EAST: An Efficient and Accurate Scene Text Detector

Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang

Megvii Technology Inc., Beijing, China
{zxy, yaocong, wenhe, wangyuzhi, zsc, hwr, liangjiajun}@megvii.com

Abstract

Previous approaches for scene text detection have already achieved promising performances across various benchmarks. However, they usually fall short when dealing with challenging scenarios, even when equipped with deep neural network models, because the overall performance is determined by the interplay of multiple stages and components in the pipelines. In this work, we propose a simple yet powerful pipeline that yields fast and accurate text detection in natural scenes. The pipeline directly predicts words or text lines of arbitrary orientations and quadrilateral shapes in full images, eliminating unnecessary intermediate steps (e.g., candidate aggregation and word partitioning), with a single neural network. The simplicity of our pipeline allows concentrating efforts on designing loss functions and neural network architecture. Experiments on standard datasets including ICDAR 2015, COCO-Text and MSRA-TD500 demonstrate that the proposed algorithm significantly outperforms state-of-the-art methods in terms of both accuracy and efficiency. On the ICDAR 2015 dataset, the proposed algorithm achieves an F-score of 0.7820 at 13.2fps at 720p resolution.

1. Introduction

Recently, extracting and understanding textual information embodied in natural scenes have become increasingly important and popular, which is evidenced by the unprecedented large numbers of participants of the ICDAR series contests [30, 16, 15] and the launch of the TRAIT 2016 evaluation by NIST [1].

Text detection, as a prerequisite of the subsequent processes, plays a critical role in the whole procedure of textual information extraction and understanding. Previous text detection approaches [2, 33, 12, 7, 48] have already obtained promising performances on various benchmarks in this field. The core of text detection is the design of features to distinguish text from backgrounds. Traditionally,
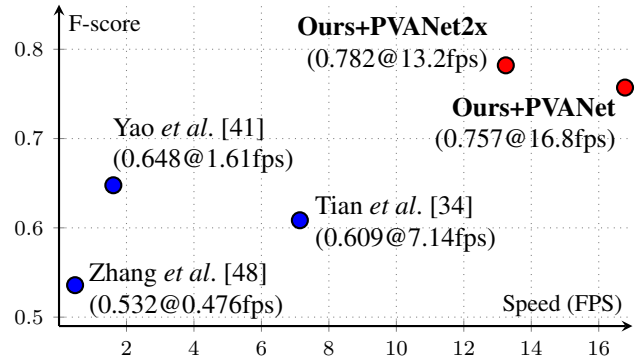

Figure 1. Performance versus speed on ICDAR 2015 [15] text localization challenge. As can be seen, our algorithm significantly surpasses competitors in accuracy, whilst running very fast. The specifications of hardware used are listed in Tab. 6.

features are manually designed [5, 25, 40, 10, 26, 45] to capture the properties of scene text, while in deep learning based methods [3, 13, 11, 12, 7, 48] effective features are directly learned from training data.

However, existing methods, either conventional or deep neural network based, mostly consist of several stages and components, which are probably sub-optimal and time-consuming. Therefore, the accuracy and efficiency of such methods are still far from satisfactory.

In this paper, we propose a fast and accurate scene text detection pipeline that has only two stages. The pipeline utilizes a fully convolutional network (FCN) model that directly produces word or text-line level predictions, excluding redundant and slow intermediate steps. The produced text predictions, which can be either rotated rectangles or quadrangles, are sent to Non-Maximum Suppression to yield final results. Compared with existing methods, the proposed algorithm achieves significantly enhanced performance, while running much faster, according to the qualitative and quantitative experiments on standard benchmarks.

Specifically, the proposed algorithm achieves an F-score of 0.7820 on ICDAR 2015 [15] (0.8072 when tested in multi-scale), 0.7608 on MSRA-TD500 [40] and 0.3945 on
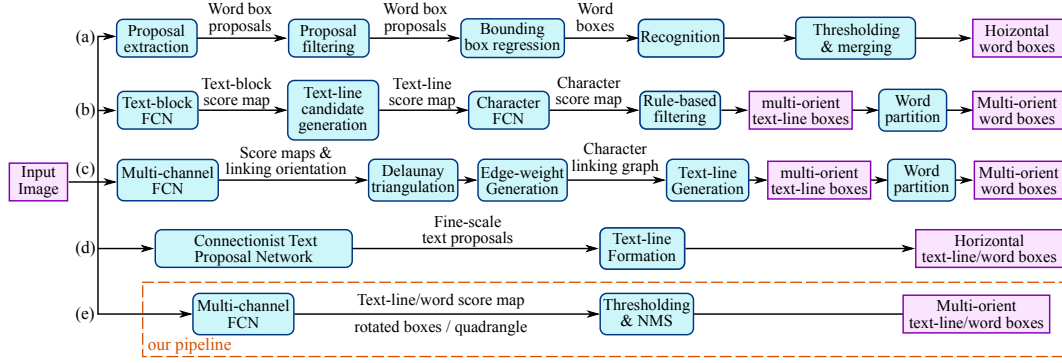
Figure 2. Comparison of pipelines of several recent works on scene text detection: (a) Horizontal word detection and recognition pipeline proposed by Jaderberg *et al.* [12]; (b) Multi-orient text detection pipeline proposed by Zhang *et al.* [48]; (c) Multi-orient text detection pipeline proposed by Yao *et al.* [41]; (d) Horizontal text detection using CTPN, proposed by Tian *et al.* [34]; (e) Our pipeline, which eliminates most intermediate steps, consists of only two stages and is much simpler than previous solutions.

COCO-Text [36], outperforming previous state-of-the-art algorithms in performance while taking much less time on average (13.2fps at 720p resolution on a Titan-X GPU for our best performing model, 16.8fps for our fastest model).

The contributions of this work are three-fold:

- We propose a scene text detection method that consists of two stages: a Fully Convolutional Network and an NMS merging stage. The FCN directly produces text regions, excluding redundant and time-consuming intermediate steps.
- The pipeline is flexible to produce either word level or line level predictions, whose geometric shapes can be rotated boxes or quadrangles, depending on specific applications.
- The proposed algorithm significantly outperforms state-of-the-art methods in both accuracy and speed.

2. Related Work

Scene text detection and recognition have been active research topics in computer vision for a long period of time. Numerous inspiring ideas and effective approaches [5, 25, 26, 24, 27, 37, 11, 12, 7, 41, 42, 31] have been investigated. Comprehensive reviews and detailed analyses can be found in survey papers [50, 35, 43]. This section will focus on works that are mostly relevant to the proposed algorithm.

Conventional approaches rely on manually designed features. Stroke Width Transform (SWT) [5] and Maximally Stable Extremal Regions (MSER) [25, 26] based methods generally seek character candidates via edge detection or extremal region extraction. Zhang *et al.* [47] made use of the local symmetry property of text and designed various features for text region detection. FASText [2] is a fast text detection system that adapted and modified the well-known FAST key point detector for stroke extraction. However, these methods fall behind of those based on deep neural networks, in terms of both accuracy and adaptability, especially when dealing with challenging scenarios, such as

low resolution and geometric distortion.

Recently, the area of scene text detection has entered a new era that deep neural network based algorithms [11, 13, 48, 7] have gradually become the mainstream. Huang *et al.* [11] first found candidates using MSER and then employed a deep convolutional network as a strong classifier to prune false positives. The method of Jaderberg *et al.* [13] scanned the image in a sliding-window fashion and produced a dense heatmap for each scale with a convolutional neural network model. Later, Jaderberg *et al.* [12] employed both a CNN and an ACF to hunt word candidates and further refined them using regression. Tian *et al.* [34] developed vertical anchors and constructed a CNN-RNN joint model to detect horizontal text lines. Different from these methods, Zhang *et al.* [48] proposed to utilize FCN [23] for heatmap generation and to use component projection for orientation estimation. These methods obtained excellent performance on standard benchmarks. However, as illustrated in Fig. 2(a-d), they mostly consist of multiple stages and components, such as false positive removal by post filtering, candidate aggregation, line formation and word partition. The multitude of stages and components may require exhaustive tuning, leading to sub-optimal performance, and add to processing time of the whole pipeline.

In this paper, we devise a deep FCN-based pipeline that directly targets the final goal of text detection: word or text-line level detection. As depicted in Fig. 2(e), the model abandons unnecessary intermediate components and steps, and allows for end-to-end training and optimization. The resultant system, equipped with a single, light-weighted neural network, surpasses all previous methods by an obvious margin in both performance and speed.

3. Methodology

The key component of the proposed algorithm is a neural network model, which is trained to directly predict the existence of text instances and their geometries from full
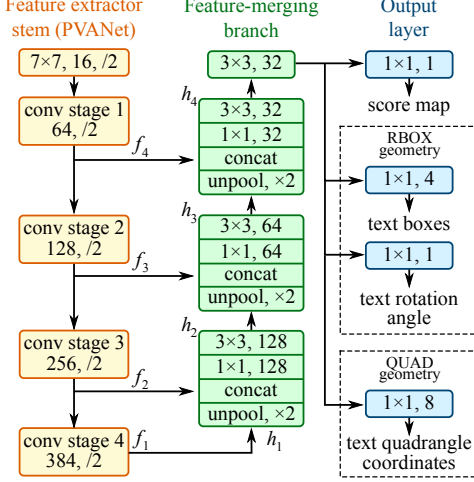
Figure 3. Structure of our text detection FCN.

images. The model is a fully-convolutional neural network adapted for text detection that outputs dense per-pixel predictions of words or text lines. This eliminates intermediate steps such as candidate proposal, text region formation and word partition. The post-processing steps only include thresholding and NMS on predicted geometric shapes. The detector is named as **EAST**, since it is an **E**fficient and **A**ccuracy **S**cene **T**ext detection pipeline.

3.1. Pipeline

A high-level overview of our pipeline is illustrated in Fig. 2(e). The algorithm follows the general design of DenseBox [9], in which an image is fed into the FCN and multiple channels of pixel-level text score map and geometry are generated.

One of the predicted channels is a score map whose pixel values are in the range of $[0, 1]$. The remaining channels represent geometries that encloses the word from the view of each pixel. The score stands for the confidence of the geometry shape predicted at the same location.

We have experimented with two geometry shapes for text regions, rotated box (RBOX) and quadrangle (QUAD), and designed different loss functions for each geometry. Thresholding is then applied to each predicted region, where the geometries whose scores are over the predefined threshold is considered valid and saved for later non-maximum-suppression. Results after NMS are considered the final output of the pipeline.

3.2. Network Design

Several factors must be taken into account when designing neural networks for text detection. Since the sizes of word regions, as shown in Fig. 5, vary tremendously, determining the existence of large words would require features from late-stage of a neural network, while predicting accurate geometry enclosing a small word regions need low-level information in early stages. Therefore the network

must use features from different levels to fulfill these requirements. HyperNet [19] meets these conditions on features maps, but merging a large number of channels on large feature maps would significantly increase the computation overhead for later stages.

In remedy of this, we adopt the idea from *U-shape* [29] to merge feature maps gradually, while keeping the upsampling branches small. Together we end up with a network that can both utilize different levels of features and keep a small computation cost.

A schematic view of our model is depicted in Fig. 3. The model can be decomposed in to three parts: feature extractor *stem*, feature-merging *branch* and output layer.

The *stem* can be a convolutional network pre-trained on ImageNet [4] dataset, with interleaving convolution and pooling layers. Four levels of feature maps, denoted as f_i, are extracted from the stem, whose sizes are $\frac{1}{32}$, $\frac{1}{16}$, $\frac{1}{8}$ and $\frac{1}{4}$ of the input image, respectively. In Fig. 3, PVANet [17] is depicted. In our experiments, we also adopted the well-known VGG16 [32] model, where feature maps after pooling-2 to pooling-5 are extracted.

In the feature-merging branch, we gradually merge them:

$$g_i = \begin{cases} \text{unpool}(h_i) & \text{if} \quad i \leq 3 \\ \text{conv}_{3\times3}(h_i) & \text{if} \quad i = 4 \end{cases} \tag{1}$$

$$h_i = \begin{cases} f_i & \text{if} \quad i = 1 \\ \text{conv}_{3\times3}(\text{conv}_{1\times1}([g_{i-1}; f_i])) & \text{otherwise} \end{cases} \tag{2}$$

where g_i is the merge base, and h_i is the merged feature map, and the operator $[\cdot; \cdot]$ represents concatenation along the channel axis. In each merging stage, the feature map from the last stage is first fed to an unpooling layer to double its size, and then concatenated with the current feature map. Next, a $\text{conv}_{1\times1}$ bottleneck [8] cuts down the number of channels and reduces computation, followed by a $\text{conv}_{3\times3}$ that fuses the information to finally produce the output of this merging stage. Following the last merging stage, a $\text{conv}_{3\times3}$ layer produces the final feature map of the merging branch and feed it to the output layer.

The number of output channels for each convolution is shown in Fig. 3. We keep the number of channels for convolutions in *branch* small, which adds only a fraction of computation overhead over the *stem*, making the network computation-efficient. The final output layer contains several $\text{conv}_{1\times1}$ operations to project 32 channels of feature maps into 1 channel of score map F_s and a multi-channel geometry map F_g. The geometry output can be either one of RBOX or QUAD, summarized in Tab. 1

For RBOX, the geometry is represented by 4 channels of axis-aligned bounding box (AABB) $\mathbf{R}$ and 1 channel rotation angle θ. The formulation of $\mathbf{R}$ is the same as that in [9], where the 4 channels represents 4 distances from the

Geometry	channels	description
AABB	4	$\mathbf{G} = \mathbf{R} = \{d_i | i \in \{1,2,3,4\}\}$
RBOX	5	$\mathbf{G} = \{\mathbf{R}, \theta\}$
QUAD	8	$\mathbf{G} = \mathbf{Q} = \{(\Delta x_i, \Delta y_i) | i \in \{1,2,3,4\}\}$

Table 1. Output geometry design

(a)   (b)

box edge
distances

line angle
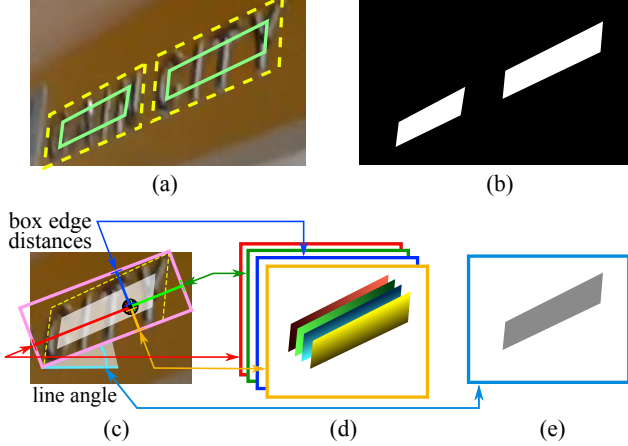
(c)   (d)   (e)

Figure 4. Label generation process: (a) Text quadrangle (yellow dashed) and the shrunk quadrangle (green solid); (b) Text score map; (c) RBOX geometry map generation; (d) 4 channels of distances of each pixel to rectangle boundaries; (e) Rotation angle.

pixel location to the top, right, bottom, left boundaries of the rectangle respectively.

For QUAD $\mathbf{Q}$, we use 8 numbers to denote the coordinate shift from four corner vertices $\{p_i | i \in \{1,2,3,4\}\}$ of the quadrangle to the pixel location. As each distance offset contains two numbers $(\Delta x_i, \Delta y_i)$, the geometry output contains 8 channels.

3.3. Label Generation

3.3.1 Score Map Generation for Quadrangle

Without loss of generality, we only consider the case where the geometry is a quadrangle. The positive area of the quadrangle on the score map is designed to be roughly a shrunk version of the original one, illustrated in Fig. 4 (a).

For a quadrangle $\mathbf{Q} = \{p_i | i \in \{1,2,3,4\}\}$, where $p_i = \{x_i, y_i\}$ are vertices on the quadrangle in clockwise order. To shrink $\mathbf{Q}$, we first compute a *reference length* r_i for each vertex p_i as

$$
\begin{aligned}
r_i = \min(&\mathrm{D}(p_i, p_{(i \bmod 4)+1}), \\
&\mathrm{D}(p_i, p_{((i+2) \bmod 4)+1}))
\end{aligned}
\tag{3}
$$

where $\mathrm{D}(p_i, p_j)$ is the L_2 distance between p_i and p_j.

We first shrink the two longer edges of a quadrangle, and then the two shorter ones. For each pair of two opposing edges, we determine the "longer" pair by comparing the mean of their lengths. For each edge $\langle p_i, p_{(i \bmod 4)+1} \rangle$, we shrink it by moving its two endpoints inward along the edge by $0.3r_i$ and $0.3r_{(i \bmod 4)+1}$ respectively.

3.3.2 Geometry Map Generation

As discussed in Sec. 3.2, the geometry map is either one of RBOX or QUAD. The generation process for RBOX is illustrated in Fig. 4 (c-e).

For those datasets whose text regions are annotated in QUAD style (*e.g.*, ICDAR 2015), we first generate a rotated rectangle that covers the region with minimal area. Then for each pixel which has positive score, we calculate its distances to the 4 boundaries of the text box, and put them to the 4 channels of RBOX ground truth. For the QUAD ground truth, the value of each pixel with positive score in the 8-channel geometry map is its coordinate shift from the 4 vertices of the quadrangle.

3.4. Loss Functions

The loss can be formulated as

$$
L = L_s + \lambda_g L_g
\tag{4}
$$

where L_s and L_g represents the losses for the score map and the geometry, respectively, and λ_g weighs the importance between two losses. In our experiment, we set λ_g to 1.

3.4.1 Loss for Score Map

In most state-of-the-art detection pipelines, training images are carefully processed by balanced sampling and hard negative mining to tackle with the imbalanced distribution of target objects [9, 28]. Doing so would potentially improve the network performance. However, using such techniques inevitably introduces a non-differentiable stage and more parameters to tune and a more complicated pipeline, which contradicts our design principle.

To facilitate a simpler training procedure, we use class-balanced cross-entropy introduced in [38], given by

$$
\begin{aligned}
L_s &= \text{balanced-xent}(\hat{\mathbf{Y}}, \mathbf{Y}^*) \\
&= -\beta \mathbf{Y}^* \log \hat{\mathbf{Y}} - (1-\beta)(1-\mathbf{Y}^*) \log(1-\hat{\mathbf{Y}})
\end{aligned}
\tag{5}
$$

where $\hat{\mathbf{Y}} = F_s$ is the prediction of the score map, and $\mathbf{Y}^*$ is the ground truth. The parameter β is the balancing factor between positive and negative samples, given by

$$
\beta = 1 - \frac{\sum_{y^* \in \mathbf{Y}^*} y^*}{|\mathbf{Y}^*|}.
\tag{6}
$$

This balanced cross-entropy loss is first adopted in text detection by Yao *et al.* [41] as the objective function for score map prediction. We find it works well in practice.

3.4.2 Loss for Geometries

One challenge for text detection is that the sizes of text in natural scene images vary tremendously. Directly using L1

or L2 loss for regression would guide the loss bias towards larger and longer text regions. As we need to generate accurate text geometry prediction for both large and small text regions, the regression loss should be scale-invariant. Therefore, we adopt the IoU loss in the AABB part of RBOX regression, and a scale-normalized smoothed-L1 loss for QUAD regression.

RBOX For the AABB part, we adopt IoU loss in [46], since it is invariant against objects of different scales.

$$L_{\text{AABB}} = -\log \text{IoU}(\hat{\mathbf{R}}, \mathbf{R}^*) = -\log \frac{|\hat{\mathbf{R}} \cap \mathbf{R}^*|}{|\hat{\mathbf{R}} \cup \mathbf{R}^*|} \quad (7)$$

where $\hat{\mathbf{R}}$ represents the predicted AABB geometry and $\mathbf{R}^*$ is its corresponding ground truth. It is easy to see that the width and height of the intersected rectangle $|\hat{\mathbf{R}} \cap \mathbf{R}^*|$ are

$$\begin{aligned} w_{\mathbf{i}} &= \min(\hat{d}_2, d_2^*) + \min(\hat{d}_4, d_4^*) \\ h_{\mathbf{i}} &= \min(\hat{d}_1, d_1^*) + \min(\hat{d}_3, d_3^*) \end{aligned} \quad (8)$$

where d_1, d_2, d_3 and d_4 represents the distance from a pixel to the top, right, bottom and left boundary of its corresponding rectangle, respectively. The union area is given by

$$|\hat{\mathbf{R}} \cup \mathbf{R}^*| = |\hat{\mathbf{R}}| + |\mathbf{R}^*| - |\hat{\mathbf{R}} \cap \mathbf{R}^*|. \quad (9)$$

Therefore, both the intersection/union area can be computed easily. Next, the loss of rotation angle is computed as

$$L_\theta(\hat{\theta}, \theta^*) = 1 - \cos(\hat{\theta} - \theta^*). \quad (10)$$

where $\hat{\theta}$ is the prediction to the rotation angle and θ^* represents the ground truth. Finally, the overall geometry loss is the weighted sum of AABB loss and angle loss, given by

$$L_{\text{g}} = L_{\text{AABB}} + \lambda_\theta L_\theta. \quad (11)$$

Where λ_θ is set to 10 in our experiments.

Note that we compute L_{AABB} regardless of rotation angle. This can be seen as an approximation of quadrangle IoU when the angle is perfectly predicted. Although it is not the case during training, it could still impose the correct gradient for the network to learn to predict $\hat{\mathbf{R}}$.

QUAD We extend the smoothed-L1 loss proposed in [6] by adding an extra normalization term designed for word quadrangles, which is typically longer in one direction. Let all coordinate values of $\mathbf{Q}$ be an ordered set

$$C_{\mathbf{Q}} = \{x_1, y_1, x_2, y_2, \ldots, x_4, y_4\} \quad (12)$$

then the loss can be written as

$$\begin{aligned} L_{\text{g}} &= L_{\text{QUAD}}(\hat{\mathbf{Q}}, \mathbf{Q}^*) \\ &= \min_{\tilde{\mathbf{Q}} \in P_{\mathbf{Q}^*}} \sum_{\substack{c_i \in C_{\mathbf{Q}}, \\ \tilde{c}_i \in C_{\tilde{\mathbf{Q}}}}} \frac{\text{smoothed}_{L1}(c_i - \tilde{c}_i)}{8 \times N_{\mathbf{Q}^*}} \end{aligned} \quad (13)$$

where the normalization term $N_{\mathbf{Q}^*}$ is the shorted edge length of the quadrangle, given by

$$N_{\mathbf{Q}^*} = \min_{i=1}^{4} D(p_i, p_{(i \bmod 4)+1}), \quad (14)$$

and $P_{\mathbf{Q}}$ is the set of all equivalent quadrangles of $\mathbf{Q}^*$ with different vertices ordering. This ordering permutation is required since the annotations of quadrangles in the public training datasets are inconsistent.

3.5. Training

The network is trained end-to-end using ADAM [18] optimizer. To speed up learning, we uniformly sample 512x512 crops from images to form a minibatch of size 24. Learning rate of ADAM starts from 1e-3, decays to one-tenth every 27300 minibatches, and stops at 1e-5. The network is trained until performance stops improving.

3.6. Locality-Aware NMS

To form the final results, the geometries survived after thresholding should be merged by NMS. A naïve NMS algorithm runs in $O(n^2)$ where n is the number of candidate geometries, which is unacceptable as we are facing tens of thousands of geometries from dense predictions.

Under the assumption that the geometries from nearby pixels tend to be highly correlated, we proposed to merge the geometries row by row, and while merging geometries in the same row, we will iteratively merge the geometry currently encountered with the last merged one. This improved technique runs in $O(n)$ in best scenarios[1]. Even though its worst case is the same as the naïve one, as long as the locality assumption holds, the algorithm runs sufficiently fast in practice. The procedure is summarized in Algorithm 1

It is worth mentioning that, in WEIGHTEDMERGE(g, p), the coordinates of merged quadrangle are weight-averaged by the scores of two given quadrangles. To be specific, if $a = \text{WEIGHTEDMERGE}(g, p)$, then $a_i = V(g)g_i + V(p)p_i$ and $V(a) = V(g) + V(p)$, where a_i is one of the coordinates of a subscripted by i, and $V(a)$ is the score of geometry a.

In fact, there is a subtle difference that we are "averaging" rather than "selecting" geometries, as in a standard NMS procedure will do, acting as a voting mechanism, which in turn introduces a stabilization effect when feeding videos. Nonetheless, we still adopt the word "NMS" for functional description.

4. Experiments

To compare the proposed algorithm with existing methods, we conducted qualitative and quantitative experiments on three public benchmarks: ICDAR2015, COCO-Text and MSRA-TD500.

---

[1] Consider the case that only a single text line appears the image. In such case, all geometries will be highly overlapped if the network is sufficiently powerful

Algorithm 1 Locality-Aware NMS

```
1:  function NMSLOCALITY(geometries)
2:      S ← ∅, p ← ∅
3:      for g ∈ geometries in row first order do
4:          if p ≠ ∅ ∧ SHOULDMERGE(g, p) then
5:              p ← WEIGHTEDMERGE(g, p)
6:          else
7:              if p ≠ ∅ then
8:                  S ← S ∪ {p}
9:              end if
10:             p ← g
11:         end if
12:     end for
13:     if p ≠ ∅ then
14:         S ← S ∪ {p}
15:     end if
16:     return STANDARDNMS(S)
17: end function
```

Network	Description
PVANET [17]	small and fast model
PVANET2x [17]	PVANET with 2x number of channels
VGG16 [32]	commonly used model

Table 2. Base Models

for all the benchmarks, it may suffer from either over-fitting or under-fitting. We experimented with three different base networks, with different output geometries, on all the datasets to evaluate the proposed framework. These networks are summarized in Tab. 2.

VGG16 [32] is widely used as base network in many tasks [28, 38] to support subsequent task-specific fine-tuning, including text detection [34, 48, 49, 7]. There are two drawbacks of this network: (1). The receptive field for this network is small. Each pixel in output of conv5_3 only has a receptive field of 196. (2). It is a rather large network.

PVANET is a light weight network introduced in [17], aiming as a substitution of the feature extractor in Faster-RCNN [28] framework. Since it is too small for GPU to fully utilizes computation parallelism, we also adopt *PVANET2x* that doubles the channels of the original PVANET, exploiting more computation parallelism while running slightly slower than PVANET. This is detailed in Sec. 4.5. The receptive field of the output of the last convolution layer is 809, which is much larger than VGG16.

The models are pre-trained on the ImageNet dataset [21].

4.3. Qualitative Results

Fig. 5 depicts several detection examples by the proposed algorithm. It is able to handle various challenging scenarios, such as non-uniform illumination, low resolution, varying orientation and perspective distortion. Moreover, due to the voting mechanism in the NMS procedure, the proposed method shows a high level of stability on videos with various forms of text instances[2].

The intermediate results of the proposed method are illustrated in Fig. 6. As can be seen, the trained model produces highly accurate geometry maps and score map, in which detections of text instances in varying orientations are easily formed.

4.4. Quantitative Results

As shown in Tab. 3 and Tab. 4, our approach outperforms previous state-of-the-art methods by a large margin on ICDAR 2015 and COCO-Text.

In ICDAR 2015 Challenge 4, when images are fed at their original scale, the proposed method achieves an F-score of 0.7820. When tested at multiple scales [3] using the

4.1. Benchmark Datasets

ICDAR 2015 is used in Challenge 4 of ICDAR 2015 Robust Reading Competition [15]. It includes a total of 1500 pictures, 1000 of which are used for training and the remaining are for testing. The text regions are annotated by 4 vertices of the quadrangle, corresponding to the QUAD geometry in this paper. We also generate RBOX output by fitting a rotated rectangle which has the minimum area. These images are taken by Google Glass in an incidental way. Therefore text in the scene can be in arbitrary orientations, or suffer from motion blur and low resolution. We also used the 229 training images from ICDAR 2013.

COCO-Text [36] is the largest text detection dataset to date. It reuses the images from MS-COCO dataset [22]. A total of 63,686 images are annotated, in which 43,686 are chosen to be the training set and the rest 20,000 for testing. Word regions are annotated in the form of axis-aligned bounding box (AABB), which is a special case of RBOX. For this dataset, we set angle θ to zero. We use the same data processing and test method as in ICDAR 2015.

MSRA-TD500 [40] is a dataset comprises of 300 training images and 200 test images. Text regions are of arbitrary orientations and annotated at sentence level. Different from the other datasets, it contains text in both English and Chinese. The text regions are annotated in RBOX format. Since the number of training images is too few to learn a deep model, we also harness 400 images from HUST-TR400 dataset [39] as training data.

4.2. Base Networks

As except for COCO-Text, all text detection datasets are relatively small compared to the datasets for general object detection[21, 22], therefore if a single network is adopted

---

[2]Online video: `https://youtu.be/o5asMTdhmvA`. Note that each frame in the video is processed independently.

[3]At relative scales of 0.5, 0.7, 1.0, 1.4, and 2.0.

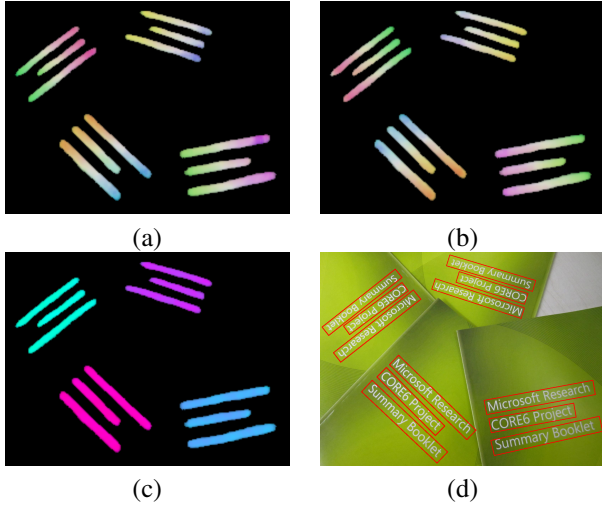Figure 5. Qualitative results of the proposed algorithm. (a) ICDAR 2015. (b) MSRA-TD500. (c) COCO-Text.

Figure 6. Intermediate results of the proposed algorithm. (a) Estimated geometry map for d_1 and d_4. (b) Estimated geometry map for d_2 and d_3. (c) Estimated angle map for text instances. (d) Predicted rotated rectangles of text instances. Maps in (a), (b) and (c) are color-coded to represent variance (for d_1, d_2, d_3 and d_4) and invariance (for angle) in an pixel-wise manner. Note that in the geometry maps only the values of foreground pixels are valid.

same network, our method reaches 0.8072 in F-score, which is nearly 0.16 higher than the best method [41] in terms of absolute value (0.8072 vs. 0.6477).

Comparing the results using VGG16 network[34, 48, 41], the proposed method also outperforms best previous work [41] by 0.0924 when using QUAD output, 0.116 when using RBOX output. Meanwhile these networks are quite efficient, as will be shown in Sec.4.5.

In COCO-Text, all of the three settings of the proposed algorithm result in higher accuracy than previous top performer [41]. Specifically, the improvement over [41] in F-

score is 0.0614 while that in recall is 0.053, which confirm the advantage of the proposed algorithm, considering that COCO-Text is the largest and most challenging benchmark to date. Note that we also included the results from [36] as reference, but these results are actually not valid baselines, since the methods (A, B and C) are used in data annotation.

The improvements of the proposed algorithm over previous methods prove that a simple text detection pipeline, which directly targets the final goal and eliminating redundant processes, can beat elaborated pipelines, even those integrated with large neural network models.

As shown in Tab. 5, on MSRA-TD500 all of the three settings of our method achieve excellent results. The F-score of the best performer (Ours+PVANET2x) is slightly higher than that of [41]. Compared with the method of Zhang *et al*. [48], the previous published state-of-the-art system, the best performer (Ours+PVANET2x) obtains an improvement of 0.0208 in F-score and 0.0428 in precision.

Note that on MSRA-TD500 our algorithm equipped with VGG16 performs much poorer than that with PVANET and PVANET2x (0.7023 vs. 0.7445 and 0.7608), the main reason is that the effective receptive field of VGG16 is smaller than that of PVANET and PVANET2x, while the evaluation protocol of MSRA-TD500 requires text detection algorithms output line level instead of word level predictions.

In addition, we also evaluated Ours+PVANET2x on the ICDAR 2013 benchmark. It achieves 0.8267, 0.9264 and 0.8737 in recall, precision and F-score, which are comparable with the previous state-of-the-art method [34], which obtains 0.8298, 0.9298 and 0.8769 in recall, precision and F-score, respectively.

4.5. Speed Comparison

The overall speed comparison is demonstrated in Tab. 6. The numbers we reported are averages from running

Algorithm	Recall	Precision	F-score
Ours + PVANET2x RBOX MS*	**0.7833**	0.8327	**0.8072**
Ours + PVANET2x RBOX	0.7347	**0.8357**	**0.7820**
Ours + PVANET2x QUAD	0.7419	0.8018	0.7707
Ours + VGG16 RBOX	0.7275	0.8046	0.7641
Ours + PVANET RBOX	0.7135	0.8063	0.7571
Ours + PVANET QUAD	0.6856	0.8119	0.7434
Ours + VGG16 QUAD	0.6895	0.7987	0.7401
Yao et al. [41]	0.5869	0.7226	0.6477
Tian et al. [34]	0.5156	0.7422	0.6085
Zhang et al. [48]	0.4309	0.7081	0.5358
StradVision2 [15]	0.3674	0.7746	0.4984
StradVision1 [15]	0.4627	0.5339	0.4957
NJU [15]	0.3625	0.7044	0.4787
AJOU [20]	0.4694	0.4726	0.4710
Deep2Text-MO [45, 44]	0.3211	0.4959	0.3898
CNN MSER [15]	0.3442	0.3471	0.3457

Table 3. Results on ICDAR 2015 Challenge 4 Incidental Scene Text Localization task. MS means multi-scale testing.

Algorithm	Recall	Precision	F-score
Ours + VGG16	**0.324**	**0.5039**	**0.3945**
Ours + PVANET2x	0.340	0.406	0.3701
Ours + PVANET	0.302	0.3981	0.3424
Yao et al. [41]	0.271	0.4323	0.3331
Baselines from [36]			
A	0.233	0.8378	0.3648
B	0.107	0.8973	0.1914
C	0.047	0.1856	0.0747

Table 4. Results on COCO-Text.

Algorithm	Recall	Precision	F-score
Ours + PVANET2x	0.6743	**0.8728**	**0.7608**
Ours + PVANET	0.6713	0.8356	0.7445
Ours + VGG16	0.6160	0.8167	0.7023
Yao et al. [41]	**0.7531**	0.7651	0.7591
Zhang et al. [48]	0.67	0.83	0.74
Yin et al. [44]	0.63	0.81	0.71
Kang et al. [14]	0.62	0.71	0.66
Yin et al. [45]	0.61	0.71	0.66
TD-Mixture [40]	0.63	0.63	0.60
TD-ICDAR [40]	0.52	0.53	0.50
Epshtein et al. [5]	0.25	0.25	0.25

Table 5. Results on MSRA-TD500.

through 500 test images from the ICDAR 2015 dataset at their original resolution (1280x720) using our best performing networks. These experiments were conducted on a server using a single NVIDIA Titan X graphic card with Maxwell architecture and an Intel E5-2670 v3 @ 2.30GHz CPU. For the proposed method, the post-processing includes thresholding and NMS, while others should refer to

Approach	Res.	Device	T_1/T_2 (ms)	FPS
Ours + PVANET	720p	Titan X	58.1 / 1.5	16.8
Ours + PVANET2x	720p	Titan X	73.8 / 1.7	13.2
Ours + VGG16	720p	Titan X	150.9 / 2.4	6.52
Yao et al. [41]	480p	K40m	420 / 200	1.61
Tian et al. [34]	ss-600*	GPU	130 / 10	7.14
Zhang et al. [48]*	MS*	Titan X	2100 / N/A	0.476

Table 6. Overall time consumption compared on different methods. T_1 is the network prediction time, and T_2 accounts for the time used on post-processing. For Tian et al. [34], ss-600 means short side is 600, and 130ms includes two networks. Note that they reach their best result on ICDAR 2015 using a short edge of 2000, which is much larger than ours. For Zhang et al. [48], MS means they used 200, 500, 1000 three scales, and the result is obtained on MSRA-TD500. The theoretical flops per pixel for our three models are 18KOps, 44.4KOps and 331.6KOps respectively, for PVANET, PVANET2x and VGG16.

their original paper.

While the proposed method significantly outperforms state-of-the-art methods, the computation cost is kept very low, attributing to the simple and efficient pipeline. As can be observed from Tab. 6, the fastest setting of our method runs at a speed of 16.8 FPS, while slowest setting runs at 6.52 FPS. Even the best performing model Ours+PVANET2x runs at a speed of 13.2 FPS. This confirm that our method is among the most efficient text detectors that achieve state-of-the-art performance on benchmarks.

4.6. Limitations

The maximal size of text instances the detector can handle is proportional to the receptive field of the network. This limits the capability of the network to predict even longer text regions like text lines running across the images.

Also, the algorithm might miss or give imprecise predictions for vertical text instances as they take only a small portion of text regions in the ICDAR 2015 training set.

5. Conclusion and Future Work

We have presented a scene text detector that directly produces word or line level predictions from full images with a single neural network. By incorporating proper loss functions, the detector can predict either rotated rectangles or quadrangles for text regions, depending on specific applications. The experiments on standard benchmarks confirm that the proposed algorithm substantially outperforms previous methods in terms of both accuracy and efficiency.

Possible directions for future research include: (1) adapting the geometry formulation to allow direct detection of curved text; (2) integrating the detector with a text recognizer; (3) extending the idea to general object detection.

References

[1] Text Recognition Algorithm Independent Evaluation (TRAIT). http://www.nist.gov/itl/iad/ig/trait-2016.cfm. Accessed: 2015-11-1.

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

[3] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu, and A. Y. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Proc. of ICDAR*, 2011.

[4] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. of CVPR*, 2009.

[5] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *Proc. of CVPR*, 2010.

[6] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[7] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. *arXiv preprint arXiv:1604.06646*, 2016.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[9] L. Huang, Y. Yang, Y. Deng, and Y. Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015.

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

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

[12] 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, jan 2016.

[13] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *Proc. of ECCV*, 2014.

[14] L. Kang, Y. Li, and D. Doermann. Orientation robust text line detection in natural images. In *Proc. of CVPR*, 2014.

[15] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. 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. of ICDAR*, 2015.

[16] 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 *Proc. of ICDAR*, 2013.

[17] K.-H. Kim, S. Hong, B. Roh, Y. Cheon, and M. Park. PVANET: Deep but lightweight neural networks for real-time object detection. *arXiv preprint arXiv:1608.08021*, 2016.

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. *arXiv preprint arXiv:1604.00600*, 2016.

[20] H. Koo and D. H. Kim. Scene text detection via connected component clustering and nontext filtering. *IEEE Trans. on Image Processing*, 22(6):2296–2305, 2013.

[21] 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.

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

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

[24] A. Mishra, K. Alahari, and C. V. Jawahar. Top-down and bottom-up cues for scene text recognition. In *Proc. of CVPR*, 2012.

[25] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Proc. of ACCV*, 2010.

[26] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *Proc. of CVPR*, 2012.

[27] T. Novikova, , O. Barinova, P. Kohli, and V. Lempitsky. Large-lexicon attribute-consistent text recognition in natural images. In *Proc. of ECCV*, 2012.

[28] 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*, pages 91–99, 2015.

[29] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[30] A. Shahab, F. Shafait, and A. Dengel. ICDAR 2011 robust reading competition challenge 2: Reading text in scene images. In *Proc. of ICDAR*, 2011.

[31] 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 Trans. Pattern Analysis and Machine Intelligence*, 2016.

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

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

[34] 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.

[35] S. Uchida. Text localization and recognition in images and video. *Handbook of Document Image Processing and Recognition*, pages 843–883, 2014.

[36] 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*, 2016.

[37] J. J. Weinman, Z. Butler, D. Knoll, and J. Feild. Toward integrated scene text reading. *IEEE Trans. on PAMI*, 36(2):375–387, 2013.

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

[39] 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.

[40] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *Proc. of CVPR*, 2012.

[41] 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.

[42] C. Yao, X. Bai, B. Shi, and W. Liu. Strokelets: A learned multi-scale representation for scene text recognition. In *Proc. of CVPR*, 2014.

[43] Q. Ye and D. Doermann. Text detection and recognition in imagery: A survey. *IEEE Trans. PAMI*, 37(7):1480–1500, 2014.

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

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

[46] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang. Unitbox: An advanced object detection network. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 516–520. ACM, 2016.

[47] Z. Zhang, W. Shen, C. Yao, and X. Bai. Symmetry-based text line detection in natural scenes. In *Proc. of CVPR*, 2015.

[48] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. In *Proc. of CVPR*, 2015.

[49] Z. Zhong, L. Jin, S. Zhang, and Z. Feng. Deeptext: A unified framework for text proposal generation and text detection in natural images. *arXiv preprint arXiv:1605.07314*, 2016.

[50] Y. Zhu, C. Yao, and X. Bai. Scene text detection and recognition: Recent advances and future trends. *Frontiers of Computer Science*, 2016.