxes, and the ICDAR online evaluation metric uses word-level annotations. This results in poor scores for what are qualitatively very good outputs. Fig. 8 highlights some failure cases for the ICDAR evaluation metric. The middle row of images in each pane is our system's output probability map, and the upper and lower rows are the recall and precision visualizations taken directly from ICDAR's online evaluation tool. Colors mean different things in each image: for our output probability map, blue means a region without text, red means a region with text, and green means a border region. In the ICDAR metric visualizations, red means either false negative or false positive, green means true positive, and yellow and blue indicate true positive under the rule of many-to-one or one-to-many matching. A more detailed explanation of this online system can be found at <http://rrc.cvc.uab.es/>.

The main take away from this figure is that: in all cases our system correctly detects line boundaries, but is nonetheless penal-

(a)

(b)

(c)

Figure 7. Probability maps and decoded results of the proposed self-organized text detection on testing dataset. (a) ICDAR 2015 Robust Reading Competition Task 1.1 (born-digital images); (b) ICDAR 2015 Robust Reading Competition Task 2.1 (focused scene text); and (c) MSRA-TD500 dataset. Colors in a probability map represent **non-text**, **border**, and **text**, respectively.

ized by the evaluation metric. For instance, in Fig. 8-(a) our precision is hurt due to the large amount of space between some of the words; in Fig. 8-(b) we correctly determine the diagonal line, but because the evaluation metric expects three separate non-rotated rectangles for each word, both our precision and recall are penalized. The lower portion of Fig. 8 shows the original online evaluation scores and corrected scores after manual evaluation of these samples. As one can see, the F-score difference (far-right column) between the online evaluation and the manual evaluation is huge. **This indicates that our actual performance is even better than that shown in Table 3.**

Sample	Online Evaluation			Manual Evaluation			F-score Difference
	Precision	Recall	F-Score	Precision	Recall	F-Score	
(a)	0.33	0.67	0.44	1.00	1.00	1.00	0.56
(b)	0.50	0.67	0.57	1.00	1.00	1.00	0.43
(c)	0.43	0.52	0.47	1.00	1.00	1.00	0.53

Figure 8. Samples of underestimated performance. This underestimate is due to use of the online evaluation tool which unfairly penalizes word spacing. Details of underestimation for each sample are shown in the above figure.

Figure 9. Failure cases. (a) over-segmentation; (b) under-segmentation; and (c) incorrect semantic grouping.

4.5.2 Failure Cases

Although the proposed text detector works well on the majority of testing images, we find several failure cases. Fig. 9 shows the three types of failures that we found. It is worth noting that texts in Fig. 9-(c) should be vertically grouped according to word meanings. However, this kind of grouping is rarely seen in training samples.

4.5.3 Beyond the Border

The border class provides more information than just barriers to separate adjacent text candidates. Specifically, 1) border and text have a strong co-occurrence relationship, and this can be used to further improve the accuracy of text detection; and 2) border shape is closely related to the contour of a text region, and thus it can be used to rectify a text region. Though neither is used in this paper, they can be used to further improve text detection accuracy and are very important and useful for text recognition.

5. Conclusion

In this paper we analyze why previous efforts failed to distinguish adjacent text groups in the text detection problem, and we show that this issue can be solved by introducing the new border class. To fully validate this conceptual solution, we first created a new text detection dataset with the three-class annotations, *i.e.*, non-text, border, and text, and proposed a new lightweight multi-resolution FCN for text detection. We demonstrated that text detection models with the border class outperform those without the border class by a large margin (see Table 2). Our extensive experiments on external benchmarks further indicate that we can effectively detect text lines via border learning. This is a consummate improvement compared to all previous methods that require heavy post-processing to decide when to merge, split, or reject a detected text candidate. The proposed solution is applicable to other-levels of text detection, *e.g.* word-level, and also reveals the possibility of using the “border” class to distinguish different object instances in a more general image segmentation/localization problem.

References

- [1] M. Buta et al. Fasttext: Efficient unconstrained scene text detector. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1206–1214. IEEE, 2015. 1, 6
- [2] H. Chen, S. S. Tsai, G. Schroth, D. M. Chen, R. Grzeszczuk, and B. Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *2011 18th IEEE International Conference on Image Processing*, pages 2609–2612. IEEE, 2011. 1
- [3] K. Chen, F. Yin, A. Hussain, and C.-L. Liu. Efficient text localization in born-digital images by local contrast-based segmentation. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 291–295. IEEE, 2015. 1, 2, 6
- [4] K. Chen, F. Yin, and C.-L. Liu. Effective candidate component extraction for text localization in born-digital images by combining text contours and stroke interior regions. In *Document Analysis Systems (DAS), 2016 12th IAPR Workshop on*, pages 352–357. IEEE, 2016. 1, 2, 6
- [5] H. Cho, M. Sung, and B. Jun. Canny text detector: Fast and robust scene text localization algorithm. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 2, 6
- [6] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2963–2970. IEEE, 2010. 1

- [7] L. A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE transactions on pattern analysis and machine intelligence*, 10(6):910–918, 1988. **1**
- [8] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. **1**
- [9] T. He, W. Huang, Y. Qiao, and J. Yao. Accurate text localization in natural image with cascaded convolutional text network. *arXiv preprint arXiv:1603.09423*, 2016. **1, 2**
- [10] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20, 2016. **1**
- [11] L. Kang, Y. Li, and D. Doermann. Orientation robust text line detection in natural images. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4034–4041. IEEE, 2014. **1, 2, 6**
- [12] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. Icdar 2015 competition on robust reading. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1156–1160. IEEE, 2015. **1, 6**
- [13] 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 *2013 12th International Conference on Document Analysis and Recognition*, pages 1484–1493. IEEE, 2013. **1, 5**
- [14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. **2**
- [15] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. Textboxes: A fast text detector with a single deep neural network. In *AAAI*, pages 4161–4167, 2017. **1**
- [16] J. Mao, H. Li, W. Zhou, S. Yan, and Q. Tian. Scale based region growing for scene text detection. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM ’13, pages 1007–1016, New York, NY, USA, 2013. ACM. **1**
- [17] A. Mosleh, N. Bouguila, and A. B. Hamza. Image text detection using a bandlet-based edge detector and stroke width transform. In *BMVC*, pages 1–12, 2012. **1**
- [18] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3538–3545. IEEE, 2012. **1**
- [19] L. Neumann and J. Matas. Real-time lexicon-free scene text localization and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1872–1885, Sept 2016. **1, 2, 6**
- [20] W. Scherl, F. Wahl, and H. Fuchsberger. Automatic separation of text, graphic and picture segments in printed material. *Pattern Recognition in Practice*, pages 213–221, 1980. **1**
- [21] A. Shahab, F. Shafait, and A. Dengel. Icdar 2011 robust reading competition challenge 2: Reading text in scene images. In *2011 international conference on document analysis and recognition*, pages 1491–1496. IEEE, 2011. **1, 5**
- [22] C. Shi, C. Wang, B. Xiao, Y. Zhang, and S. Gao. Scene text detection using graph model built upon maximally stable extremal regions. *Pattern recognition letters*, 34(2):107–116, 2013. **1, 2**
- [23] F. Su and H. Xu. Robust seed-based stroke width transform for text detection in natural images. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 916–920. IEEE, 2015. **1**
- [24] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 4651–4659, 2015. **1, 2, 6**
- [25] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *European Conference on Computer Vision*, pages 56–72. Springer, 2016. **1, 2**
- [26] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. In *arXiv preprint arXiv:1601.07140*, 2016. **3**
- [27] F. M. Wahl, K. Y. Wong, and R. G. Casey. Block segmentation and text extraction in mixed text/image documents. *Computer graphics and image processing*, 20(4):375–390, 1982. **1**
- [28] X. Wang, Y. Song, Y. Zhang, and J. Xin. Natural scene text detection with multi-layer segmentation and higher order conditional random field based analysis. *Pattern Recogn. Lett.*, 60(C):41–47, Aug. 2015. **1, 2**
- [29] Y. Wei, Z. Zhang, W. Shen, D. Zeng, M. Fang, and S. Zhou. Text detection in scene images based on exhaustive segmentation. *Signal Processing: Image Communication*, 50:1–8, 2017. **1, 2, 6**
- [30] C. Wolf and J.-M. Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International Journal of Document Analysis and Recognition (IJDAR)*, 8(4):280–296, 2006. **4, 6**
- [31] I. Z. Yalniz, D. Gray, and R. Manmatha. Efficient exploration of text regions in natural scene images using adaptive image sampling. In *European Conference on Computer Vision*, pages 427–439. Springer, 2016. **1, 2**
- [32] C. Yao, X. Bai, and W. Liu. A unified framework for multi-oriented text detection and recognition. *IEEE Transactions on Image Processing*, 23(11):4737–4749, 2014. **3**
- [33] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1083–1090. IEEE, 2012. **1, 2, 3, 5, 6**
- [34] 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. **1, 2, 3, 6**
- [35] C. Yao, J. Wu, X. Zhou, C. Zhang, S. Zhou, Z. Cao, and Q. Yin. Incidental scene text understanding: Recent progresses on icdar 2015 robust reading competition challenge 4. *arXiv preprint arXiv:1511.09207*, 2015. **1, 2, 6**

- [36] C. Yao, X. Zhang, X. Bai, W. Liu, Y. Ma, and Z. Tu. Rotation-invariant features for multi-oriented text detection in natural images. *PloS one*, 8(8):e70173, 2013. **1**
- [37] Q. Ye and D. Doermann. Text detection and recognition in imagery: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 37(7):1480–1500, 2015. **1**
- [38] X.-C. Yin, W.-Y. Pei, J. Zhang, and H.-W. Hao. Multi-orientation scene text detection with adaptive clustering. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1930–1937, 2015. **1, 2, 6**
- [39] X.-C. Yin, X. Yin, K. Huang, and H.-W. Hao. Robust text detection in natural scene images. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):970–983, 2014. **1, 2, 6**
- [40] C. Yu, Y. Song, and Y. Zhang. Scene text localization using edge analysis and feature pool. *Neurocomputing*, 175:652–661, 2016. **1, 2, 6**
- [41] Z. Zhang, W. Shen, C. Yao, and X. Bai. Symmetry-based text line detection in natural scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2558–2567, 2015. **1, 2, 6**
- [42] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. **1, 2, 3, 4, 5, 6**
- [43] J. Zhou and D. Lopresti. Extracting text from www images. In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, volume 1, pages 248–252. IEEE, 1997. **1**