Cascaded Segmentation-Detection Networks for Word-Level Text Spotting

Siyang Qin and Roberto Manduchi
Computer Engineering Department
University of California Santa Cruz
California, United States
Email: {siqin, manduchi}@soe.ucsc.edu

***Abstract*—We introduce an algorithm for word-level text spotting that is able to accurately and reliably determine the bounding regions of individual words of text "in the wild". Our system is formed by the cascade of two convolutional neural networks. The first network is fully convolutional and is in charge of detecting areas containing text. This results in a very reliable but possibly inaccurate segmentation of the input image. The second network (inspired by the popular YOLO architecture) analyzes each segment produced in the first stage, and predicts oriented rectangular regions containing individual words. No post-processing (e.g. text line grouping) is necessary. With execution time of 450 ms for a 1000×560 image on a Titan X GPU, our system achieves good performance on the ICDAR 2013, 2015 benchmarks [2], [1].**

***Keywords*-scene text detection; convolutional neural network;**

I. Introduction

Fast automatic detection and reading of text (such a license plate number, a posted sign, or a street name) in images taken by a fixed or a moving camera, is very desirable for applications such as surveillance, forensics, autonomous vehicles, augmented reality (e.g., visual translation), and information access for blind people. Traditionally, OCR systems were designed for documents scanned into well-framed, good resolution images without excessive clutter, and taken under good illumination. Recent mobile OCR software implemented in smartphones (e.g. ABBYY TestGrabber or KNFBReader), dedicated hardware (OrCam), or in the cloud (Google Vision API) produces very good results, but none of these systems is designed for real-time deployment (multiple frames per second), which is critical for the applications mentioned above. For computational efficiency, a two-step process is often implemented. The first stage (text spotting) quickly detects the presence of areas in the image that are likely to contain text. These are then passed on to a recognition engine that decodes the textual content, using machine learning normally coupled with lexicon priors.

This contribution focuses on fast and accurate word-level text spotting. The ability to detect individual words may simplify the work of the recognizer, and word-level detection is part of typical benchmarks such as the ICDAR incidental and focused datasets [2], [1]. Individual word detection could be cast as an object detection task, for example using popular algorithms such as as Faster R-CNN [3] or YOLO [4], that can directly predict the coordinates of each object using axis-aligned rectangular bounding boxes. Unfortunately, direct application of these algorithms to general text-bearing images produces unsatisfactory results [5], [6]. This is because general object detection methods have difficulties at detecting groups of very small objects such as words in a text line.

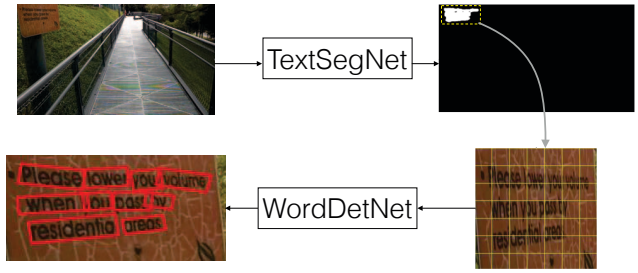


Figure 1. The general architecture of our word-level text spotting system. TextSegNet finds text blocks with arbitrary shapes and size. A squared resized block is passed on to WordDetNet, which generates oriented rectangular regions containing individual words.

Another possible approach to text spotting is the use of segmentation algorithms (such as the fully convolutional networks, or FCN [7]) to identify images areas that are likely to contain text. These algorithms have proved very effective in terms of detecting text at variable size, but are generally poor at identifying individual words [8], [9].

In this work, we combine FCN's remarkable robustness at segmenting text regions, with YOLO's efficient mechanism for detecting objects (in this case, words), appropriately modeled as oriented rectangles. The two systems are integrated as a cascade (see Fig. 1): text regions produced by our fully convolutional network (TextSegNet) are cropped out of the image and resized to a square shape with fixed size. Then, a YOLO-like network (WordDetNet) is trained to generate oriented rectangular bounding boxes around each word. A simple non-maximum suppression stage takes care of overlapping boxes. In analogy with foveated vision, TextSegNet takes the role of a "spotter", determining regions of interest to be analyzed in detail by WordDetNet. The resized text regions contain a limited density of words, matching the inherently limited capacity of WordDetNet.

2379-2140/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDAR.2017.210

The scheme is simple and elegant, and requires none of the post-processing steps (region grouping into straight lines, word splitting) that are typical of prior approaches. With execution time of 450 ms per image, our method achieves excellent results on popular benchmarks.

II. Prior Work

Early attempts at text spotting used hand-designed features to capture characteristics of text images, both statistical (bimodal marginal brightness distribution) and morphological (uniform stroke width, connectivity, consistent width and height, alignment into text rows). Two of the most successful examples were [10], based on MSER segmentation, and [11], based on the stroke width transform. While these techniques worked reasonably well, a substantial increase in detection and localization accuracy was achieved with the use of convolutional neural networks (CNN). The first such methods [12], [13], [14], [15] used specific techniques to extract regions (*proposals*) with good likelihood of containing text (or individual characters), which were then passed on to a CNN classifier that would rule out false detections. This strategy, however, was plagued by several drawbacks. Detecting individual characters is difficult in the presence of blur, noise, or poor contrast. Since the operators used for character detection were typically local, text-like background elements were often confused with text characters. In order to ensure good recall rate, many (possibly overlapping) templates must be processed by the CNN, resulting in long computational time.

Recent progress in semantic segmentation and object detection has offered new tools that are well suited to text spotting. Fully convolutional networks (FCN) [7], [16], [17], [18], [19] and end-to-end object detection architectures such as Faster R-CNN [3] and YOLO [4] process a full image, and produce pixel-wise labelling or labelled regions containing objects of interest. In particular, through the use of skip layers, FCN are able to analyze an image using both large and narrow receptive fields, effectively encoding both local features and global context. Unfortunately, the segmentation produced by FCN, while highly reliable, cannot in general separate individual text lines or words, and further processing is required (see Fig. 2).

The use of FCN for text spotting was pioneered by Zhang *et al.* [8], who trained an FCN model to predict a saliency map; text line hypotheses were formed by combining this saliency map with individual character templates found via MSER. A final character-level FCN was used to remove false detections. The work of Yao *et al.* [9] added supervision on the scale and center of characters as well as the linking orientation of nearby characters when training the text block FCN. With additional information produced by FCN, the task of the text line grouping module was much simplified. Individual words were found based on the detection of indents between characters in a text line.

In order to accurately locate text lines, Tian *et al.* [6] used a recurrent neural network to connect proposal regions into individual lines. This algorithm could be trained end-to-end; compared to other bottom-up methods, it required no dedicated post processing to form a text line. Unfortunately, this method only worked for near horizontal text lines, and was unable to identify individual words. Gupta *et al.* [5] introduced a method for individual word detection that did not require any post-processing (such as grouping individual character templates). They trained a fully convolutional regression network similar to [4] using a large dataset of synthetic images. This algorithm splits an image into cells in a grid, where each cell is responsible for detection of a word centered in it. Each image cell predicts the pose parameters (location, width, height, and orientation) of a word, along with a confidence value. This method gives good results on relative simple benchmarks [2], but suffers from its inherent inability to detect small words in the image, when several such words locate inside the same (fixed size) image cell. This limits its performance on the challenging ICDAR incidental dataset [1].

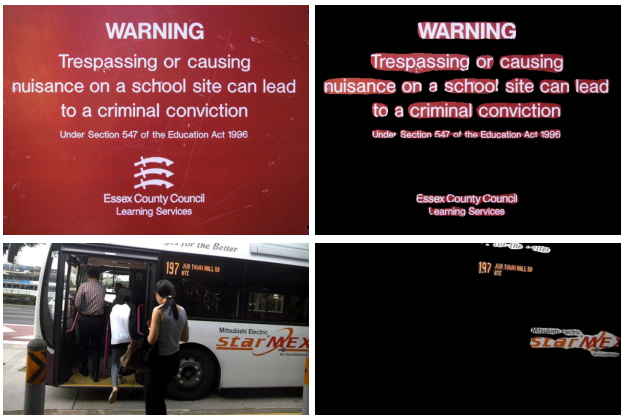
For more comprehensive surveys, the reader is referred to [20], [21], [22], [23].

III. Method

As discussed in Sec. II, the traditional bottom-up approach (from individual characters to text line grouping to individual words) has recently been replaced by top-down strategies, which start by detecting text regions (e.g. using FCN), then proceed to identifying individual words. Unfortunately, while FCN enables very robust pixel-level text block segmentation, detecting individual text lines or words with the same mechanism becomes more challenging. This can be intuitively justified as follows.

FCN is designed to produce a pixel-level classification, where the label assigned to each pixel comes from a multi-scale analysis of the pixel's neighborhood (*scale* here identifies the effective size of each node's receptive field). The ability to utilize both local and extended context allows FCN to produce high quality semantic segmentation. Text patterns, however, are a special category of "objects", characterized by a specific geometric structure: characters are spaced regularly along a mostly straight line, with words in a line separated by relatively small gaps. With large receptive fields, it is hard for FCN to reliably separate individual words, resulting in segments that may contain a group of text lines, an individual line, or even an individual word or character (see Fig. 2).

Direct application of object detection algorithms such as R-CNN or YOLO also generally produces poor results. R-CNN [24] relies on multiple region proposals, generated by methods such as selective search; this limits both performance and speed. Its successor, Faster R-CNN [3], replaces selective search with a learning-based algorithm for proposal



Input image FCN segmentation

Figure 2. FCN is in general unable to separate individual words.

Figure 3. By cascading segmentation (TextSegNet) with detection (WordDetNet), our algorithm can detect both large and small words. If only WordDetNet is used, as trained on whole images, detection will fail in the case of too small words. Detected words by TextSegNet + WordDetNet are shown by red rectangles, the result of WordDetNet only are shown by yellow rectangles.

generation. But due to the large variation in scale and aspect ratio that is typical of word regions, Faster R-CNN does not produce very good results, as reported in [6]. It should also be noted that most object detection algorithms predict axis-aligned rectangular regions, while words may have arbitrary orientation.

YOLO [4] formulates object detection as a regression problem. This algorithm is very fast and has shown good results for general object detection, but is by nature constrained in terms of its "capacity (the maximum number and density of regions produced in output). YOLO generates two boxes centered at each cell of a set defined on a regular grid. If more than two regions (e.g. words) are centered within the same cell, they cannot all be detected. The network capacity could be increased by changing the size of the cells or the number of boxes generated per cell. This, however, would result in increased computation and false positive rate.

Our architecture is a cascade of two stages (see Fig. 1): a segmentation stage (TextSegNet), that detects areas with high likelihood to contain text; and a word detection stage (WordDetNet), that, from the area cropped out by TextSegNet, resized to a common size, identifies and localizes individual words. The two stages are described in detail in the following.

A. TextSegNet

The first stage in our cascade, TextSegNet, is a fully convolutional network that takes both local and global context information into consideration to determine the label of each pixel. Text block detection is cast as a semantic segmentation problem with two labels ('text' and 'non-text').

1) Architecture: TextSegNet is based on the FCN-8s [7] model, which is derived from the VGG 16-layer network [25] with the final classifier layer removed, and the fully connected layers converted to convolution layers. We attach an additional final 1×1 convolution layer with channel dimension 2 to obtain prediction scores for 'text' and 'non-text'. Two skip layers are used to combine finer details (*pool 3* and *pool 4*) with high level semantic information; the output of the main and skip layers are combined and interpolated to the original image resolution using bilinear kernels (the weights of these kernels are also learnt during training). Softmax loss is used for training. The output of the network is a binary map representing likely areas containing text. For more details about the network structure of FCN-8s, the reader is referred to [7].

2) Training: During training and testing, all input images are resized so that their largest side has length of 1000 pixels. Both datasets considered in the experiments have ground-truth word-level labelling. Specifically, individual words are labelled by axis-aligned rectangular bounding boxes in the ICDAR 2013 focused dataset, and by generic quadrilaterals in the ICDAR 2015 incidental dataset. In order to train TextSegNet, a binary mask is first created, where all pixels in the word labelled regions are marked as 'text', while the other pixels are marked as 'non-text'.

B. WordDetNet

Each individual region identified by TextSegNet is first resized to a fixed square shape. More precisely, a square box tightly bounding and co-centered with the region is computed; the square box is then reshaped uniformly to a fixed size. In this way, the aspect ratio of the text image is not changed. The only exception is for segments that are very close to the image edges, where a co-centered square bounding box would extend outside the image area. In this case, a rectangular bounding box is considered, which is then re-sized as a square (see eg. Fig. 1). Reshaping text image segments to uniform size provides some degree of scale invariance, except for segments containing one or a few very long text lines, in which case the reshaped characters may have small size. The square reshaped images are then

fed to a network (WordDetNet) to predict the locations of individual words.

Inspired by the YOLO architecture [4] and the work of Gupta *et al.* [5], WordDetNet is tasked with identifying individual words. It defines a $N \times N$ grid on the image, where each cell in the grid predicts B candidate oriented rectangular regions (boxes), all centered within the cell. A box can be parameterized in terms of the position (x, y) of its center relative to the bounds of the grid cell, its width and height (w, h) relative to the size of image, and its orientation angle θ, which is normalized to a value between 0 and 1. In addition, the network produces a value (C) that represents the confidence that this box actually contains a word.

1) Loss Function: The network is trained to minimize a multi-part squared loss function $L(\mathbf{P})$. $\mathbf{P}$ represents the set of all $N^2 \cdot B$ box parameter vectors $\mathbf{p}_i^j = (x_i^j, y_i^j, w_i^j, h_i^j, \theta_i^j, C_i^j)$, where i identifies the cell and j identifies the box. $L(\mathbf{P})$ is defined as follows:

$$L(\mathbf{P}) = \sum_{i=1}^{N\times N} \delta_i^{obj} \sum_{j=1}^{B} \left[\delta_i^j L^{obj}(\mathbf{p}_i^j) + (1-\delta_i^j)(C_i^j)^2\right] + \lambda_{noobj} \sum_{i=1}^{N\times N} (1-\delta_i^{obj}) \sum_{j=1}^{B} (C_i^j)^2 \quad (1)$$

$$L^{obj}(\mathbf{p}_i^j) = \left(1 - C_i^j\right)^2 + \lambda_{ang} \left(\hat{\theta}_i - \theta_i^j\right)^2 + \lambda_{coord} \cdot \left[(\hat{x}_i - x_i^j)^2 + (\hat{y}_i - y_i^j)^2 + (\sqrt{\hat{w}_i} - \sqrt{w_i^j})^2 + (\sqrt{\hat{h}_i} - \sqrt{h_i^j})^2\right]$$

In the equation above, δ_i^{obj} is equal to 1 if the training image contains a word (represented by a oriented rectangle) centered at the i-th cell, 0 otherwise. δ_i^j is 1 only for the j-th predicted box with the largest Intersection-over-Union (IoU) with the word centered at the i-th cell (this is the *responsible* box [4]). The hatted notation represents "ground truth" values for the oriented rectangular word regions. In practice, for cells with no words centered on them, only the second term of the loss expression is activated, which penalizes predicted box confidence values larger than 0. Otherwise, only one responsible box is considered, with a penalty that takes into account both the box's confidence (which should be close to 1) and its localization accuracy; the non-responsible boxes are given a penalty for large confidence values. Note that Eq. (1) is equivalent to the loss function for the original YOLO network, except that (i) it contains an additional term for the rectangle orientation (θ), and (ii) there is no term assessing the posterior distribution of class assignment, as only one class (text) is considered here. We set the weights as follows: $\lambda_{ang} = 10$, $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.1$, which approximately balance multiple loss terms. The smaller value for λ_{noobj} is justified by the fact that only few cells are expected to be the center of words in the text region.

Predicted boxes with confidence value C_i^j less than 0.5 are discarded. Non-maximum suppression is used to remove overlapping detection with IoU less than 0.3. Note that general object detection algorithms use a larger threshold on the IoU. However, since words are normally placed along a line, a smaller overlap is expected in this case.

2) Architecture: WordDetNet (see Fig. 4) is built on the VGG 16-layer architecture [25], with the front layers initialized with the weights learned for TextSegNet (which also based on VGG 16-layer network). The original VGG 16-layer network is decapitated after *conv 5-3*, and two additional convolutional layers (*conv 6* and *prediction*) are added, along with a ReLU and dropout layer after *conv 6*. *conv 6* has 4096 filters with kernels size of 7×7, while *prediction* has $B \cdot 6$ filters with size of 3×3, resulting in B sets of box parameters ($\{\mathbf{p}_i^j\}$). Both *conv 6* and *prediction* are properly padded to maintain the size of the feature map. Note that that there are four 2×2 max pooling layers before *conv 5-3*. This means that, in order for the channels in output of *conv 5-3* to have size of $N \times N$ (corresponding to the cell grid described earlier), the input image must have size of $16 \cdot N \times 16 \cdot N$. For example, an input 240×240 color image will result in a grid of 15×15 cells. Note that the original YOLO network contains multiple fully connected layers, which are replaced by convolutional layers in WordDetNet. This results in a smaller number of parameters, allowing for easier training.

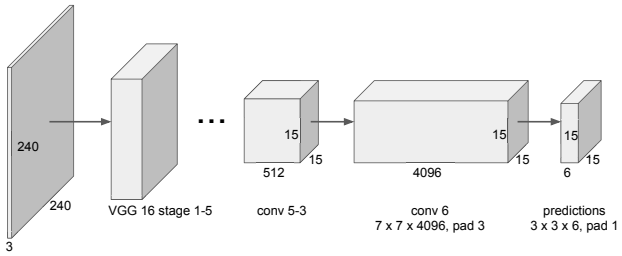


Figure 4. The structure of WordDetNet, shown here for N=15 and B=1. Our experiments show the above configuration achieves the best result.

3) Training: WordDetNet operates on square text block regions. During training, text blocks are generated by the following algorithm, which mimics the expected output of a typical segmenter, where nearby words are likely to be grouped within the same segment. A graph is formed on the ground truth word-level rectangular bounding boxes, where two such bounding boxes are linked in the graph if their minimum distance is smaller than a threshold (set to be equal to the sum of the heights of the two rectangles). Then, the connected components of this graph are found. For each connected component, the tightest axis-aligned bounding rectangle is computed. Then, as explained in Sec. III-B, this rectangle expanded to a square and resized, before being passed on to WordDetNet for training.

Figure 5. Examples of word-level text spotting on the ICDAR 2015 Incidental dataset. Detected word regions are shown by red boxes.

IV. Experiments

A. Implementation Details

Our training data come from three sources: the training portion of the ICDAR 2013 focused dataset (229 images) and of the ICDAR 2015 incidental dataset (1000 images), as well as the large scale synthetic dataset (8 million images) described in [5]. We augment the images from the ICDAR datasets by means of random rotations, translations, and color adjustment, and add a subset of 20K images randomly selected from the synthetic dataset [5]. Note that we use only a small portion of the synthetic dataset in order to maintain a balance between natural and synthetic images. Overall, our training dataset contains about 35K images. We found that the addition of synthetic images has a moderate effect on performance (F-score increase by 1% in the ICDAR incidental dataset only).

The training dataset for WordDetNet contains 40K text blocks which are automatically mined using the algorithm mentioned earlier in Sec. III-B3. Weight sharing between TextSegNet and WordDetNet enables good performance in spite of relatively small training data size.

Our implementation is based on Caffe [26], and runs on a workstation (3.3Ghz 6-score CPU, 32G RAM, GTX Titan X GPU and Ubuntu 14.04). As mentioned earlier, the images given in input to TextSegNet are resized to 1000 pixels in their longer side. We use the same training strategy as in [7], with batch size of 1, learning rate of 10^{-9}, momentum of 0.99, and weight decay of 0.0005. Training TextSegNet takes 20 hours for 100K iterations. WordDetNet takes text blocks resized to 240×240 pixels, resulting in a 15×15 grid. Training parameters are: batch size of 16, learning rate of 10^{-5}, momentum of 0.9 and weight decay of 0.0005. At 50K iterations, training of WordDetNet takes 10 hours.

At deployment, one image is processed by TextSegNet in about 250 ms, then each text block is processed by WordDetNet in about 50 ms. End-to-end processing takes about 450 ms per image on average.

B. ICDAR 2015 Incidental Dataset

1) Dataset Description: The ICDAR 2015 incidental dataset [1] contains 1000 training images and 500 images used for testing. These images were taken by wearable cameras, without intentional focus on text regions. They are characterized by large variance in text size and orientation; some amount of blur is often visible. This dataset thus represents a much more challenging benchmark than the older ICDAR 2013 Focused dataset, which is described later in Sec. IV-C. A quadrilateral bounding box is defined for each word in the dataset; however, only the bounding boxes for the training portion are made available to the public. Detection results are evaluated by measuring the Intersection-over-Union (IoU) between a predicted box and the closest ground truth quadrilateral; if the IoU is larger than 0.5, the predicted box is deemed a true positive. A score is produced in terms of precision (ratio of true positives count and all detections count) and recall (ratio of true positives count and all ground truth labels count), as well as of their harmonic mean (F-score). Note that some unreadable words are marked as "do not care"; they are still counted as ground truth labels when computing precision, but not when computing recall. Our algorithm is only trained on the words not labelled as "do not care".

2) Results: Tab. I shows results of our method as compared with other state of the art published algorithms [8], [6], [9], as well as with Google Vision API [1], which produces word-level detection and recognition. Our cascaded (segmentation + detection) network outperforms the previous best results by a large margin (7% higher F-score than its closest competitor). In order to highlight the importance

Table I
RESULTS ON THE ICDAR 2015 INCIDENTAL DATASET

Method	Precision (%)	Recall (%)	F-score (%)
HUST [20]	44	38	41
AJON [20]	47	47	47
NJU-Text [20]	70	36	47
StradVision [20]	53	46	50
Zhang [8]	71	43	54
Google Vision API [1]	68	53	59
CTPN [6]	74	52	61
Megvii-image++ [9]	72	58	64
Proposed (Seg+Det)	**79**	**65**	**71**
Proposed (Det only)	61	40	48

of the prior segmentation step, we also show results using only WordDetNet as applied on the whole image, rather than on the segments detected by TextSegNet. Performance decreases substantially in this case, showing the importance of a prior segmentation step. Some detection examples in challenging images are shown in Fig. 5.

The network capacities (defined as the maximum density of candidate boxes generated) can be changed by varying the number of cells in the $N \times N$ grid defined in WordDetNet, or the number B of boxes generated in each cell. Fig. 6 plots the F-score resulting from varying N between 7 and 19 (only odd values [4]) and with B equal to 1 and 2. This data shows that generating more than one candidate box per cell doesn't seem to provide an advantage, and that the optimal number of cells is 15×15. Note that with fewer cells, the burden is on the network to correctly localize the box within a larger cell. With more (hence smaller) cells, correct localization is easier, but the risk of false positives increases.

In spite of the good quantitative results, we noticed that sometime the box orientation and/or size as estimated by WordDetNet is somewhat inaccurate (see examples in Fig. 8). Incorrect orientation or size of an estimated word region may reduce the Intersection-over-Union with the corresponding ground truth region, thus affecting the resulting recall rate.

C. ICDAR 2013 Focused Dataset

1) Dataset Description: The ICDAR 2013 focused dataset [2] contains images "explicitly focused around the text content of interest". 229 images are used for training and 233 for testing. In these images, text is seen at good resolution and at approximately horizontal orientation. Each word is labelled with an axis-aligned rectangle. The evaluation protocol is described in [21], [34].

2) Results: Tab. II shows comparative results against other published algorithms. Our system has F-score of 86%, ranking among the top performers. The best current result is achieved by CTPN [6], with F-score of 88%. Note, however, that CTPN cannot identify individual words, and does not work for text with arbitrary orientation. In the more challenging ICDAR 2015 incidental dataset, our system outperforms CTPN by 10%. Looking closer at the data, one may notice that our algorithm produces the same recall value (83%) as the best performing systems, but lags behind in precision (90%, vs. 93% as obtained with CTPN [6] and Zhu *et al.* [33]). Part of the reason is that our method is able to find existing text that doesn't appear in the ground-truth labelling (see e.g. Fig. 7), resulting in an (incorrect) penalty in terms of precision.

[1] https://cloud.google.com/vision/

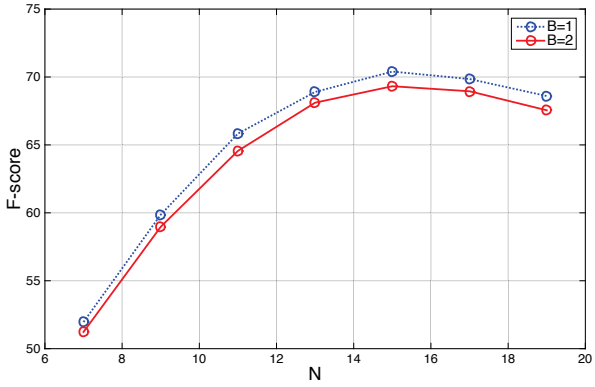


Figure 6. The F-score of ours system on the ICDAR 2015 incidental dataset as a function of the parameters N (which determines the number of cells in grid considered by WordDetNet) and B (the number of boxes generated per cell).

Table II
RESULTS ON THE ICDAR 2013 FOCUSED DATASET

Method	Precision (%)	Recall (%)	F-score (%)
Neumann *et al.* [27]	85	68	75
Yin *et al.* [28]	86	68	76
FASText [29]	84	69	77
Huang *et al.* [13]	88	71	78
Zhang *et al.* [14]	88	74	80
TextFlow [30]	85	76	80
He *et al.* [31]	93	73	82
Qin *et al.* [15]	88	77	82
Zhang *et al.* [8]	88	78	83
Gupta *et al.* [5]	92	76	83
Yao *et al.* [9]	88	80	84
TextBoxes [32]	88	**83**	85
Zhu *et al.* [33]	**93**	81	87
CTPN [6]	**93**	**83**	**88**
Proposed	90	**83**	86

V. CONCLUSION

We have presented a new approach for word-level text detection and localization. Our algorithm identifies individual words and draws bounding boxes in the shape of oriented rectangles. The system is formed by the cascaded of a segmentation network (TextSegNet) and a detection network (WordDetNet), where the latter operates on regions segmented out by the former, resized to a common size.

Figure 7. Our method finds words in very challenging situations (yellow boxes), even when these words are not labelled in the ICDAR 2013 focused dataset.

Figure 8. Examples of failure cases: incorrect box orientation or size, missing words in curved text.

By combining segmentation and detection, we leverage on the strengths of each network. TextSegNet (which is based on the fully convolutional architecture of [7]) can very robustly detect the presence of text in the image, but is unable to identify individual words. WordDetNet (inspired by the YOLO architecture [4]) can effectively detect and localize individual words from a resized regions identified by TextSegNet. Unlike most existing algorithms, our system does not require a post-processing step to enforce alignment of the detected words.

We show experimentally that the first step (segmentation and resizing) is critical for the second step to be effective. On the challenging ICDAR 2015 incidental dataset, our system achieves top results among the published algorithms, outperforming the closest one by 7% in terms of F-score. In the more benign ICDAR 2013 focused dataset, our system produces an F-score that is only 2% less than the top performing algorithm (CTPN [6]).

REFERENCES

[1] "Icdar incidental scene text dataset," http://rrc.cvc.uab.es/?ch=4&com=introduction.

[2] "Icdar focused scene text dataset," http://rrc.cvc.uab.es/?ch=2.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[5] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2315–2324.

[6] 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*. Springer, 2016, pp. 56–72.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[8] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai, "Multi-oriented text detection with fully convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4159–4167.

[9] C. Yao, X. Bai, N. Sang, X. Zhou, S. Zhou, and Z. Cao, "Scene text detection via holistic, multi-channel prediction," *arXiv preprint arXiv:1606.09002*, 2016.

[10] L. Neumann and J. Matas, "Real-time scene text localization and recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 3538–3545.

[11] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 2963–2970.

[12] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *21st International Conference on Pattern Recognition (ICPR)*. IEEE, 2012, pp. 3304–3308.

[13] W. Huang, Y. Qiao, and X. Tang, "Robust scene text detection with convolution neural network induced mser trees," in *ECCV 2014*. Springer, 2014, pp. 497–511.

[14] 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*, 2015, pp. 2558–2567.

[15] S. Qin and R. Manduchi, "A fast and robust text spotter," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2016.

[16] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.

[17] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv preprint arXiv:1606.00915*, 2016.

[18] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1395–1403.

[19] S. Qin, S. Kim, and R. Manduchi, "Automatic skin and hair masking using fully convolutional networks," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2017.

[20] 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*. IEEE, 2015, pp. 1156–1160.

[21] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. Gomez i Bigorda, S. Robles Mestre, J. Mas, D. Fernandez Mota, J. Almazan Almazan, and L.-P. de las Heras, "Icdar 2013 robust reading competition," in *12th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2013, pp. 1484–1493.

[22] A. Shahab, F. Shafait, and A. Dengel, "Icdar 2011 robust reading competition challenge 2: Reading text in scene images," in *11th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2011, pp. 1491–1496.

[23] Q. Ye and D. Doermann, "Text detection and recognition in imagery: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 1480–1500, 2015.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[26] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[27] L. Neumann and J. Matas, "On combining multiple segmentations in scene text recognition," in *12th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2013, pp. 523–527.

[28] 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*, vol. 36, no. 5, pp. 970–983, 2014.

[29] M. Busta, L. Neumann, and J. Matas, "Fastext: Efficient unconstrained scene text detector," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1206–1214.

[30] 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*, 2015, pp. 4651–4659.

[31] T. He, W. Huang, Y. Qiao, and J. Yao, "Text-attentional convolutional neural network for scene text detection," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2529–2541, 2016.

[32] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "Textboxes: A fast text detector with a single deep neural network," *arXiv preprint arXiv:1611.06779*, 2016.

[33] S. Zhu and R. Zanibbi, "A text detection system for natural scenes with convolutional feature learning and cascaded classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 625–632.

[34] 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)*, vol. 8, no. 4, pp. 280–296, 2006.