Learning Shape-Aware Embedding for Scene Text Detection

Zhuotao Tian[†], Michelle Shu[‡], Pengyuan Lyu[§], Ruiyu Li[§],
Chao Zhou[§], Xiaoyong Shen[§], Jiaya Jia[†,§]

[†]The Chinese University of Hong Kong, [‡]Johns Hopkins University, [§]YouTu Lab, Tencent

{zttian,leojia}@cse.cuhk.edu.hk, mshu1@jhu.edu, {pengyuanlv,royryli,brycezhou,dylanshen}@tencent.com

Abstract

We address the problem of detecting scene text in arbitrary shapes, which is a challenging task due to the high variety and complexity of the scene. We treat text detection as instance segmentation and propose a segmentation-based framework, which extracts each text instance as an independent connected component. To distinguish among different text instances, our method maps pixels onto an embedding space where pixels belonging to the same text are encouraged to appear closer to each other and vise versa. In addition, we introduce a shape-aware Loss to make training adaptively accommodate various aspect ratios of text instances and even the tiny gaps among them. A new post-processing pipeline yields precise bounding box prediction. Experimental results on three challenging datasets (ICDAR15 [20], MSRA-TD500 [55] and CTW1500 [32]) demonstrate the effectiveness of our work.

1. Introduction

As an indispensable part of Optical Character Recognition (OCR) systems, scene text detection is essential to the subsequent text recognition. High performing scene text detection, as a fundamental tool, benefits a wide spectrum of applications, including multilingual translation from images, human machine interaction, and environment understanding. This task, however, is challenging due to the varying attributes in natural images, such as the degree of image blur, lighting condition, and aspect ratios.

Because of the complicated text shapes and aspect ratios, although existing regression-based methods [30, 53, 60, 49, 15] have achieved impressive results on benchmark data annotated by rectangular or quadrilateral bounding boxes, they hardly generalize to curved text data in CTW1500 [32] or TotalText [3] where text can be in arbitrary shape. Recently, Long *et al*. [34] handled curved text by modeling text instances as a sequence of disks with different radius. The method relies on radius regression that may result in drop of precision.

So far, predicting text boxes at various scales and aspect ratios is challenging for dense regression based methods such as EAST [60], since the regression distance is confined to training crop sizes, making the model hard to predict long (or large) instances during inference as shown in Table 4 where dilating center area (marked with 'No Regression') predicted by EAST even outperforms regression. Therefore we aim at a regression-free solution to circumvent the drawback.

(a) (b) (c)

Figure 1. Given an input image (a), our model learns to map pixels in text regions (b) into an embedding space (c) where pixels belonging to the same instance are pulled together and pixels from different instances are pushed away from each other.

Existing regression-free solutions in text detection community are mainly based on segmentation frameworks, where the algorithms generate foreground masks before recovering corresponding text instances. Specifically, Yao *et al*. [56] and Zhang *et al*. [58] linked candidates to form text instances. But they fail to separate close text due to the lack of ability to deal with tiny intervals that are very common in natural scenes. Small space is also easily overlooked after down-sampling in Convolutional Neural Networks (CNNs). Wu *et al*. [51] predicted text masks alongside text borders, and the model depends on boundaries to isolate individual text regions. Nevertheless, long text instances often have thin borders that may lead to inaccurate results if the text borders are not correctly revealed.

In this work, we draw inspiration from [1] and propose an alternative segmentation-based method to mitigate the issues mentioned above. As shown in Figure 1, our model considers each text instance as a cluster and learns to map pixels onto an embedding space where pixels belonging to the same text instance are encouraged to appear close.

By constraining embedding feature of pixels inside the same text region to share similar properties, our model is capable of learning intrinsic representation, *i.e*. the embedding feature, to separate instances rather than simply relying on intervals and unclear boundaries. Moreover, to further

improve the robustness against tiny intervals and various shapes, we introduce a Shape-Aware Loss that can adaptively adjust pulling and pushing force on the embedding feature based on scales and adjacency of text instances.

Finally, our novel adjustment pipeline produces high-quality bounding boxes, as it effectively utilizes information from both embedding space and segmentation space generated by two parallel branches from our proposed network. We conduct experiments on three challenging datasets. Our results demonstrate the superiority of our new design. Our contributions are threefold.

- We propose a Shape-Aware Loss to ease separating adjacent instances and detecting large instances.
- We propose a new text detection pipeline that detects text instances of arbitrary shape.
- Our approach achieves competitive performance on three representative scene text datasets.

2. Related Work

2.1. Scene Text Detection

Scene text detection has long been a popular research topic with many solutions proposed [2, 60, 30, 6, 32, 26, 15, 50, 36, 33, 52, 34, 53, 49, 35, 18, 17, 14, 42]. Early text detectors [8, 40, 39, 37, 57] used hand-crafted features based on characteristics of text, such as Stroke Width Transform (STW) [8], Maximally Stable Extremal Regions (MSER) [37] and symmetry feature [57]. Recently, several deep neural network based methods were proposed, which lead to more accurate text detection. These methods can be divided into two categories, i.e., regression-based and segmentation-based methods.

Regression-based methods generate text boxes via predicting the bounding box offsets from anchors or pixels. In [25, 59, 11], following SSD [29], faster R-CNN [44] and YOLO [43], text boxes are detected directly. In [46] and [49], to detect long text effectively, Shi *et al*. and Tian *et al*. proposed SegLink and CTPN, which predict text segments and then link these segments to text boxes. To handle detection of long and oriented text, Lyu *et al*. [36] obtained corner points of text, and group them into boxes. Different from these methods that regress candidate boxes/segments/corners from anchors, Zhou *et al*. [60], He *et al*. [17] and Long *et al*. [34] performed box regression by predicting offsets from pixels in text region. Although regression-based methods have achieved state-of-the-art performance, regressing text boxes at various scales and aspect ratios is still challenging.

Segmentation-based methods infer candidate text boxes from segmentation maps. Compared to regression-based methods, they detect text of arbitrary shapes more easily, and yet struggle with overlapping predicted text regions. To split text regions from each other, in [58, 56], character localization and text orientation are used. In [7], link relationship between a pixel and its neighboring pixels are predicted in order to group pixels that belong to the same instance. Wu *et al*. [51] introduced a border class and obtained the text region directly separated by text borders.

Different from existing regression-free methods that utilize link relationship or border classes, our method uses embedding features to provide instance information, and achieves decent performance.

2.2. Instance Segmentation

There are several instance segmentation methods [4, 5, 24, 12, 28, 10, 1, 22]. Among them, proposal-free methods relate to our work the most. In [10], pixels are grouped based on seediness (the measure if it is a good seed for segmentation) using similarity measure between their embeddings. In [1], discriminative loss is presented to concentrate pixels inside the same instance and separate those from different instances. A recurrent grouping model is introduced in [22] to map pixel embeddings onto a n-sphere space.

Our method is different from the classic proposal-free instance segmentation, as text instances are special cases of objects, and their characteristics are usually very different from common objects. Therefore, to better capture characteristics of text instances, we propose effective Shape-Aware Loss (SA Loss) to deal with this difference. Moreover, with embedding feature trained on the SA Loss, our new cluster processing method generates proposals of text of arbitrary shape.

3. Our Method

3.1. Network Structure

Our method is a segmentation-based framework, which generates prediction for text instances of arbitrary shape via embedding clustering. Given an input image, our network first produces embedding features and text foreground masks, which are subsequently processed to obtain final predicted text boxes.

The overall structure of our network is shown in Figure 2. It has a mirror symmetry of FPN [27]. First, We extract features from intermediate layers of ResNet50[13]. Next, in each feature merging module we use the similar feature merging strategy as the adaptive feature pooling of PANet [28] to combine extracted features from different layers by upsampling and pixel-wise addition. Pooling is not involved and it requires keeping the same number of channels.

Different from other multi-task networks designed with a single module [60, 30, 53], we use two separate feature merging modules to form a pair of independent but complementary branches. One branch produces the embedding map with 8-channel embedding feature at each pixel for distinguishing among text instances, while the other is designed to generate two text foreground masks for segmentation. By disentangling weight sharing, our single-stage network allows these two quite different tasks to benefit from each other. Analysis and experiments related to the dual-branch design are given in Section 4.4.3.

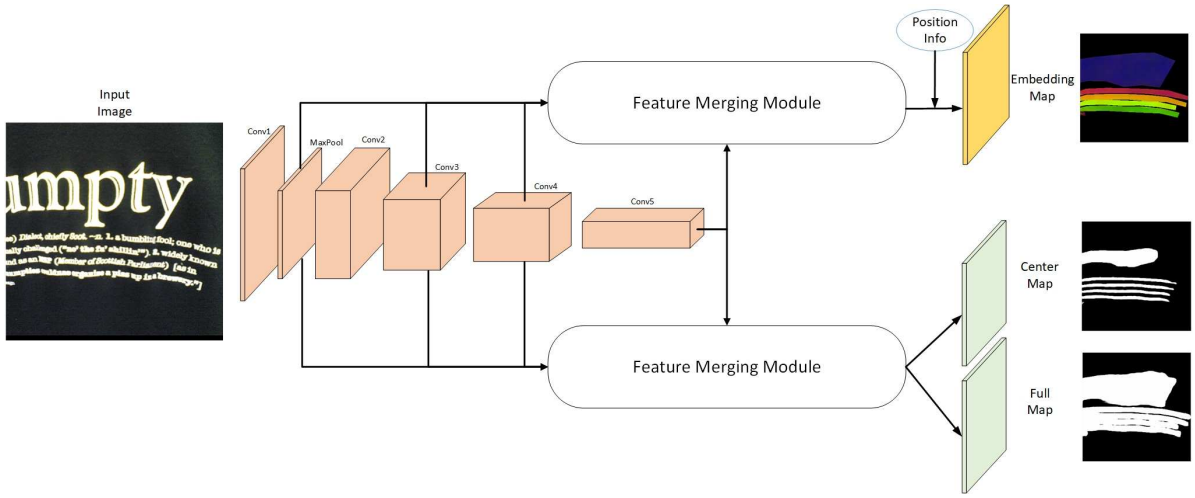


Figure 2. Overall architecture of our network designed for scene text segmentation.

3.1.1 Shape-Aware Embedding

Motivation. Scene text instances are different from normal objects since text strokes often blend into the background. Although overlapping may occur between two object instances, it does not happen so often. General boundaries between two normal object instances are clear, hence easier to determine than those of text instances. What make things worse is that aspect ratios of text instances may vary largely from a tiny word to a very long sentence over the entire image, which make it more difficult to detect text instances. To overcome these difficulties, we propose learning Shape-Aware Embedding for text instances that accommodate various aspect ratios and imprecise boundaries.

Design. The embedding branch receives features from one feature merging module, and an additional 2-channel position information represented by x and y coordinates [41]. We concatenate features from the feature merging module with the position information, and pass them through three consecutive 3×3 convolutional layers with 32, 16, and 8 output channels respectively. The final output is a 8-channel embedding feature for each pixel.

Loss Function. Given a set of text instances and the embedding features for pixels within each text region, we propose a Shape-Aware Loss (SA Loss) that comprises a variance loss L_{var} to gather embedding of pixels from the same text instance and a distance loss L_{dist} to push embedding of pixels of different instances apart. They are expressed as

$$
\begin{aligned}
L_{var}(I_j) &= \frac{1}{N_j} \sum_{i=1}^{N_j} \max\left(W_{Scale_{(j)}} * |\mu_j - x_i| - \eta, 0\right), \\
L_{dist}(I_j, I_k) &= \max\left(\gamma - W_{Dist_{(j,k)}} * |\mu_j - \mu_k|, 0\right), \quad (1)
\end{aligned}
$$

where μ_j and μ_k are the average embedding of text instances I_j and I_k respectively. x_i is the embedding feature of pixel i, and N_j is the number of pixels within I_j. η and γ represent margins for variance loss and distance loss, and we set them to 0.5 and 1.5 respectively.

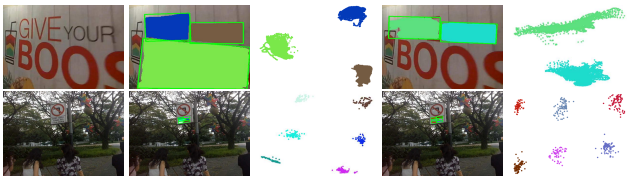
Different from [1], we include two balance weights $W_{Scale_{(j)}}$ and $W_{Dist_{(j,k)}}$ to accommodate various text shapes and adjacency where

$$
\begin{aligned}
W_{Scale_{(j)}} &= e^{\frac{\max_{side(j)}}{2\max(h,w)}}, \\
W_{Dist_{(j,k)}} &= (1 - 20e^{-(4+\frac{\min(Distance_{j,k})}{\max(h,w)} * 10)}). \quad (2)
\end{aligned}
$$

In Eq. (2), $\max(h, w)$ is the longer edge of an input image. $\max_{side(j)}$ for quadrangle text is the length of the longer edge. For curved text (with polygon annotation), it is the longest distance between vertices of a polygon. $\min(Distance_{j,k})$ is the shortest distance between text instances I_j and I_k. To avoid dominance by one scaled loss, we set these two weights empirically. The value range of $W_{Scale_{(j)}}$ is $(1, 1.65)$, and the value range of $W_{Dist_{(j,k)}}$ is roughly $(0.63, 1)$ whose scaling ratio is comparable to $W_{Scale_{(j)}}$ to balance their effect to gradients.

To make SA Loss adaptive to the scale and adjacency, we design the two weights with the following consideration. $W_{Scale_{(j)}}$ is proportional to the scale of text instance I_j. A large $W_{Scale_{(j)}}$ makes $L_{var}(I_j)$ significant, which brings strong force to pull pixels as closer as possible to the lower $L_{var}(I_j)$. $W_{Dist_{(j,k)}}$ is proportional to the shortest distance between two instances I_j and I_k, which results in an extra force to push the embedding of two close text instances further apart. In contrast to $W_{Scale_{(j)}}$, a smaller $W_{Dist_{(j,k)}}$ (a short distance between I_j and I_k) makes $L_{dist}(I_j, I_k)$ larger. So when we minimize $L_{dist}(I_j, I_k)$, a smaller $W_{Dist_{(j,k)}}$ makes the model better at moving pixels of different instances away.

Given N text instances in an image, the final SA Loss



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

Figure 3. Comparison of SA Loss and Disc Loss on large (top) and small (bottom) text. (a) Input images. (b)-(c) Detection results and embedding visualization of the model trained by SA Loss; (d)-(e) Detection result and embedding visualization of model trained by Disc Loss.

takes the form of

$$L_{SA} = \frac{1}{N}\sum_{j=1}^{N} L_{var}(I_j) + \frac{1}{N(N-1)}\sum_{j=1}^{N}\sum_{k=1,k\neq j}^{N} L_{dist}(I_j, I_k). \quad (3)$$

Analysis. SA Loss contains two balance weights to adjust the pulling and pushing force according to scales and adjacency of text instances. By utilizing these two weights, clustering pixels of large instances and separating close text instances become much easier, even if the distance between two close text instances only compose of one or two pixels.

Figure 3 shows the detection results and embedding visualization of models trained by different loss functions including the discriminative loss (Disc Loss) [1]. Visualization is created by projecting original 8D embedding features onto a 2D space using Principal Component Analysis (PCA). Comparing Figure 3(c)-(e), the embedding distribution of pixels from the same instance is more compact and the distance among different clusters is larger. This means the SA Loss provides more precise instance information and more accurate detection results.

To better illustrate that the SA Loss can help detect large instances. We conduct experiments on TD500 where instances are large and long text. The results are listed in Table 4.

3.1.2 Segmentation Masks

The segmentation branch provides two segmentation masks to guide cluster processing. In our design, it connects 2D space (segmentation masks) and embedding space (Shape-Aware Embedding) to yield better results.

Design. The segmentation branch generates two 1-channel segmentation maps, namely Full Map and Center Map, by applying two separate 3×3 convolutional operations on features produced from previous module. Although both segmentation maps tell whether a pixel belongs to background or text, they serve different purposes. The Full Map reveals the overall location and distribution of text, while the Center Map only captures the center area of each text, allowing separation of spatially close text instances and providing reasonable starting points for pixel clustering later.

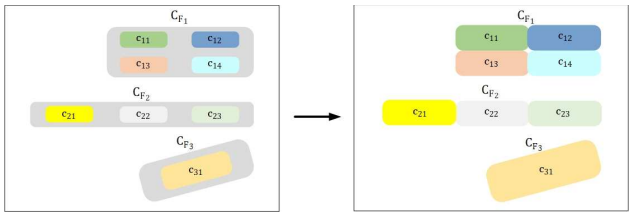


Figure 4. Illustration of cluster processing. Left: Three clusters $C_{F_i}(i = 1, 2, 3)$ output from the Full Map, and small clusters c_{ij} inside are from the Center Map. Right: Each pixel x ($x \in C_{F_i}, x \notin c_{ij}$) is assigned to a c_{ij} according to the shortest embedding distance between x and c_{ij}. Then we form text instances ($c_{ij} \cup p_{c_{ij}}$) in different colors. $p_{c_{ij}}$ represents the set of pixels x assigned to c_{ij}.

Loss Function. The Full Map and Center Map are both trained by minimizing the Dice loss [38] of

$$L_D = 1 - D(P, Q), \quad (4)$$

where P and Q represent the prediction and ground truth respectively. $D(,)$ is the Dice coefficient, which is formulated as

$$D(P, Q) = \frac{2 * \sum_{x,y} P_{x,y} Q_{x,y}}{\sum_{x,y} P_{x,y}^2 + \sum_{x,y} Q_{x,y}^2}. \quad (5)$$

The final loss for the segmentation branch is a weighted combination of the two maps, balanced by $\lambda \in (0, 1)$ as

$$L_{Seg} = \lambda L_{CenterMap} + (1 - \lambda) L_{FullMap}. \quad (6)$$

In our experiments, we set λ to 0.5, assigning equal importance to both maps. Note that text instances in the Center Map are shrunk from the instances in Full Map with a shrinking ratio r. Generally, r is set to 0.7, same as that in EAST. We keep the text instances in the Full Map without shrinking to reduce search space and ensure that following clustering is performed within a valid text region.

3.1.3 Overall Loss Function

The overall loss function used for training is

$$L = L_{SA} + L_{Seg}, \quad (7)$$

where L_{SA} is the SA Loss of the embedding branch and L_{Seg} is the loss of the segmentation branch.

3.2. Cluster Processing

As aforementioned, our model predicts three maps: Embedding Map, Full Map and Center Map. The Embedding Map is comprised of 8-channel embedding for each pixel.

The Full Map contains text regions in their original size with binary values (1 for text and 0 for background). Text instances in the Center Map are represented by the shrunk regions of the Full Map with a shrinking ratio of r. Here, we conduct pixel clustering by utilizing information from these three maps.

Specially, our algorithm first uses DBSCAN [9] to obtain two sets of clusters (C_{F_i} from the Full Map and $C_{C_i} = \cup_j c_{ij}$ from the Center Map). Then we assign each pixel inside C_{F_i} and outside C_{C_i} to the closest cluster $c_{ij} \in C_{C_i}$ by the following logic. Denoting pixels assigned to c_{ij} as $p_{c_{ij}}$, if the smallest embedding distance between the pixel and a cluster $c_{ij} \in C_{C_i}$ is still smaller than a threshold σ, this pixel is assigned to the closest cluster c_{ij} as part of $p_{c_{ij}}$. Otherwise, this pixel is ignored. In other words, each pixel is assigned based on the embedding distance between pixel embedding and the average embedding of pixels belonging to each cluster c_{ij} of the Center Map ($c_{ij} \in C_{C_i} \in C_{F_i}$). After all pixels in cluster C_{F_i} are processed, a new cluster $c'_{ij} = c_{ij} \cup p_{c_{ij}}$ is formed. We continue to apply this course of action to the other center clusters c_{ij} until all center clusters are processed.

Finally, for each of the new clusters c'_{ij}, we generate a corresponding minimum bounding box as output.

4. Experiments

4.1. Datasets

We conduct experiments on three challenging datasets. They are oriented scene text dataset ICDAR15 [20], the long oriented scene text dataset MSRA-TD500 [55] and the curved scene text CTW1500 [32]. We pre-train our model with SynthText [11], and then finely tune it on other datasets.

SynthText contains more than 800 thousand synthetic images with nearly 8 million text instances. Text instances of SynthText are annotated on string (line), word and character level. We only use word-level annotation in the pre-train stage.

ICDAR15 comprises 1,000 training images and 500 testing images where text instances are annotated on word level by 4 vertices of quadrangle. Images in ICDAR15 dataset are taken by Google Glass in natural scenes. Instances suffered from motion blur and other problems are marked as 'DO NOT CARE'. In our training, we simply ignore these instances.

MSRA-TD500 is composed of 300 training images and 200 testing images collected from natural scenes. Text in MSRA-TD500 contains both Chinese and English. They are annotated on string (line) level. Since the size of MSRA-TD500 training images is small, we include additional 400 training images from HUST-TR400 [54] for training.

CTW1500 is a curved text dataset, which includes 1,000 training images and 500 testing images with over 10 thousand text annotations. It contains both horizontal and multi-oriented text instances. Text instances in CTW1500 are annotated by 14 vertices of polygons.

4.2. Implementation Details

The backbone of our network is ResNet50 [13] pre-trained on ImageNet dataset [23]. For each branch, we apply four inception modules [48] on four feature maps of ResNet50 (after max pooling) with 128 output channels.

Data augmentation is used. We first randomly rescale the longer edge of the input image to a length from 640 to 2,560. Then random rotation, transpose and flipping are performed. Finally, we randomly crop 640×640 patches from the rotated image as the training images. The optimizer we use for training is Adam [21]. Our implementation also includes batch normalization [19] and OHEM [47] whose ratio of positive and negative samples is $1:3$. All models are pre-trained on SynthText [11] with initial learning rate $1e-4$.

During inference, there are five hyper-parameters. Threshold σ and τ are for measuring the embedding distance on the Embedding Map and obtaining confident pixels from segmentation maps respectively in post processing. IoU threshold δ is for NMS [45] while eps and $MinSamples$ are for DBSCAN. In all our experiments, we use the same setting where σ is 1.0, τ is 0.7, δ is 0.5, and $(eps, MinSamples)$ is $(5, 8)$ when clustering on the Full Map and $(1, 3)$ when clustering on the Center Map.

4.3. Comparison with State-of-the-Arts

4.3.1 Quadrangular Text

We first evaluate our method on ICDAR15 and MSRA-TD500. With the evaluation criteria proposed in [20] and [55], we report the results in Tables 1 and 2.

For ICDAR15, similar to those of [46, 60, 25, 34, 16, 36], we evaluate our model with the original image size ($720 \times 1,280$). Since there are many small text instances in ICDAR15, we also evaluate our model with a bigger size as [49, 26, 6, 30, 53] for fair comparisons by re-sizing the longer side of the input image to 1,760 with the aspect ratio fixed. When evaluated at the original scale, our method achieves recall, precision and H-mean rate of 84.5%, 85.1% and 84.8%, outperforming previous methods [46, 60, 25, 34, 16, 36] tested at the original scale and is comparable to those of [30, 53] tested on larger-resolution input. When evaluated at the larger scale, our method achieves new state-of-the-art.

As for MSRA-TD500, because the majority of text instances are long and large, larger input does not make much improvement. Therefore, we simply re-size the longer side of the testing images to 800 to fit our model. As shown in Table 2, our method achieves 82.9% in H-mean, which is comparable with previous best performance (82.9% *vs.* 83.0%).

In general, our method yields prominent improvement on recall for both ICDAR15 and MSRA-TD500, since two segmentation maps in our method are the key factors. Our

Method	*Recall*	*Precision*	*H-mean*
CTPN [49]	74.2	51.6	60.9
SegLink [46]	73.1	76.8	75.0
EAST [60]	73.5	83.6	78.2
Lyu *et al.* [36]	70.7	**94.1**	80.7
TextBoxes++ [25]	76.7	87.2	81.7
RRD [26]	79.0	85.6	82.2
TextSnake [34]	84.9	80.4	82.6
EAA [16]	83.0	84.0	83.0
Lyu *et al.* [35]	81.2	85.8	83.4
FTSN [6]	80.0	88.6	84.1
FOTS [30]	82.0	88.8	85.3
IncepText [53]	80.6	90.5	85.3
Ours (W/O, 1280)	79.1	83.6	81.3
Ours (W/O, 1760)	82.9	85.8	84.3
Ours (1280)	84.5	85.1	84.8
Ours (1760)	**85.0**	88.3	**86.6**

Table 1. Results on ICDAR15. We do not include results of multi-scale testing and ensemble. Results including recognition are not compared. 'W/O' represents the result by only enlarging boxes generated by the Center Map.

method also works well in terms of precision. Compared to FOTS and IncepText, our method is segmentation-based, and it occasionally suffers from tiny clusters of pixels on text-like structures, which causes precision loss. Note that results marked with 'W/O' are obtained by directly dilating center areas for comparison, which will be discussed in Section 4.4.2.

4.3.2 Curved Text

We evaluate our model on CTW1500 to demonstrate the ability of our method to detect curved text. We follow the evaluation rules [32] and set the longer side of the input image to 800. The result is shown in Table 3, which is new state-of-the-art.

Compared with previous best method TextSnake, ours shows advantage in terms of both precision and H-mean where the relative improvement reaches 14.8% and 4.5% respectively. We present several detection results in Figure 6.

4.4. Ablation Study

We evaluate the Shape-Aware Loss designed for text detection and the cluster processing pipeline that utilizes embedding clustering to detect text instances. Discussion on the proposed two segmentation maps and the dual-branch design is also included.

4.4.1 Effectiveness of Shape-Aware Loss

To verify the effectiveness of our proposed Shape-Aware Loss (SA Loss), we compare SA Loss with Discriminative (Disc) Loss [1]. For fair comparison, we train a new model with Disc Loss[1] and keep the other setting fixed. The two models are both trained only on ICDAR15 dataset and evaluated at the original image scale. SA loss yields recall 79.6, precision 84.9 and H-mean 82.2. Recall and precision are improved by 4.4% and 3.9% respectively over the original Disc Loss (recall 75.2, precision 81.0 and H-mean 78.0), demonstrating the effectiveness of SA Loss in modeling text instances. Also, as the results on TD500 shows in Table 4, SA Loss can detect large instances better as well. The average intra embedding distance produced by SA Loss is 0.4 and the inter embedding distance is 1.9, compared to the Disc Loss that yields 0.5 and 1.7 respectively. It explains why SA Loss helps generate better results.

Method	*Recall*	*Precision*	*H-mean*
Zhang *et al.* [58]	67.0	83.0	74.0
Yao *et al.* [56]	75.3	76.5	75.9
EAST [60]	67.4	87.3	76.1
SegLink [46]	70.0	86.0	77.0
RRD [26]	73.0	87.0	79.0
ITN [50]	72.3	**90.3**	80.3
Lyu *et al.* [36]	76.2	87.6	81.5
FTSN [6]	77.1	87.6	82.0
IncepText [53]	79.0	87.5	**83.0**
Ours (W/O)	76.8	77.2	77.0
Ours	**81.7**	84.2	82.9

Table 2. Results on MSRA-TD500. 'W/O' denotes the result by only enlarging minimum bounding boxes generated by the Center Map.

Method	*Recall*	*Precision*	*H-mean*
CTPN* [49]	53.8	60.4	56.9
EAST* [60]	49.1	78.7	60.4
DMPNet* [31]	56.0	69.9	62.2
CTD [32]	65.2	74.3	69.5
CTD+TLOC [32]	69.8	74.3	73.4
TextSnake [34]	**85.3**	67.9	75.6
Ours	77.8	**82.7**	**80.1**

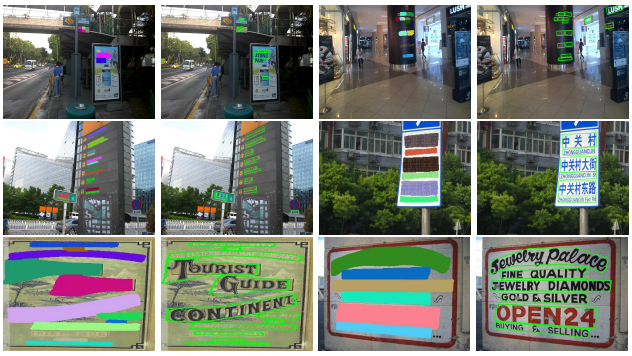
Table 3. Results on CTW1500. Results marked with * are collected from [32].

Methods	*Recall*	*Precision*	*H-mean*
Shape-Aware Loss	**81.7**	**84.2**	**82.9**
Disc Loss*	80.3	81.9	81.1
EAST *	66.2	72.1	69.0
EAST *(No Regression)	74.9	67.3	70.9

Table 4. Comparison on TD500. Results marked with $*$ are reproduced. Result of 'Disc Loss'[1] is produced with our proposed post processing method.

(a) (b) (c) (d) (e) (f) (g)

Figure 5. Comparison of cluster methods. From left to right are ground truth (a), embedding mask and box results of DBSCAN (b and c), MeanShift (d and e) and our proposed method (f and g) respectively.



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

Figure 6. Results on ICDAR15 (top), MSRA-TD500 (middle) and CTW1500 (bottom). (a) and (c) are clusters formed by embedding. (b) and (d) are results of detected bounding boxes. Some detected text regions in (b) and (d) that are not colored in (a) and (c) are directly from clusters of the Full Map where no embedding information is used.

4.4.2 Effectiveness of Cluster Processing

We verify the effectiveness of our proposed cluster processing method by answering two questions below.

Why not directly cluster on embedding? In [1], the instance masks are generated from clustering on embedding features masked out by segmentation mask. However, compared with direct clustering, our solution is better.

First, as shown in Figure 4, C_{F_i} from Full Map reduces search space from $\cup C_{F_i}$ to C_{F_i} when conducting clustering for each instance, which largely improves the efficiency of our solution. Then c_{ij} from Center Map in the post-processing are similar to pivots, providing accurate average embedding that guarantees general precision of clustering results.

In contrast, directly applying clustering algorithms may largely degrade the final performance. On one hand, the foreground mask cannot be 100% accurate. Thus the noise on the boundary between text and background is hard to avoid, resulting in inaccurate centroids in the embedding space for later cluster processing. On the other hand, our proposed post-processing pipeline builds a bridge between 2-D space (segmentation maps) and embedding space (Embedding Map), by which two branches become complementary to each other. We note the segmentation branch reduces search space by separating easy instances and provides accurate centroids at 2D space while the embedding branch helps separate close and difficult instances.

We make comparison with strategies where the clustering algorithm is directly applied (such as DBSCAN and MeanShift) to Embedding Map (masked out by Full Map), and show it in Figure 5 and Figure 7. Although our algorithm uses DBSCAN to generate clusters from two segmentation maps (Full Map and Center Map) before conducting label assignment to generate instances, the curve of DBSCAN (green) and ours (blue) show different trends when varying distance threshold (eps for DBSCAN, σ for our algorithm). This proves that there exists a gap between 2D space (segmentation maps) and embedding space (Embedding Map). Simply applying clustering algorithms on Embedding Map (masked by Full Map) overlooks useful information in the 2D space, leading to lower performance.

Why not directly dilate center areas? Directly dilating the minimum bounding boxes generated by center areas of the Center Map to cover the original text regions seems to be a feasible solution. However, it may fail in the following two cases.

The first occurs when spit center areas exist. As shown in Figure 8, for large or thin text instances, the predicted center area is sometimes split into several parts. In this case, expanding boxes generated by these areas may tear the true regions apart. Hence the performance is significantly degraded. The fracture, nonetheless, can be fixed by embedding clustering thanks to the bridge built by our pipeline. Like Figure 8(a) and (b), redundant clusters from the Center Map are enclosed by correct clusters from embedding.

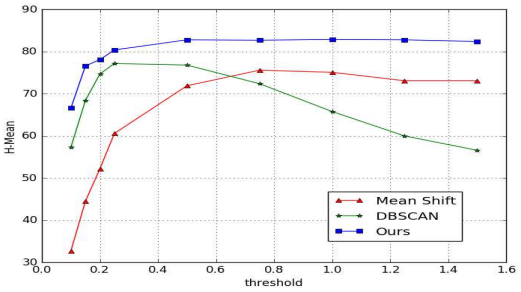
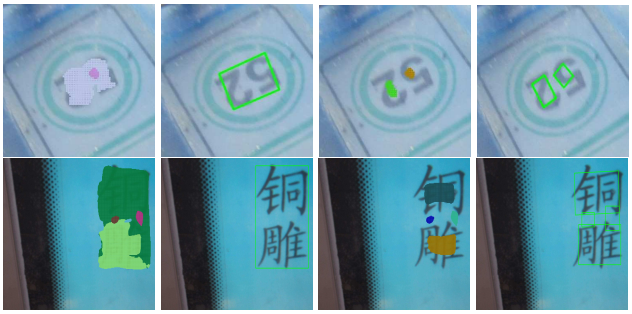


Figure 7. Comparison on MSRA-TD500 of directly applying clustering algorithms (MeanShift and DBSCAN) to Embedding Map and our proposed pipeline. The x-axis varies the threshold (bandwidth for MeanShift, eps for DBSCAN, and σ for our algorithm) used for measuring embedding distance and the y-axis is the H-mean.



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

Figure 8. Comparison between our pipeline and direct dilated minimum bounding box. Each cluster is marked with a random color. (a) Clusters from our pipeline. (b) Minimum bounding boxes of clusters in (a). (c) Clusters from Center Map. (d) Minimum bounding boxes of clusters in (c).

Thus false predictions are removed by NMS later.

The seconds failure stems from the uncertainty of expanding ratio where the predicted center area does not always cover exactly 70% of the original text area. When using a constant expanding ratio 1.43 (the reciprocal of shrinking ratio 0.7), boxes dilated from center areas are sometimes smaller or larger than the ground truth boxes. Although slight inaccuracy is tolerable based on IoU, this solution is not optimal for real-world OCR applications.

For a clearer comparison, we perform direct dilation on boxes generated by center areas on ICDAR15 and MSRA-TD500 datasets. The results have been showed in Tables 1 and 2. Results from enlarging minimum bounding boxes are marked with (W/O). All results show that generating boxes from our embedding clustering is more robust and effective.

4.4.3 Effectiveness of Network Design

Importance of Full Map and Center Map. The Full Map first separates the more obvious text/non-text areas to help the Center Map focus on separating close text areas and reduce computation overhead. The Center Map utilizes mean embedding to improve post-processing. Direct dilating boxes generated by Center Map may produce many false predictions as shown in Tables 1 and 2. Replacing Full Map with dilated Center Map is not feasible since dilatiion ratio is hard to estimate for each instance (with Recall 67.8, Precision 70.4, H-mean 69.1 on ICDAR15). As Figure 7 shows, directly applying embedding distance based clustering on Full Map is not ideal. Therefore both maps are indispensable.

(a) (b) (c)

Figure 9. Sample images of failure cases. The missing ground truth boxes are in yellow and false predictions are circled in red.

Effectiveness of Dual-Branch Network. To manifest the effectiveness of the mirror-like dual-branch design, we merge segmentation branch and embedding branch into a single branch. In this case, three output maps are generated by a shared feature merging module. Note that if other parts keep the same, combination of the two branches reduces half of the parameters. In order to eliminate the effect brought by different parameter numbers, in the model with a single branch, each extracted feature from intermediate layers of ResNet50 has 256 channels, which doubles the channel number (128) in the dual-branch design. The result of using a single branch on ICDAR15 is with recall 77.2, precision 81.4, and H-mean 79.2.

4.5. Limitations

Because our pipeline needs to perform clustering twice, the inference speed on 720P images from ICDAR15 is average 3FPS on a single NVIDIA TITAN X Pascal GPU. In addition, sample images of failure cases are shown in Figure 9 where erroneously suppressed small words, text-like structures and hard instances cause performance reduction.

5. Conclusion

We have presented a new framework for detecting scene text of arbitrary shape. Our model with two individual branches can simultaneously generate text masks and embedding features. We introduced a Shape-Aware Loss and a new cluster processing pipeline to distinguish among text instances with various aspect ratios and small gaps among them. Experiments on benchmark datasets demonstrate the effectiveness and robustness of our proposed model. Possible future work includes extending our findings to text spotting task and further shortening the running time.

References

[1] B. D. Brabandere, D. Neven, and L. V. Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017. 1, 2, 3, 4, 6, 7

[2] M. Bušta, L. Neumann, and J. Matas. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *ICCV*, 2017. 2

[3] C. K. Chng and C. S. Chan. Total-text: A comprehensive dataset for scene text detection and recognition. In *IAPR*, 2017. 1

[4] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016. 2

[5] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016. 2

[6] Y. Dai, Z. Huang, Y. Gao, and K. Chen. Fused text segmentation networks for multi-oriented scene text detection. In *ICPR*, 2018. 2, 5, 6

[7] D. Deng, H. Liu, X. Li, and D. Cai. Pixellink: Detecting scene text via instance segmentation. *arXiv preprint arXiv:1801.01315*, 2018. 2

[8] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *CVPR*, 2010. 2

[9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, 1996. 5

[10] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017. 2

[11] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *CVPR*, 2016. 2, 5

[12] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *ICCV*, 2017. 2

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 5

[14] P. He, W. Huang, T. He, Q. Zhu, Y. Qiao, and X. Li. Single shot text detector with regional attention. In *ICCV*, 2017. 2

[15] T. He, Z. Tian, W. Huang, C. Shen, Y. Qiao, and C. Sun. An end-to-end textspotter with explicit alignment and attention. In *CVPR*, 2018. 1, 2

[16] T. He, Z. Tian, W. Huang, C. Shen, Y. Qiao, and C. Sun. An end-to-end textspotter with explicit alignment and attention. In *CVPR*, 2018. 5, 6

[17] W. He, X. Zhang, F. Yin, and C. Liu. Deep direct regression for multi-oriented scene text detection. In *ICCV*, 2017. 2

[18] H. Hu, C. Zhang, Y. Luo, Y. Wang, J. Han, and E. Ding. Wordsup: Exploiting word annotations for character based text detection. In *ICCV*, 2017. 2

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[20] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. K. Ghosh, A. D. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. ICDAR 2015 competition on robust reading. In *ICDAR*, 2015. 1, 5

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[22] S. Kong and C. C. Fowlkes. Recurrent pixel embedding for instance grouping. In *CVPR*, 2018. 2

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 5

[24] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, 2017. 2

[25] M. Liao, B. Shi, and X. Bai. Textboxes++: A single-shot oriented scene text detector. *IEEE Trans. Image Processing*, 2018. 2, 5, 6

[26] M. Liao, Z. Zhu, B. Shi, G.-s. Xia, and X. Bai. Rotation-sensitive regression for oriented scene text detection. In *CVPR*, 2018. 2, 5, 6

[27] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2

[28] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. 2

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016. 2

[30] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan. FOTS: fast oriented text spotting with a unified network. In *CVPR*, 2018. 1, 2, 5, 6

[31] Y. Liu and L. Jin. Deep matching prior network: Toward tighter multi-oriented text detection. In *CVPR*, 2017. 6

[32] Y. Liu, L. Jin, S. Zhang, and S. Zhang. Detecting curve text in the wild: New dataset and new solution. *arXiv preprint arXiv:1712.02170*, 2017. 1, 2, 5, 6

[33] Z. Liu, G. Lin, S. Yang, J. Feng, W. Lin, and W. Ling Goh. Learning markov clustering networks for scene text detection. In *CVPR*, 2018. 2

[34] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *ECCV*, 2018. 1, 2, 5, 6

[35] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In *ECCV*, 2018. 2, 6

[36] P. Lyu, C. Yao, W. Wu, S. Yan, and X. Bai. Multi-oriented scene text detection via corner localization and region segmentation. In *CVPR*, 2018. 2, 5, 6

[37] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, 2002. 2

[38] F. Milletari, N. Navab, and S. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016*, 2016. 4

[39] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *ACCV*, 2010. 2

[40] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *CVPR*, 2012. 2

[41] D. Novotny, S. Albanie, D. Larlus, and A. Vedaldi. Semi-convolutional operators for instance segmentation. In *ECCV*, 2018. 3

[42] S. Prasad and A. W. Kong. Using object information for spotting text. In *ECCV*, 2018. 2

[43] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2

[44] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[45] R. Rothe, M. Guillaumin, and L. J. V. Gool. Non-maximum suppression for object detection by passing messages between windows. In *ACCV*, 2014. 5

[46] B. Shi, X. Bai, and S. J. Belongie. Detecting oriented text in natural images by linking segments. In *CVPR*, 2017. 2, 5, 6

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

[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 5

[49] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *ECCV*, 2016. 1, 2, 5, 6

[50] F. Wang, L. Zhao, X. Li, X. Wang, and D. Tao. Geometry-aware scene text detection with instance transformation network. In *CVPR*, 2018. 2, 6

[51] Y. Wu and P. Natarajan. Self-organized text detection with minimal post-processing via border learning. In *ICCV*, 2017. 1, 2

[52] C. Xue, S. Lu, and F. Zhan. Accurate scene text detection through border semantics awareness and bootstrapping. In *ECCV*, 2018. 2

[53] Q. Yang, M. Cheng, W. Zhou, Y. Chen, M. Qiu, and W. Lin. Inceptext: A new inception-text module with deformable PSROI pooling for multi-oriented scene text detection. In *IJCAI*, 2018. 1, 2, 5, 6

[54] C. Yao, X. Bai, and W. Liu. A unified framework for multi-oriented text detection and recognition. *IEEE Transactions on Image Processing*, 2014. 5

[55] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *CVPR*, 2012. 1, 5

[56] 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. 1, 2, 6

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

[58] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. In *CVPR*, 2016. 1, 2, 6

[59] Z. Zhong, L. Jin, and S. Huang. Deeptext: A new approach for text proposal generation and text detection in natural images. In *Acoustics, Speech and Signal Processing (ICASSP)*, 2017. 2

[60] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. EAST: an efficient and accurate scene text detector. In *CVPR*, 2017. 1, 2, 5, 6