Aggregating Local Context for Accurate Scene Text Detection

Dafang He[1], Xiao Yang[2], Wenyi Huang,[1], Zihan Zhou[1], Daniel Kifer[2], and C.Lee Giles[1]

[1] Information Science and Technology, Penn State University
[2] Department of Computer Science and Engineering, Penn State University

Abstract. Scene text reading continues to be of interest for many reasons including applications for the visually impaired and automatic image indexing systems. Here we propose a novel end-to-end scene text detection algorithm. First, for identifying text regions we design a novel Convolutional Neural Network(CNN) architecture that aggregates local surrounding information for cascaded, fast and accurate detection. The local information serves as context and provides rich cues to distinguish text from background noises. In addition, we designed a novel grouping algorithm on top of detected character graph as well as a text line refinement step. Text line refinement consists of a text line extension module, together with a text line filtering and regression module. Jointly they produce accurate oriented text line bounding box. Experiments show that our method achieved state-of-the-art performance in several benchmark data sets: ICDAR 2003 (IC03), ICDAR 2013 (IC13) and Street View Text(SVT).

1 Introduction

Scene text provides rich semantic cues about an image. Many useful applications, such as image indexing system and autonomous driving system could be built on top of a robust scene text reading algorithm. Thus detecting and recognizing scene text has recently attracted great attention from both research community and industry.

Even with the increasing attention in reading text in the wild, scene text reading is still a challenging problem. Unusual font, distortion, reflection and low contrast make the problem still unsolved. Algorithms for scene text detection could be classified into three categories [1]: (1) sliding window based methods, (2) region proposal based methods, and (3) hybrid methods. Sliding window based approaches [2, 3] try to determine whether a fixed-sized small image patch is a text area or not. Such methods need to examine every locations in different scales in an exhaustive manner, thus are very time consuming. Region-based methods [4–9] propose a set of connected components as candidate text regions with predefined rules. These approaches greatly reduce the number of image patches needed to be examined, however, they may miss some text areas or generate regions with too many characters connected. Hybrid methods [10, 11] integrate the region based methods with the sliding window methods to combine

the advantages of both categories. One of the common components in these methods is a classifier that determines whether a given image patch or a region is text or not. Given the success of the convolutional neural networks(CNN) in object detection and recognition [12], they have also been used as the text/non-text classifier in scene text detection tasks [13, 8, 14]. The strong power of CNN makes scene text detection more robust to outlier noises. A typical pipeline is to first generate a set of region proposals and then apply a binary classifier trained on text/non-text data. Although this method is efficient but even CNN trained on millions of samples are not stable for robust scene text reading on complex scenes. This is because context is often necessary for disambiguation. It is needed to determine whether a vertical line is an "I", a "1", or part of background noise like the space between bricks or the edge of a window. In Fig. 4 (a), we show some examples of text-like background noise which are confusing, even to people, when appearing without their contexts.

Recently, Zhu, et al [15] proposed to use highly-semantic context, which tried to explore the assumption that text are typically on specific background, such as sign board, but seldom on the others, for e.g, sky in natural image. However, modeling text which potentially exists in a wide variety of places is impossible with an exhaustive manner, and whether a new class of semantic object which does not appear in the training set will hurt results is in doubt. In addition, in a lot images that are not purely natural, text could be placed in unusual places, e.g sky. In those cases, the model might hurt the performance. In this paper, we define a region's local context to be its horizontal surroundings. A text localization algorithm is proposed which efficiently aggregates local context information in detecting candidate text regions. The basic idea is that the surrounding information of a candidate region usually contains strong cues about whether the region is text or background noise similar to text, and thus helping our model localize text more accurately. Some examples of these context images are shown in Fig. 4 (b). In addition, we also propose a grouping algorithm to form lines from verified regions and a line refinement step to extend text lines by searching for missing character components, as well as to regress text lines to obtain accurate oriented bounding boxes.

To be more specific, our major contributions are the following aspects:

1. A method that efficiently aggregates local information for cascaded and accurate classification of proposed regions. This step could be part of any other region based framework.
2. An effective grouping algorithm as well as a novel text line refinement step. Text line refinement includes a Gaussian Mixture Model(GMM) based text line extension module to find new character components, and jointly, a sliding window based oriented bounding box regression and filtering module. They are efficient and robust for post processing, and give an accurate oriented bounding box, instead of a mere horizontal bounding box.
3. A cascaded end-to-end detection pipeline for accurate scene text detection in unconstrained images. State-of-the-art performance is achieved in IC03, IC13 and SVT data sets.

In the following sections, we first describe related works on scene text detection in Section 2. Our method will be described in detail in Section 3 and experimental results are shown in Section 4. Several text detection results from images using the proposed algorithm are shown in Fig 1.

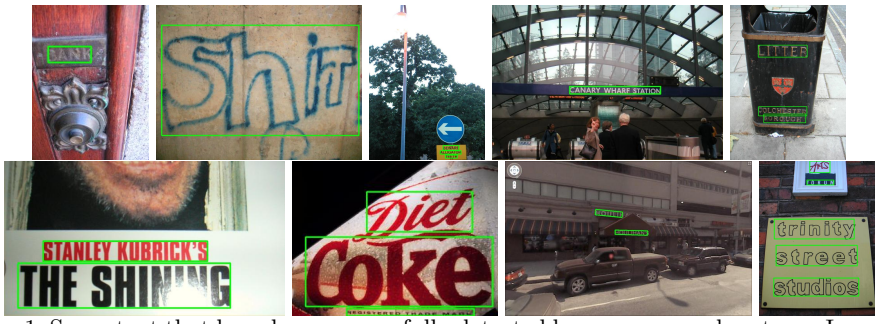

Fig. 1: Scene text that have been successfully detected by our proposed systems. Images are from IC13 and SVT dataset

2 Related Work

Scene text detection is much more challenging than Optical Character Recognition (OCR). Early work in scene text detection focused on sliding window based approaches. Chen et al [3] proposed an end-to-end scene text reading system which used Adaboost algorithm for aggregating weak classifiers trained on several carefully designed, hand crafted features into a strong classifier. They used sliding window to find candidate text regions. However, the scales of text lines in scene images vary a lot so sliding window based methods are typically very inefficient. Region based methods have recently received more attention. Most works focused on two region based methods: (1) Stroke width transform (SWT) [6] and its variants [16, 7]; (2) Extreme Region (ER) detector [5, 8] and Maximally Extreme Region (MSER) detector [9]. SWT explicitly explores the assumption that text consists of strokes with nearly constant width. It first detects edges in the image and tries to group pixels into regions based on the orientation of the edge. However, its performance is severely decreased when the text are of low contrast or in unusual font style. ER based methods [5, 8, 17, 9] are now popular since they are computationally efficient and achieve high recall as well. Neumann et al [5] proposed an ER based text detection algorithm which utilized several carefully designed region features, such as hole area ratio, Euler number, etc. However, the designed region features are not representative enough to remove background noise that is similar to text. Several other works [17, 18] follows a similar patterns in the classification step. These region features are fast to compute, but they typically lack the ability of robust classification.

More recently, CNN based methods have been used with significant success [8, 13]. The ability to synthesize data for training, which was specifically explored in [19], has greatly accelerated the research in scene text detection due to the unlimited amount of training data. Several works [14, 9], trained on synthetic images, have achieved the state-of-the-art performance in real image as well.

However, even with the introduction of CNN models and synthetic image generation for training, it is still hard for the CNN models to achieve good performance in some complex scenarios. We observe that the main reason of this incorrect classification lies on the fact that some characters have simple shapes that are also contained in non-text objects and CNNs are often fooled into classifying such objects as text. In this work, we explore ways to overcome these challenges and propose a system for efficient and accurate scene text detection.

3 Methodology

Our proposed detection pipeline is as follows. First, an ER detector is conducted on 5 channels of an input image. For each detected region, we first classify it to get a coarse prediction and filter out most non-text regions. Then the local context is aggregated to classify the remaining regions in order to obtain a final prediction. Text lines will be formed on top of a character component graph by grouping the verified regions with similar properties together. A text line refinement step is also designed to further filter out false positive and obtain accurate oriented bounding boxes. Several successive image examples of the proposed method can be seen in Fig. 2.

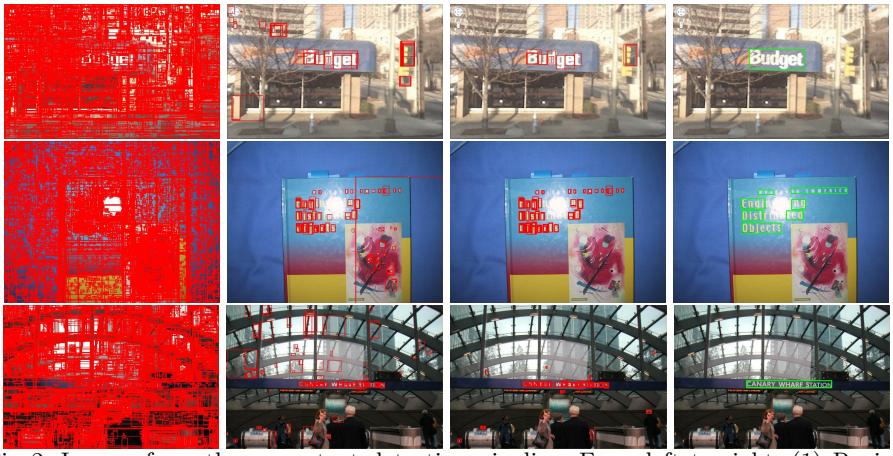

Fig. 2: Images from the scene text detection pipeline. From left to right: (1) Region proposals; (2) Coarse predictions; (3) Predictions after aggregating local context; (4) Final detected text lines.

3.1 Cascaded Classification and Local Context Aggregation

Context information is critical for object detection and recognition [20, 21]. Previous region based methods often focus on classifying each region independently and the image patch is cropped tightly from a generated region [8]. Here, we consider the local context of a given region as its local surroundings and argue that surrounding information should be incorporated in determining whether a given region should be classified as text or not. We observe that characters, which are often represented as simple shapes such as "I" or "l", cannot be well distinguished from background noises that are similar to them. However, text in an image is often represented as lines of characters, and for a given region, its

local surroundings give rich information about whether it is text or not. There have been works that explored relation between text regions before, such as [17, 18]. They proposed to use graphcut on top of MSER region graph to refine the results. Instead, we try to aggregate more higher level information in classification step of each region by the proposed network, and we show that this aggregation provides rich information. Some background regions, which are difficult to be distinguished from text when cropped from a tight bounding box, can be accurately predicted by our model after we aggregate this context information.

Design Rationale: The architecture of the proposed framework is shown in Fig. 3. We call this network a text local feature aggregation network (TLFAN). This is a two-column CNN with joint feature prediction on the top. It is designed for cascaded classification that will be explained in the next part. One CNN branch with fully connected layers is for coarse prediction, and we refer to this branch as standard CNN. The other branch takes an input with aggregated local information to produce a context vector, and we refer it as context CNN. The first column of the architecture is for learning features from the tight image patch which we are focusing at, and the other is for learning features from its surroundings. This CNN structure is specifically designed for scene text reading and several design rationales are here: (1) local context typically provides rich information about whether a specific region is text or not. Some background noise can not be well distinguished with text from a mere tight bounding box. In Fig. 4, there are some example image patches cropped from IC13 where traditional text/non-text binary classification will easily fail. But our model, by aggregating local context, can robustly distinguish it from text. (2) Even though here we consider text lines with arbitrary orientation, horizontal neighborhood already provides rich clues of whether the given region is text or not. (3) Since the input to the proposed TLFAN and its context have the same scale, we can use shared CNN parameters for the two columns and thus reduce the number of parameters that need to be learned. We also tried to use central surround network [22] which will consider a larger surrounding region. However, by doing so, we will have to learn more parameters either in CNN part or in fully connected part. In addition, We found that this will not improve the performance as much as the proposed manner, and it is likely to cause overfitting, since it considers information that is mostly unrelated.

Region Proposal and Cascaded Classification: We use an ER detector as our region proposal because of its efficiency and robustness. We extract ERs from RGB, Gray scale and gradient of intensity channels. In order to achieve high recall, approximately thousands of regions will be generated from the 5 channels for each image. We preprocess each region as described in [8] and resize them into 32×32. We then run the standard CNN branch to remove false positives in a similar manner as [8]. For regions with aspect ratios larger than 3.0 or smaller than 1/3, we will do sliding window on top of it since each region might contain blurred text with several characters connected, and in that case, we should not simply resize and classify them. In our experiment, **91.5%** of the regions will be removed, and it achieves **92%** recall on the IC13 testing set.

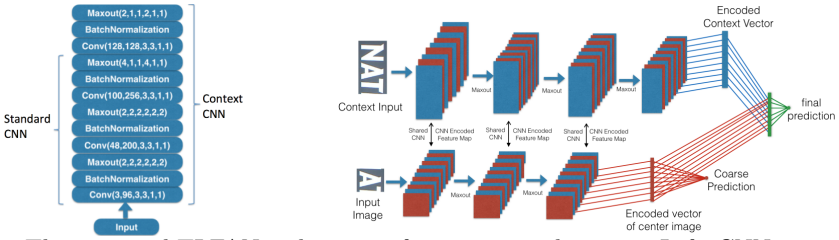

Fig. 3: The proposed TLFAN architecture for scene text detection. Left: CNN structure of the proposed network. The bottom 3 layers of convolution and pooling are shared, and for context branch, another CNN layer and pooling layer is added to produce deeper representation. (b) The whole architecture of the network. One column is for the given patch that we are trying to classify. The other is for extracting context information for this patch, and the generated feature vector will be further used to give an accurate prediction of the region.

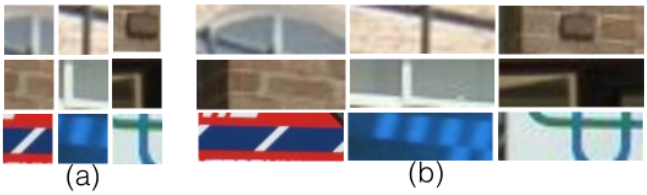

Fig. 4: Cropped image patches which demonstrate that local context helps in distinguishing between background noise and text. (a) The original image patches cropped tightly from the generated regions. (b) The horizontal context images which correspond to the region on the left, respectively. All the examples here are background noises which easily cause false positive if we only consider a tight bounding box for classification. Instead, our model can efficiently aggregate local context, so as to give accurate prediction.

After this step, the retained regions will be passed into the context branch to generate context vectors. To be more specific, we calculate the width and height of the region, and extend the patch in the horizontal direction so that the resulting context input patch is with 3 times the width of the original image patch. Mean value is padded when there is not enough space in the image for the context input. Because of the strong ability of CNNs in extracting high level information, this context vector provides rich cues in helping with the classification of the given region. This step can further remove some false positives that are similar to text and only regions with high confidence of text will be retained. In IC13 testing set, **94.5%** regions will be removed and it achieves **91%** recall.

In the final prediction, we still consider the two column structure instead of only the context branch for tow reasons: (1) The generated feature vector of standard CNN already contains rich information and it actually produces the feature of the region we are looking at. (2) It will be much easier to train since the standard CNN already produce a really meaningful result. For the context column, they only need to 'figure out' that in some certain cases, the input is not a text even though the feature produced by standard CNN says that it is close to text. Such cases include, but not limited to, repetitive patterns, corner of objects and so on.

Training: The proposed model is more difficult to train than a simple text/non-text classifier. We follow the same manner as described in [19, 14] by synthesizing image patches for training which provides unlimited number of training data. Since our proposed architecture needs context information, our synthetic positive images need to cover different situations that will happen in real natural images, such as characters with one or two near neighboring characters. Randomly cropped images with their context from several image sources will be considered as negative samples. Several example images for training are shown in Fig. 5. In order to train a better classifier, a two-step training scheme is introduced. First we train a character recognizer with negative samples. This is a 46 classes classification problem, and the positive 45 classes contain all 10 digits and letters with both capitalized or lower cases. Here we merged several similar shaped characters into one class. For example, 'O' and 'o', '0' will be merged into one class. We train with negative log likelihood as the loss function:

$$\mathrm{NLL}(\theta, D) = -\sum_i log(Y = y^i | x^i, \theta) \quad (1)$$

and the 46 classes training makes the learned filters better than binary text/non-text training.

Fig. 5: Several training samples. left: positive training samples and their context input patches. Right: negative training samples and their context input patches.

The parameters of the trained convolutional layers are then used to initialize the proposed TLFAN architecture and only the fully connected layers will be tuned. After the loss become stable, we train the two parts jointly with smaller learning rate for finetuning. In addition to this, it is necessary to collect harder examples, and we will explain in Section 3.2.

Fig. 6 shows several images demonstrating the effectiveness of the proposed network. It can effectively use local context to determine whether a given region is text or not, and thus make the prediction more robust. The generated saliency image is the raw output by sliding the classifier on the whole image. We could see that even a well-trained text/non-text classifier [13] has problem when background is noisy. However, by aggregating local context, our model gives much more robust performance.

3.2 Hard Example Mining

In this section, we are going to describe how we collect hard training samples. This could also been seen as a way of bootstrapping. Mining hard examples is a critical step for training accurate object detector, and here we focus on hard negative examples. One of the reasons is that most training examples cropped from negative images have few geometric patterns. Training on these negative examples will make the model less robust to noisy backgrounds. So here we collect more hard negative training data from two sources: (1) **ImageNet**: We specifically collect images from several challenging topics, such as buildings, fences,

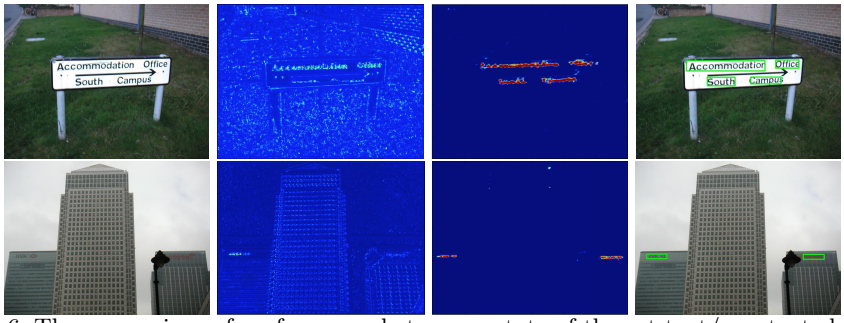

Fig. 6: The comparison of performance between a state-of-the-art text/non-text classifier proposed in [13], and our method in two challenging image in IC13 test set. From left to right: (1) Original image; (2) The saliency image generated by [13]; (3) The saliency image generated by our method; (4) Final detected text lines by our method.

and trees, and windows. These are objects that typically cause troubles in text detection, since their shapes are close to text. (2) **Synthetic hard negative samples**: we also synthesized a large bunch of negative samples. These samples are not texts but with similar structures as texts, such as stripes and cluttered rectangles. We follow a typical, iterative manner by training until the loss becomes stable and testing on these data to collect hard examples.

We found that this step improved the robustness of the text detector, especially on some hard testing images. These hard examples are also used in the later part for line refinement training.

3.3 Character Component Graph

Grouping text lines from regions is conducted after classification. This step differentiates text detection and object detection. Previous methods [6] typically used relatively heuristic rules to connect text components. This will cause troubles when there are false positives regions. Here we treat it as an assignment problem on top of a character component graph, and the best assignment without conflicts will be chosen based on scores calculated by several text line features as well as several empirical and useful selection standards. In this section, we first describe the ways of how we build the graph. Then we describe how we optimize on the assignment of each character component. The proposed algorithm is illustrated in Fig. 7.

Character Graph We first build a connected graph on top of the extracted regions, and each node represents a verified region, and each edge represents the neighborhood relation of the text components. We define a function Sim which calculates the similarity between two regions with several low level properties:
(1) Perceptual divergence p, which is calculated as the KL divergence between the color histogram of two regions: $\sum_i x_p(i) * log(\frac{x_p(i)}{y_p(i)})$ where x and y represent two regions and $x_p(i)$ represent the ith entry in its color histogram.
(2) Relative Aspect ratio: $a = \frac{x_{AspectRatio}}{y_{AspectRatio}}$.
(3) Height ratio $h = \frac{x_{Height}}{y_{Height}}$.

(4) Stroke width ratio $s = \frac{x_{StrokeWidth}}{y_{StrokeWidth}}$, which is calculated by distance transform on the original region.

We trained a logistic regression on these four region features extracted from IC13 training set to determine whether two given regions are similar. For each region, we further define its neighbor as: $y \in N(x)$ if $Dis(x, y) < \theta_1$ and $Sim(x, y) > \theta_2, \forall x, y \in R$, where R represents all the regions that have been verified by previous process, and $N(x)$ represents neighbor of region x. Dis means the distance between the center of the two regions. In our experiment, we set threshold θ_1 as $3 * Max(x_{Height}, x_{Width})$ where x_{Height},x_{Width} means the height and width of the region. We set θ_2 as 0.5 to filter out regions with less probability as being its neighbor.

Stable Pair we first define stable pair as pair of regions x and y where they belong to neighbor of each other, and they has a similarity score $Sim(x, y) > 0.8$. In addition, their distance should be no more than twice the shortest distance from all other neighbors to the region. This definition aims at obtaining more "probable pairs", since only pairs which "prefer" each other as their neighbors will be considered as "stable". After going through all the regions in $O(n)$ time complexity, we will obtain a set of stable pairs. Outliers are typically not able to form a stable pair with real characters, since real character will not prefer an outlier as its neighbor. In the first row of image in Fig. 7, the defined stable pair criterion successfully prevent generating vertical lines as possible candidate lines. Note that 0.8 is selected empirically from SVT training set. The overall performance is not too sensitive to it.

Optimization In order to optimize on the assignment of each region to one of lines, for each stable pair, we estimate its orientation based on their center points, and then conduct a greedy search algorithm to find components that align with the current line. Note that it is possible to find conflicting lines because two lines might share the same region components. In order to resolve the conflict, a score is calculated for each line based on the following properties: the average, standard deviation, max value, min value of the pairwise angle, perceptual divergence, size ratio and distance of neighbor components along the line. We will calculate these 16 features in total and a Random Forest [23] is trained to give each line an alignment score. For conflict alignment, we will choose the the best assignment. Here, several empirical but useful standards are also applied. For example, assignment which creates more long lines will be preferred than assignment which creates more short lines as shown in Fig. 7. This step aims at resolving the different possible alignments and find true text lines.

3.4 Line Extension and Refinement

After we generate lines of regions, a line refinement step is taken to finalize the results. This step aims at two targets: (1) extend lines so as to find missing components. (2) filtering out false positive and predict a tight oriented bounding box. They aim at finding a better bounding box that cover the whole word and it's important for an end-to-end system which incorporates text recognition since the performance of recognition highly relies on accurate bounding boxes.

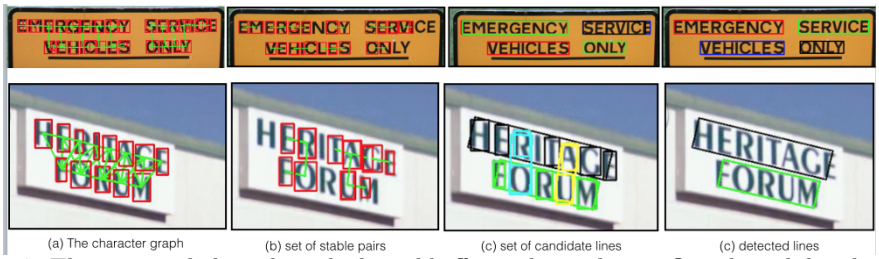

Fig. 7: The proposed algorithm which could effectively resolve conflicted candidate lines and find best line assignment of text regions. The final detected lines could be in any orientation. From left to right: (a) The constructed character component graph. (b) A set of generated stable pairs which will be used to create candidate lines. (c) Candidate lines represented as different colored bounding boxes. (d) Detected text lines.

Gaussian Mixture Model Based Line Extension Even with carefully designed classifier and grouping algorithm, there still might be some letters in a line that are not found in the previous steps. Here we propose a simple model to recover the proposed lines.

The model is based on the assumption that lines of characters typically have different color distribution with its direct background, and characters in one line typically follow the same color distribution. The algorithm pipeline is shown in Fig. 8. For each line that has been found by previous approaches with more than 2 regions, we crop the patch from the lines and estimate a GMM on the color of the patch. The Gaussian component associated with foreground (text region) is estimated by a voting mechanism: (1) we calculate the skeleton from the region patch; (2) for all pixels in the skeleton patch, we obtain which Gaussian component it belongs to by a simple voting mechanism conducted among these pixels. The reason for only using skeleton pixels lies on the fact that pixels that are close to the boundary are not accurate enough for color distribution estimation. For each line, we consider an square image patch whose side length is twice the height of the line, with its location on the two end of the lines. We estimate the color distribution in the region and a MSER detector is conducted on top of predicted color probability image. We filter out MSER regions whose size is too large or too small when compared to the height of the line. We then classify the retained region as being a text or not using the standard CNN branch in TFLAN. If its probability is larger than 0.4, then we will group it into the lines. If we find one additional character, the GMM is updated and will try to find more characters until nothing is found. Previous methods [17, 18] used a graphcut algorithm on top of the extracted regions which serves as similar purpose. However, if the graphcut is directly conducted on all the verified regions, it is still in doubt that whether it will also create more false positives. Here we use a more conservative method and only consider regions which could be easily attached to the current line that we already found.

Line filtering and Sliding Window Regression In this section, we propose a novel joint line filtering and regression model. Our model is based on making prediction in a sliding window manner on all the text lines that have

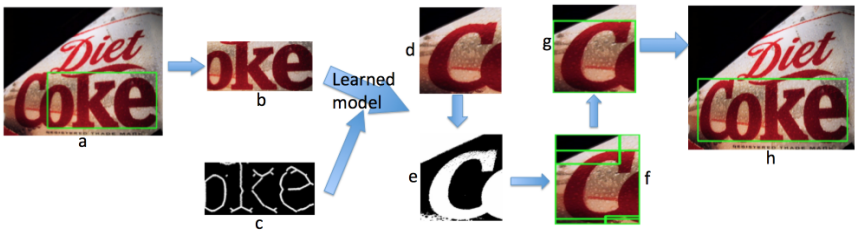

Fig. 8: Our proposed line extension pipeline: (a) The original image with the detected lines; (b) Cropped line image patch we used for estimation of GMM; (c) The skeleton of all the region components; (d) Cropped image patch whose color distribution needs to be estimated; (e) Predicted color probability image. Each pixel of it is predicted from the estimated GMM model; (f) MSER result on the estimation patch. Because of the large contrast in the predicted image patch, there are only few bounding box that we need to verify; (g) After running the standard CNN branch and non-maximum suppression; (h) The final detected line.

been verified in the previous steps. Existing methods[14] typically have two steps: (1) word-level verification. (2) bounding box regression. A CNN model is used for filtering and regression. Since fully connected layer only takes fixed sized input, the image patch needs to be resized in order to fit into the network. However, the length of text lines could vary according to the font, and number of characters. It is not desirable to resize an image patch with a text line of 2 to 3 characters to the same size as a text line with 10 characters. Instead, we consider joint regression and filtering in a sliding window manner.

The proposed architecture is shown in Fig. 9(a). The CNN is taken directly from the previous detection architecture. The proposed CNN model takes an input of 48×64 color image patch, and gives 7 prediction. One prediction is a simple part-of-word/non-word prediction which predicts whether the input patch containing part of word, or several characters. It is trained with negative log likelihood loss. The rest of 6 values all represent vertical coordinates because we are doing in a sliding window manner along the text line. Two of them are the minimal and maximal vertical values of the text in the current patch, and they are the same as the vertical coordinates that are predicted in traditional bounding box regression. The other four values represents the minimal and maximal vertical values in left and right side of the patch which are used to predict an oriented bounding box. Some training examples are shown in Fig. 9(b). We train the regression model with standard mean square error loss: $\text{MSE}(x, y) = -\sum_i (x_i - y_i)^2$. where x, y represent the predicted coordinates, and the ground truth coordinates, respectively. By predicting these four values, an oriented bounding box could be estimated. Since there are only a few lines in an image, even sliding window will not need much computation. Several predicted results have been shown in figure 9(c) on images cropped from IC13 and SVT data sets.

In order to refine the text lines based on the proposed architecture, we first crop the text line patch from the image, and resize the height of the text line to 48. We slide a window of height 48 and width 64 on the cropped patch. Background noise lines will be filtered out by part-of-word/non-word classification.

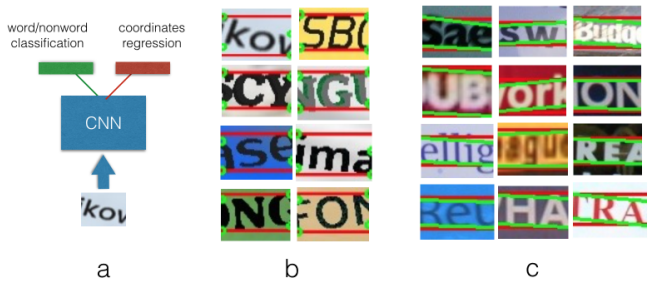

Fig. 9: Our proposed line refinement illustration:(a) The proposed architecture which used the CNN from detection part. Training is only on the fully connected layers for classification and regression. (b) Several examples of training images. The red lines are drawn from minimal and maximal vertical coordinates. The green dots are the vertical coordinates for oriented bounding box. (c) Several testing result images. The oriented bounding box is drawn with green lines instead of dots for better visualization of the orientation.

If the patch is predicted as part-of-word by the classifier, we will perform the oriented bounding box regression on it. A step by step example is shown in Fig. 10. Here we only show lines with text on them.

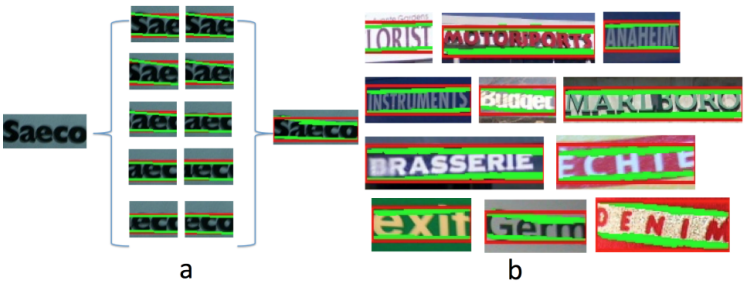

Fig. 10: Our proposed line refinement pipeline. (a) For each cropped lines, we do sliding window prediction and merge the results.(b) Several line regression results based on the proposed framework. The red lines correspond to standard regression, and the green lines represent oriented regression.

4 Experiments and Evaluation

In this section, we present an evaluation of the proposed method on several benchmark datasets. We report the precision, recall and F-measure scores on our detection results.

Implementation Details We implemented our algorithms in python and torch 7 [24] on a work station with 64G RAM and Nvidia GPU tesla k40 and 16 processors (3000MHz). All the generation of region proposals and post processing with different channels are parrallized.

ICDAR Robust Reading We tested our algorithm on IC13 and IC03 testing sets. IC13 and IC03 testing sets contain 233 and 251 testing images, respectively.

For IC13, it provides an online evaluation system where we evaluated our proposed method. For IC03, we evaluate our result according to the metric in [25]. The results are shown in Table 1. Evaluation shows that our algorithm gives good performance in both data sets.

method	precision	recall	F-measure
Neumann[5]	73	65	69
Shi[18]	83	63	72
Bai et[26]	79	68	73
Zamberletti[9]	86	70	77
Tian[27]	85	**76**	80
Zhang[28]	88	74	80
Our model no post	89	73	80
Our model	**90**	75	**81**

method	precision	recall	F-measure
Li[17]	79	64	71
Yao[16]	69	66	67
Kim[29]	78	65	71
Yi[30]	73	67	66
Epshtein[6]	73	60	66
Zamberletti[9]	71	**74**	70
Our model	**84**	70	**76**

Table 1: Localization performances on: left: IC13(%), right: IC03(%) data sets. Bold number outperforms previous methods. 'Our model no post' represents final results without line extension and refinement steps.

Street View Text The SVT data set contains 249 testing images used for evaluation. It was first introduced by Wang et al [31]. One of the problems om the data set is that it is not fully annotated: some text in the image are not included in the annotation. This problem has been mentioned in [14], and we call this annotation as *partial annotated*. Our proposed algorithm could efficiently detect most of the text in images and thus the unlabeled text will decrease the precision of detection result and makes it hard to compare with other methods. So we manually labeled all the text in the images following simple rules: (1) text is not too blurry to read by human; (2) it contains more than 2 characters. We call this version *fully annotated* dataset and we tested our algorithm on both versions of the dataset for evaluation. The performance is shown in Table 2. Fig. 11 illustrates several examples of *partial annotated dataset*, *fully annotated dataset* as well as our detection results. Experiments on the *fully annotated* dataset shows that our detection algorithm have good performance in SVT dataset as well.

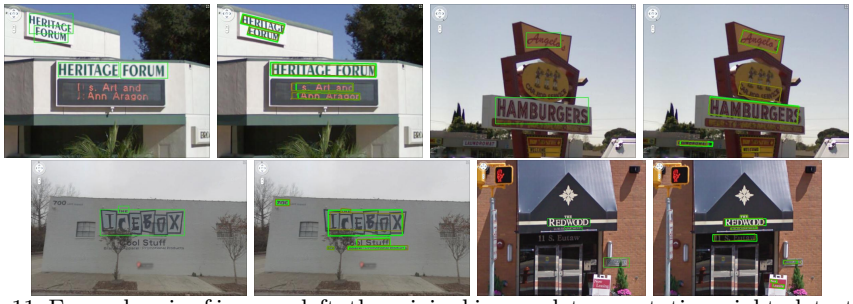

Fig. 11: For each pair of images, left: the original incomplete annotation, right: detection result of our model as well as the *fully annotated* ground truth. The *fully annotated* dataset provides oriented bounding box annotation. Green boxes represent our detected result which matches the ground truth. Yellow boxes represent the ground truth.

	partial annotated	*fully annotated*		
	recall	precision	recall	F-measure
jaderberg et al [14]	0.71	-	-	-
Our model	**0.75**	**0.87**	**0.73**	**0.79**

Table 2: Text detection performance on SVT. The bold results outperforms the previous state-of-the-art results. (1) *partial annotated*: detection recall measured with the partial annotation. The accuracy here does not makes sense, so we only tested its recall. (2)*fully annotated*: detection precision, recall, F-measure with full annotation.

Limitation Our proposed method achieved fairly good results in terms of precision, recall, and F-measure on standard datasets. However, it can still fail on several extremely challenging cases: (1) Text lines that are too blurry will cause problem in accurate region proposal generation as well as classification. (2) Strong reflection, too low contrast will still cause troubles in the detection. (3) Curved text lines might cause incomplete detection. Fig. 12 shows several failure cases that our algorithm cannot get good results. They are all challenging images in terms of text reading, and some of them are even hard for human to read.

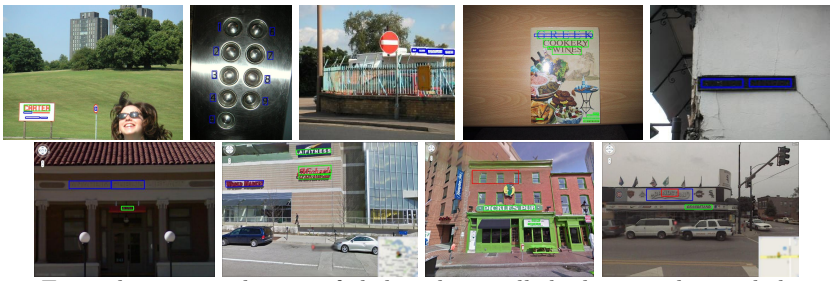

Fig. 12: Example images where we failed to detect all the lines or detected the wrong lines. The green boxes are the text lines that are correctly detected. The blue boxes are text lines that we fail to detect, and the red boxes are false positives, or incomplete detection.

5 Conclusions and Future Work

Here we proposed a novel scene text detection algorithm which efficiently aggregates local context information into detection as well as a novel two step text line refinement. Experiments show that our pipeline works well on several challenging images and achieved state-of-the-art performance on three benchmark data sets. Our future work will focus on extending our work in order to combine scene text recognition into an end-to-end scene text reading system. An image indexing system using text information retrieval will be implemented to help the visually impaired with shopping. Other applications for complex text such as sign reading will also be explored.

6 Acknowledgement

This work was supported by NSF grant CCF 1317560 and a hardware grant from NVIDIA.

References

1. Zhu, Y., Yao, C., Bai, X.: Scene text detection and recognition: Recent advances and future trends. Frontiers of Computer Science **10** (2016) 19–36
2. Wang, T., Wu, D.J., Coates, A., Ng, A.Y.: End-to-end text recognition with convolutional neural networks. In: Pattern Recognition (ICPR), 2012 21st International Conference on, IEEE (2012) 3304–3308
3. Chen, X., Yuille, A.L.: Detecting and reading text in natural scenes. In: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. Volume 2., IEEE (2004) II–366
4. Donoser, M., Bischof, H.: Efficient maximally stable extremal region (mser) tracking. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Volume 1., IEEE (2006) 553–560
5. Neumann, L., Matas, J.: Real-time scene text localization and recognition. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 3538–3545
6. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE (2010) 2963–2970
7. Huang, W., Lin, Z., Yang, J., Wang, J.: Text localization in natural images using stroke feature transform and text covariance descriptors. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 1241–1248
8. Huang, W., Qiao, Y., Tang, X.: Robust scene text detection with convolution neural network induced mser trees. In: Computer Vision–ECCV 2014. Springer (2014) 497–511
9. Zamberletti, A., Noce, L., Gallo, I.: Text localization based on fast feature pyramids and multi-resolution maximally stable extremal regions. In: Computer Vision-ACCV 2014 Workshops, Springer (2014) 91–105
10. Neumann, L., Matas, J.: Scene text localization and recognition with oriented stroke detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 97–104
11. Tonouchi, Y., Suzuki, K., Osada, K.: A hybrid approach to detect texts in natural scenes by integration of a connected-component method and a sliding-window method. In: Computer Vision-ACCV 2014 Workshops, Springer (2014) 106–118
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105
13. Jaderberg, M., Vedaldi, A., Zisserman, A.: Deep features for text spotting. In: Computer Vision–ECCV 2014. Springer (2014) 512–528
14. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Reading text in the wild with convolutional neural networks. International Journal of Computer Vision **116** (2016) 1–20
15. Zhu, A., Gao, R., Uchida, S.: Could scene context be beneficial for scene text detection? Pattern Recognition **58** (2016) 204–215
16. Yao, C., Bai, X., Liu, W., Ma, Y., Tu, Z.: Detecting texts of arbitrary orientations in natural images. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 1083–1090
17. Li, Y., Jia, W., Shen, C., van den Hengel, A.: Characterness: an indicator of text in the wild. Image Processing, IEEE Transactions on **23** (2014) 1666–1677

18. Shi, C., Wang, C., Xiao, B., Zhang, Y., Gao, S.: Scene text detection using graph model built upon maximally stable extremal regions. Pattern recognition letters **34** (2013) 107–116
19. jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Synthetic data and artificial neural networks for natural scene text recognition. arXiv preprint arXiv:1406.2227 (2014)
20. Divvala, S.K., Hoiem, D., Hays, J.H., Efros, A.A., Hebert, M.: An empirical study of context in object detection. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE (2009) 1271–1278
21. Oliva, A., Torralba, A.: The role of context in object recognition. Trends in cognitive sciences **11** (2007) 520–527
22. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 4353–4361
23. Liaw, A., Wiener, M.: Classification and regression by randomforest. R news **2** (2002) 18–22
24. Collobert, R., Kavukcuoglu, K., Farabet, C.: Torch7: A matlab-like environment for machine learning. In: BigLearn, NIPS Workshop. Number EPFL-CONF-192376 (2011)
25. Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R.: Icdar 2003 robust reading competitions. In: null, IEEE (2003) 682
26. Bai, B., Yin, F., Liu, C.L.: Scene text localization using gradient local correlation. In: Proc. of ICDAR'13, IEEE (2013) 1380–1384
27. Tian, S., Pan, Y., Huang, C., Lu, S., Yu, K., Lim Tan, C.: Text flow: A unified text detection system in natural scene images. In: Proc. of ICCV'15. (2015) 4651–4659
28. Zhang, Z., Shen, W., Yao, C., Bai, X.: Symmetry-based text line detection in natural scenes. In: Proc. of CVPR'15. (2015)
29. Koo, H.I., Kim, D.H.: Scene text detection via connected component clustering and nontext filtering. Image Processing, IEEE Transactions on **22** (2013) 2296–2305
30. Yi, C., Tian, Y.: Localizing text in scene images by boundary clustering, stroke segmentation, and string fragment classification. Image Processing, IEEE Transactions on **21** (2012) 4256–4268
31. Wang, K., Belongie, S.: Word spotting in the wild. Springer (2010)