WordSup: Exploiting Word Annotations for Character based Text Detection

Han Hu[1*] Chengquan Zhang[2*] Yuxuan Luo[2] Yuzhuo Wang[2] Junyu Han[2] Errui Ding[2]
Microsoft Research Asia[1] IDL, Baidu Research[2]
hanhu@microsoft.com {zhangchengquan,luoyuxuan,wangyuzhuo,hanjunyu,dingerrui}@baidu.com

Abstract

Imagery texts are usually organized as a hierarchy of several visual elements, i.e. characters, words, text lines and text blocks. Among these elements, character is the most basic one for various languages such as Western, Chinese, Japanese, mathematical expression and etc. It is natural and convenient to construct a common text detection engine based on character detectors. However, training character detectors requires a vast of location annotated characters, which are expensive to obtain. Actually, the existing real text datasets are mostly annotated in word or line level. To remedy this dilemma, we propose a weakly supervised framework that can utilize word annotations, either in tight quadrangles or the more loose bounding boxes, for character detector training. When applied in scene text detection, we are thus able to train a robust character detector by exploiting word annotations in the rich large-scale real scene text datasets, e.g. ICDAR15 [19] and COCO-text [39]. The character detector acts as a key role in the pipeline of our text detection engine. It achieves the state-of-the-art performance on several challenging scene text detection benchmarks. We also demonstrate the flexibility of our pipeline by various scenarios, including deformed text detection and math expression recognition.

1. Introduction

Understanding optical texts has a long history dating back to the early twentieth century [34]. For a long time, the attempts were made for texts of a few languages captured by very special devices, e.g. scanned English documents. With the growing popularity of smart phones, there have been increasing demands for reading texts of various languages captured under different scenarios.

We are interested in developing a common text extraction engine for various languages and scenarios. The first step is to localize texts. It is not easy. Firstly, languages differ in organization structures. For an example, English texts include visual blank separation between words, while Chinese not. For another example, regular human language texts are organized sequentially, while math expressions are structural. Secondly, texts may differ in visual shapes and distortions according to the individual scenarios. Nevertheless, all optical texts share one common property that they are all formed by characters, as illustrated in Fig. 1. It is natural and convenient that we base a common text detection framework on character detection.

Figure 1: The visual hierarchies for various language texts under different scenarios. Different languages and scenarios may differ in hierarchy, but they are all formed by a basic element, *character*.

When characters are localized, we can then determine the structure of texts in a bottom-up manner. The atomicity and universality of characters enable structure analysis for various languages and scenarios, e.g., oriented / deformed text lines and structural math expression recognition (see representative samples in Fig. 1).

The training of character detectors require a vast of location annotated characters. However, annotating character locations is very inconvenient and expensive, because the characters are small, easily gluing with each other and blurry. Actually, most existing large scale real text image datasets are labeled coarsely in word level, as illustrated in Table 1.

In this paper, we propose a weakly supervised learning framework to address the problem of lacking real charac-

*Equal contribution. This work is done when Han Hu is at IDL, Baidu Research.

Dataset	# im	# word	Real/Synth.	Anno.
ICDAR13 [20]	462	1,944	Real	char
ICDAR15 [19]	1,500	∼12K	Real	word
SVT [41]	350	725	Real	word
COCO-Text [39]	∼63K	∼174K	Real	word
IIIT 5k-word [30]	N.A.	3000	Real	word
Char90K [15]	N.A.	9M	Synth.	char
VGG SynthText [7]	800K	-	Synth.	char

Table 1: Popular datasets and their properties. Nearly all median and large scale real datasets are annotated in word level.

ter level annotations. It utilizes word annotations as supervision source to train the character detectors. Specifically, two alternative steps are iterated to gradually refine both the character center mask and the character model, as illustrated in Fig. 2. By applying this framework, we are able to train a robust character model by exploiting rich samples in several large scale challenging datasets, e.g. ICDAR15 [19] and COCO-Text [39].

The character model acts as a key module to our text detection pipeline. When applied to challenging scene texts, it achieves the state-of-the-art performance on several benchmarks, i.e. ICDAR13 [20], ICDAR15 [19] and COCO-Text [39]. It is also proved applicable on various scenarios, including deformed text line extraction and structural math expression recognition.

1.1. Related Works

There have been numerous approaches for text detection. According to the basic elements they rely on, the approaches can be roughly grouped into five categories:

Character based As mentioned earlier, character is a natural choice to build common detection engines. Nearly all existing character based methods rely on synthetic datasets for training [37, 9, 40, 42, 17, 51], because of lacking character level annotated real data. However, synthetic data cannot have a full coverage of characters from various scenes, limiting the model's performance in representing challenging real scene texts. Actually, none of the current top methods for the popular ICDAR15 benchmark [19] are based on character detection. Recently, some sophisticated synthetic technologies [7] have been invented that the synthetic text images look more "real". Nevertheless, real text images are still indispensable in training more robust character models, as we will show in our experiments.

Our pipeline is also character induced, but by incorporating a weakly supervised framework, we are able to exploit word annotations in several large scale real datasets to strengthen the character model. Using this model as a key to our pipeline, we achieve the state-of-the-art performance on several challenging scene text detection benchmarks. The pipeline is flexible for various scenarios such as deformed texts and structural math expressions.

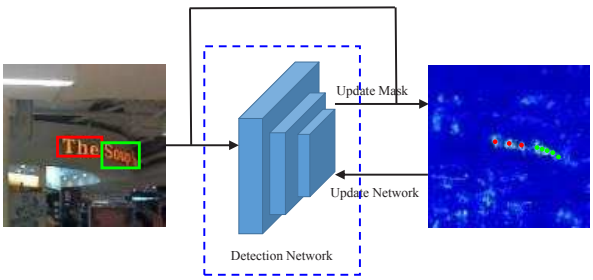

Figure 2: Illustration of our word supervision training approach for a character model. Two alternative steps are conducted: giving the current model, compute a response map which is then used together with word annotations to get a character center mask (red and green points); giving the character center mask, supervise the training of character model.

Text Line based Text line based methods directly estimate line models. These methods are widely adopted in the field of document analysis [29], where article layout provides strong priors. They are hard to be applied for non-document scenarios.

Word based A merit of these methods is that the modern object detection frameworks, such as faster RCNN [12] and SSD [24], can be conveniently adjusted [16, 7, 25, 50]. Yet, these methods are limited to languages which have word representation and visual separation between the words.

Component based Early component or word fragment based methods [13, 47, 14, 22, 43, 46, 18, 48] extract candidate text fragments by some manually designed features, e.g. MSER [3] and SWT [5], and then determine whether the fragments are real text or not. These methods once led some popular competitions for well focused texts, e.g. ICDAR13 [20]. However, the performance of these methods heavily degrades when applied to more challenging scenarios such as ICDAR15 [19] where texts are captured accidentally. In addition, as long as some texts are missed by the manually designed features, they would never be recalled in the subsequent steps.

Recently, some component based methods [49, 44, 8, 38, 35] attempt to learn text components by CNN feature learning. The components are either representative pixels [49, 44, 8] or segment boxes [38, 35]. These methods can learn from word annotations. In addition, text component is also a basic visual element, which may also benefit a common text detection engine. Nevertheless, our method takes advantages over these methods in the following aspects: first, characters provide stronger cues, e.g., character

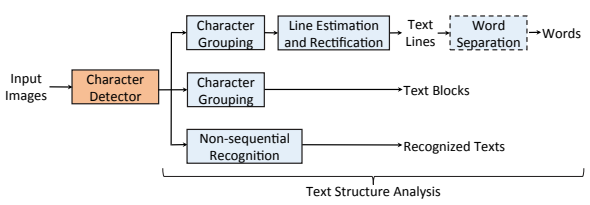

Figure 3: Our pipeline. There are two modules, character detector and text structure analysis. The pipeline is flexible for various scenarios ascribed to the atomicity and universality of characters.

scales and center locations, for the subsequent text structure analysis module; second, character is a semantic element, while component not. Thus our method is applicable to problems where direct character recognition is needed, e.g. match expression; third, our method can utilize loose word annotations for training, e.g. bounding box annotations in the COCO-Text dataset [39]. This is because our method can refine character center labels during training. For the above component based methods, their noisy labels are fixed which may harm training.

2. Our Approach

2.1. Pipeline

The pipeline of our approach is illustrated in Fig. 3. Given an image, we first detect characters on it. This module is shared by various languages and scenarios. Its performance is crucial for the whole pipeline. Instead of using synthetic character data alone for training, we strengthen it by exploiting word annotations from real scene text datasets. The details of our basic character detector and the word supervision method are presented in Section 2.2 and Section 2.3, respectively.

Then the detected characters are fed to a text structure analysis module, which is application dependent. We handle several typical scenarios. First is the sequential line, a widely used text structure. We propose a unified method to extract all of the horizontal, orientated and deformed text lines. English text lines are optionally separated into words for word based text recognition methods. Math expression recognition is another scenario, where characters are non-sequential. We first recognize all the detected characters and then recover structures connecting characters/symbols [11]. Details for text structure analysis are presented in Section 2.4.

2.2. Basic Character Detector

The fully convolution neural network is adopted, which has seen good performance on general object detection, e.g., SSD [24] and DenseBox [12]. Nevertheless, to be applied for characters, several factors need to be taken into account. First, characters may vary a lot in size on different images.

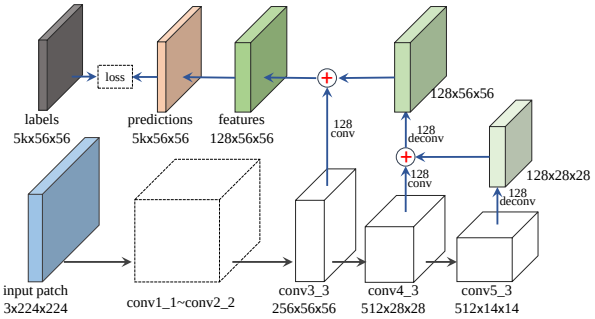

Figure 4: Our basic detection network. The network inherits from the VGG16 network model [36].

Some characters may be very small, e.g., 10×10 pixels in an 1M pixel image. Second, texts may appear in very different scenarios, such as captured documents, street scenes, advertising posters and etc, which makes the backgrounds distribute in a large space.

To cope with the character size problem, we use feature maps with higher resolution to generate character responses. Specifically, it is $1/4$ of the original image, other than $1/16$ or $1/32$ used in general object detection [31, 24]. Cues from deeper stages with coarser resolutions are merged for better representation power. We adopt the method in FPN [23] for such purpose, which uses an *eltsum* layer to merge features from two stages with $2\times$ resolution difference. It requires less parameters than other methods, e.g., [26, 32, 21], for producing the same number of feature maps. See Fig. 4 as an illustration. The network inherits from the VGG16 network model [36]. *conv5* features are $2\times$ up-sampled by deconvolution and merged with *conv4* features by an *eltsum* layer. The *eltsum*ed *conv5-conv4* features are merged with *conv3* features in the same way. The produced feature maps are used for both text/non-text classification and bounding box regression. $5k = (1 + 4)k$ score maps are generated, with 1 for text/non-text classification, 4 for bounding box regression, and k indicating the number of anchors. We use $k = 3$ anchors, representing characters with diagonal lengths of 24 pixels, 16 pixels and 12 pixels (on the 224×224 input patch), respectively. The characters with diagonal lengths of $0.7 \sim 1.4$ against the anchor's are regarded positive.

To ease the background variation problem, we adopt a two-level hard negative example mining approach for training. First is online hard negative mining [24]. All positives are used for loss computation. For negatives, only top scored ones are used in loss computation that the ratio between negatives and positives is at most $3 : 1$. Second is hard patch mining. During training, we test all training images every $10k$ iterations to find false positives (using the current character model). These false positives will be more likely sampled in the successive mini-batch sampling pro-

cedure.

Training 32 224×224 patches are randomly cropped from training images to form a mini-batch. 50% of the patches include characters. These positive patches are cropped from training images according to a randomly selected character and anchor, with some degree of translation/scale perturbation. The other 50% are randomly cropped but with no texts. After $10k$ iterations, we start to apply the hard patch mining procedure. For negative patches, half training patches will be hard ones which should include the current detected false positives.

Inference We conduct multi-scale test for an image. The used scales are $2^{\{0,-1,-2,-3,-4,-5\}}$, respectively. Since only down-sampling scales are involved, the computation overhead is afforded, at about 1.5 times, compared to single-scale test. NMS with IoU threshold of 0.5 is conducted to produce the final characters. It should be noted that multi-scale testing is indispensable for our basic detector, since we use anchors with only 3 scales. Exploring more efficient basic detector without the need for multi-scale testing will be our future work.

2.3. Learning From Word Annotations

Overview As illustrated in Table 1, most real text image datasets are annotated in word level, i.e. ICDAR15 and COCO-Text. Each word is attached with a quadrangle (e.g. ICDAR15) or a bounding box (e.g. COCO-Text) which tightly surrounds it, as well as a word category. In this paper, we suppose at least the bounding box of each word is available. If further a quadrangle or the word category is given, we use them to strengthen our word supervising procedure.

Our approach is inspired by [4] which successfully learns object segments from bounding box annotations. It is illustrated as Fig. 2. Two alternative steps are conducted: given a character model, automatically generate the character mask according to a word annotation; given a character mask, update the character network. These two steps are alternative in each network iteration. During the training, the character masks and the network are both gradually improved.

It is worth noting that the above procedure is only involved in network training. The inference is the same as in Section 2.2.

Character Mask Generation During forward and backward of each mini-batch, the first step is to generate character masks using the current character model and word annotations, as illustrated in Fig. 5 (bottom). First, we make forward using the current character model and get a set of candidate characters inside the annotated word bounding box. We select real characters from these candidate characters by maximizing score,

$$\begin{aligned} s &= w \cdot s_1 + (1-w) \cdot s_2 \\ &= w \cdot \tfrac{\text{area}(B_{\text{chars}})}{\text{area}(B_{\text{anno}})} + (1-w) \cdot (1 - \tfrac{\lambda_2}{\lambda_1}), \end{aligned} \tag{1}$$

where B_{chars} represents the bounding box of selected characters; B_{anno} represents the annotated word bounding box; area$(\cdot)$ denotes the area operator; λ_1 and λ_2 are the largest and second largest eigenvalues of a covariance matrix C, computed by center coordinates of selected characters; w is a weight balancing the two score terms. We find the learning is insensitive to the choice of w, and it is set as 0.5 by default. The first term of Eq. (1), s_1, favors larger coverage of selected characters to the annotated word bounding box, while the second one, s_2, prefers that all characters locate on a straight line.

We use a similar approach as in [45] to approximately maximize Eq. (1). Firstly, a maximum spanning tree [1], M, is constructed from the character graph G, which is built by the k-nn of all candidate characters with pair weights defined by their spatial distances and the current text/non-text scores,

$$w_{ij} = exp(-\frac{d(i,j)}{\overline{d}}) \cdot (t_i + t_j), \tag{2}$$

where $\overline{d}$ is the average of all distances between k-nn nodes; t_i denotes the current text/non-text score of candidate i. It is obvious that eliminating an edge in M equals to partitioning the characters into two groups. For each partitioning, we choose the group with larger score according to (1), and run the partitioning procedure greedily and recursively until the score (1) does not rise.

When a tight quadrangle or character number is given, we can further improve generated character mask: for the former, replacing computation of s_1 in Eq. (1) by area ratio of quadrangles; for the latter, adding a term to Eq. (1) that the mask prefers equal character number compared to ground truth.

Character Network Updating The generated character mask can be used as ground truth to supervise network training. We define a loss $\widetilde{\mathcal{L}}$ such that more reliable masks contribute more, as,

$$\widetilde{\mathcal{L}} = s\mathcal{L}, \tag{3}$$

where $\mathcal{L}$ represents a combination of the confidence loss and localization loss commonly used in modern object detection frameworks [31, 24]; s is the score computed by Eq. (1).

Fig. 5 shows the gradually updated character masks during training. During training, the character model is gradually improved.

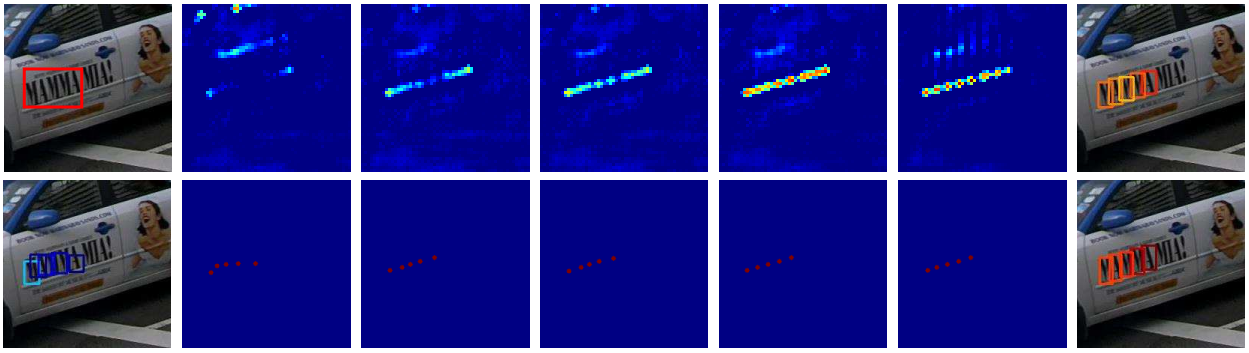

Figure 5: Updated character responses and the corresponding character masks during word supervised training on ICDAR15 datasets. The initial model in the second column is trained by $5k$ warmup iterations on synthetic data alone. The $3 \sim 6^{\text{th}}$ columns are responses during the word supervised training, where the epoch number means for ICDAR15 datasets. For illustration, we use bounding box annotations rather than the original quadrangles in training. Both the responses and character candidates are colored by their scores (indexed by *colormap* in Matlab).

2.4. Text Structure Analysis

Given characters extracted by the methods in Section 2.2 and 2.3, we conduct text structure analysis for various scenarios, e.g., text lines, words, text blocks, math expressions, and etc. Fig. 3 illustrates our text structure analysis methods for these typical text structures. For text line based applications, we propose a method which can handle arbitrarily deformed lines. The first step is to group characters. Then a line model is estimated to describe the line. With the model, we rectify the text line, which is usually required by modern sequential text recognition systems. Optionally, we separate lines into words. This is not necessary, but enables word based text recognition methods.

Characters can also be employed for text block extraction, e.g., document layout analysis [28], and non-sequential text recognition, e.g., math expression recognition [10].

In the following, we briefly describe techniques used for extracting text lines, which are frequently used in our experiments. More details can be found in appendix.

Character Grouping We adapt the method in [37] to group characters into text lines or blocks. Given characters with score larger than a threshold, [37] first builds a k-nn graph with each node denoting a character candidate. Unary and pairwise costs are defined on the graph to achieve clustering. The unary costs model relations between characters and the text category, e.g. character scores. The pairwise costs model relations between two characters, e.g. spatial and scale distances. A greedy min-cost flow algorithm is then conducted to obtain all character groups (see [37] for details).

The method in [37] is designed for horizontal text lines only. To be applied in oriented and deformed text lines, we introduce a higher-order cost which models relations between three characters. To reserve the efficiency of pairwise graph, we use character pairs instead of characters as graph nodes. The character pairs are spatially close characters with high scores and small spatial/scale distances. Then the unary and pairwise costs in the old graph can be modeled as unary costs in the new graph, while the higher order costs, e.g. angle distance, can be modeled as pairwise costs in the new graph. The same as in [37], We then conduct a greedy min-cost flow algorithm on the new graph to achieve character grouping. It can handle oriented and deformed text lines, ascribed to the introduction of higher-order costs.

Line Model Estimation and Rectification For each character group, we fit three text line models with increasing complexity. First is *0-order model*. Text lines are either horizontal or vertical. Second is *1-order model*. Text lines can be arbitrarily orientated. Last is a *piecewise linear model*, where a restricted polygon is adopted to represent a text line.

A model selection approach is conducted to choose a model with best balance between fitting accuracy and model complexity. Given the estimated model, we rectify the text line using thin plate spline (TPS) [2] method, where the vertexes of the text line model are used as control points.

Word partition Some text recognition systems can process only word inputs. To enable usage of such systems, we optionally separate text lines into words. An LSTM [6] based word blank detection method is applied on the rectified text line. Words are separated accordingly.

3. Experiments

In this section, we first do ablation studies on synthetic data where character level annotations are provided. Both our basic detector and the word supervision approach are evaluated. Then we apply our character induced text detection pipeline on scene text benchmarks. Finally, we show its applications to various scenarios.

3.1. Datasets and Evaluation

Four datasets are used in the experiments:

- **VGG SynthText-part**. The VGG SynthText datasets [7] consist of 800,000 images, generated by a synthetic engine proposed in [7]. The images have detailed character-level annotations. For experiment efficiency, we randomly select 50,000 images for training and 5,000 images for validation. This subset is referred to as *VGG SynthText-part*.
- **ICDAR13**. The ICDAR13 datasets [20] are from the ICDAR 2013 Robust Reading Competition, with 229 natural images for training and 233 for testing. The texts are annotated with character-level bounding boxes, and they are mostly horizontal and well focused.
- **ICDAR15**. The ICDAR15 datasets [20] are from the ICDAR 2015 Robust Reading Competition, with 1000 natural images for training and 500 for testing. The images are acquired using Google Glass and the texts accidentally appear in the scene without user's prior intention. All the texts are annotated with word-level quadrangles.
- **COCO-Text**. The COCO-Text [39] is a large scale dataset with 43,686 images for training and 20,000 for testing. The original images are from Microsoft COCO dataset.

The VGG SynthText-part is mainly used for ablation experiments. Both character-level and word-level evaluations are conducted by using the PASCAL VOC style criterion (≥ 0.5 Intersection-over-Union for a positive detection). For benchmark experiments on ICDAR13, ICDAR15 and COCO-Text, the evaluation protocols provided by the datasets themselves are adopted. Specifically, for ICDAR13 and ICDAR15, we use the online evaluation system provided with the datasets. For COCO-Text, the protocol provided by the dataset is used for evaluation.

3.2. Implementation Details

The VGG16 model pretrained on the ILSVRC CLS-LOC dataset [33] is adopted for all experiments.

Given different datasets, we train three character models. The first is trained by synthetic character data alone, i.e. 50k training images from VGG SynthText-part datasets. Second is trained on 1k ICDAR15 training images plus 50k VGG SynthText-part. 50% are sampled from ICDAR15 and the others are sampled from VGG SynthText-part. The third is trained on COCO and VGG SynthText-part, with mini-batch also sampled half-half from the two datasets. These three models are dubbed as "VGG16-synth", "VGG16-synth-icdar" and "VGG16-synth-coco", respectively.

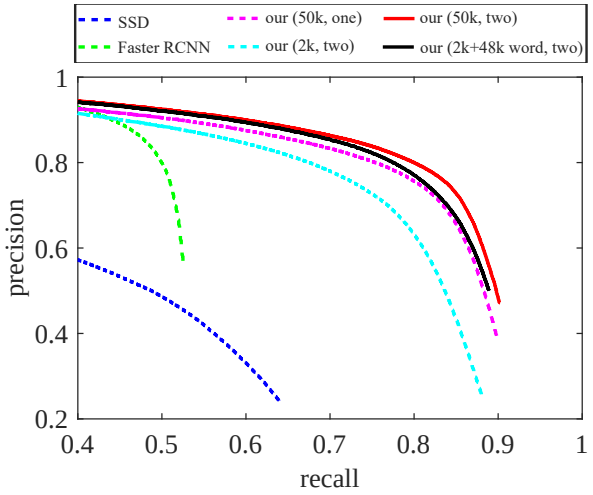

Figure 6: Character detection performance of our basic detection network, the faster RCNN and SSD methods on the VGG SynthText-part datasets. Four variants of our method are presented. The first term in brackets indicates the used supervision source. The second term indicates the used hard negative mining strategy, with "one" representing one-level hard negative mining and "two" representing two-level hard negative mining.

We use SGD with a mini-batch size of 64 on 4 GPUs (16 per GPU). A total of 50k iterations are performed for all models. For the "VGG16-synth" model, 30k are at a learning rate of 0.001, and the other 20k at 0.0001. For other models, 5k iterations with VGG SynthText-part character supervision alone are first run for warming up. The learning rate is 0.001 at this stage. Then 25k and 20k iterations are conducted using both character and word supervision at learning rates of 0.001 and 0.0001, respectively. The weight decay is set as 0.0005 and the momentum as 0.9.

For experiments on ICDAR13, ICDAR15 and COCO-text, the text line generation and word partition approaches introduced in Section 2.4 are applied to produce word localizations, which are required for evaluation of these benchmark datasets. For fair comparison, we tune hyperparameters of the line generation algorithm on a small fraction of training images, i.e. 50, for all character models.

3.3. Experiments on Synthetic Data

The VGG SynthText-part datasets are used.

Evaluation of the Basic Character Detector We first compare the proposed basic detection network presented in Section 2.2 with the state-of-the-art algorithms in the field of general object detection, e.g. faster RCNN [31] and SSD [24]. For faster RCNN and SSD, we directly use the codes provided by the authors.

Fig. 6 illustrates the precision-recalls of our basic network, faster RCNN and SSD on character detection, respectively. The main difference between our character network with the state-of-the-art general object detectors is that the feature maps used to produce character responses is finer than that of general object detectors ($1/4$ vs. $1/16$), while maintaining sufficient representation power by merging cues from deeper stages. The large gap between our basic network and general object detector demonstrates that reserving resolution is crucial for character detection. The two-level hard negative mining during training is also a plus that the second level hard patch mining can bring a moderate gain, as shown in Fig. 6.

Evaluation of Word Supervision Approach Three models are trained. The first is trained by randomly selected 2,000 images using character supervision. The second is trained using character supervision of all the 50k images. The third is trained using 2,000 character supervision images and 48,000 word supervision images. The training approaches are similar to those in 3.2.

From Fig. 6, it can be seen that the word-supervised model performs superior to 2k characters trained model and the performance degradation against the full 50k characters trained model is insignificant, demonstrating the effectiveness of our word supervision approach in exploiting weak word annotations for character model training.

3.4. Experiments on Scene Text Benchmarks

We apply our text detection approach on three real challenging scene text benchmarks: ICDAR13 [20], ICDAR15 [19] and COCO-Text [39]. These benchmarks are all based on word-level evaluation. Hence, the text line generation and word partition methods are involved. In the line model estimation step, we only use *0-order* and *1-order* models as nearly all text lines have up to orientation deformation.

Table 2, 3 and 4 show the performances of different methods on the ICDAR13, ICDAR15 and COCO-Text datasets. Our approach outperforms previous state-of-the-art methods by a large margin.

On ICDAR13, we achieve 90.34% F-measure, which is 2.65% higher than the second best one, i.e. CTPN [38].

On the more challenging ICDAR15 datasets, images are more likely to suffer from blurry, perspective distortion, extreme illumination, and etc. Our best model achieves a f-measure of 78.16%, with a large margin over the previous best method [25] (78.16% vs. 70.64%). Comparing our approach using different character models, VGG-synth-icdar performs much better than the VGG-synth model (78.16% vs. 69.18%). VGG-synth-icdar only adds *1k* training image compared to the VGG-synth model (*50k* training images). This indicates that the gain is from more *real* data, other than more data.

Method	Recall	Precision	F-measure
MCLAB-FCN [49]	79.65	88.40	83.80
Yao et al. [44]	80.22	88.88	84.33
Gupta et al.[7]	75.5	92.0	83.0
Zhu et al. [51]	81.64	93.40	87.13
CTPN [38]	82.98	92.98	87.69
our (VGG16-synth)	82.41	91.95	86.92
our (VGG16-synth-icdar)	**87.53**	**93.34**	**90.34**

Table 2: Performances of different methods on ICDAR13 using the DetEval criterion (%).

Method	Recall	Precision	F-measure
MCLAB-FCN [49]	43.09	70.81	53.58
CTPN [38]	51.56	74.22	60.85
Yao et al. [44]	58.69	72.40	64.77
SCUT-DMPNet [25]	68.22	73.23	70.64
RRPN-2 [27]	72.65	68.53	70.53
our (VGG16-synth)	64.37	74.79	69.18
our (VGG16-synth-icdar)	**77.03**	**79.33**	**78.16**

Table 3: Performances of different methods on ICDAR15 (%).

On COCO, our best model achieves 30.9%, 45.2% and 36.8% in recall, precision and F-measure, respectively. It takes over Yao's method by 3.5% in total F-measure. VGG-synth-coco also performs much better than the VGG-synth model, demonstrating the introduction of real text images helps a lot in training better character models.

Fig. 7 illustrates some detection samples from the ICDAR13, ICDAR15 and COCO-Text test images. By exploiting rich word annotations from real text image datasets, our model becomes more robust and can thus successfully detect various challenging texts, e.g. blurry, perspective distortion, handwritten/art fonts, extreme illumination and etc., which are hard to be synthesized using computers.

Computational Time For a 500×500 image, the character network takes about 500ms using an Nvidia Tesla K40 GPU. The text line generation and word partition procedures together take about 20ms using a 2GHz CPU.

3.5. Applied to Various Scenarios

We apply our pipeline to various challenging scenarios, including advertising images, deformed document texts and math expressions. A character model is trained by a privately collected text image datasets about these scenarios,

Figure 7: Sample qualitative results using the VGG16-synth model (top) and the models trained by word supervision (bottom) on the benchmark scene text datasets. Yellow and red rectangles illustrate the correctly and wrongly detected text lines, respectively.

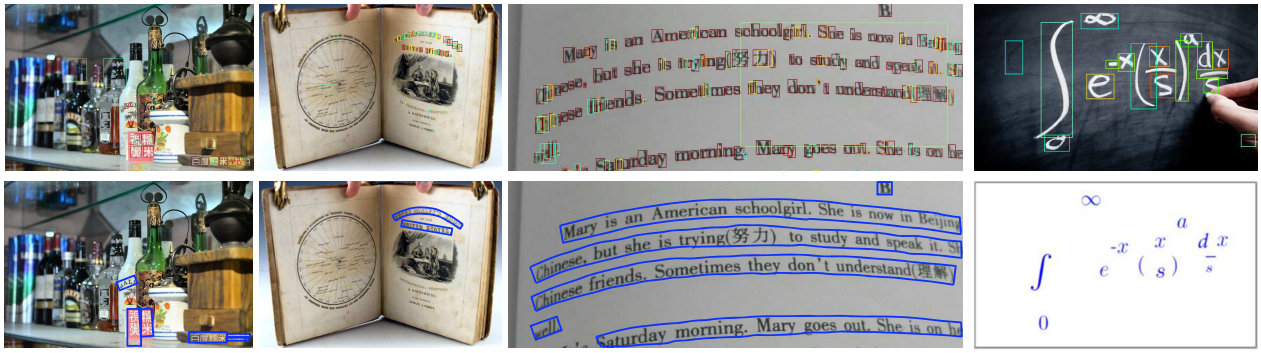

Figure 8: Applied to various scenarios. The top row shows detected characters, with colors indicating character scores (indexed by *colormap* in Matlab). The bottom row shows results of structure analysis.

Method	Recall	Precision	F-measure
A [39]	23.3	83.78	36.48
B [39]	10.7	89.73	19.14
C [39]	4.7	18.56	7.47
Yao et al.[44]	27.1	43.2	33.3
our (VGG16-synth)	26.8	42.6	32.5
our (VGG16-synth-coco)	**30.9**	**45.2**	**36.8**

Table 4: Performance of different methods on the COCO-Text (%). Notice that the annotations are obtained under the participation of method A, B and C. It is thus not fair to be compared with these methods. Yet, they are listed here for reference.

consisting of $2k$ character-level annotated images and $30k$ line-level annotated images (only images with straight text lines are involved). The training approach is similar as in Section 3.2. Text lines are generated by the approach in Section 2.4. Fig. 8 illustrates the character detection (top row) and text line generation (bottom row) results on some representative images. Our approach can handle text lines with various languages and extreme deformations. It is also worth noting that Chinese has a vast number of character classes, where some of them may not be seen by the $2k$ character-level annotated images. However, we empirically found that the initial model can still help recovering center masks of many unseen characters given only text line annotations. One possible reason is that the unseen chracters may share similar substructures or strokes with the characters seen by the initial model.

We also show application for math expression recognition (see the last column of Fig. 8). Math expressions are non-sequential, and hence sequential text recognition technique is not applicable. Given detected characters, we can recognize each of them, producing a set of math symbols.

4. Conclusion

Character based text detection methods are flexible to be applied in various scenarios. We present a weakly supervised approach to enable the use of real word-level annotated text images for training. We show the representation power of character models can be significantly strengthened. Extensive experiments demonstrate the effectiveness of our weakly supervised approach and the flexibility of our text detection pipeline.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

[2] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE TPAMI*, 11(6):567–585, 1989.

[3] H. Chen, S. S. Tsai, G. Schroth, D. M. Chen, R. Grzeszczuk, and B. Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *ICIP*, pages 2609–2612, 2011.

[4] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1635–1643, 2015.

[5] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *CVPR*, pages 2963–2970, 2010.

[6] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

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

[8] T. He, W. Huang, Y. Qiao, and J. Yao. Accurate text localization in natural image with cascaded convolutional text network. *CoRR, abs/1603.09423*, 2016.

[9] T. He, W. Huang, Y. Qiao, and J. Yao. Text-attentional convolutional neural network for scene text detection. *IEEE TIP*, 25(6):2529–2541, 2016.

[10] W. He, Y. Luo, F. Yin, H. Hu, J. Han, E. Ding, and C.-L. Liu. Context-aware mathematical expression recognition: An end-to-end framework and a benchmark. In *ICPR*, 2016.

[11] W. He, Y. Luo, F. Yin, H. Hu, J. Han, E. Ding, and C. L. Liu. Context-aware mathematical expression recognition: An end-to-end framework and a benchmark. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, 2017.

[12] L. Huang, Y. Yang, Y. Deng, and Y. Yu. DenseBox: Unifying landmark localization with end to end object detection. *CoRR*, abs/1509.04874, 2015.

[13] W. Huang, Z. Lin, J. Yang, and J. Wang. Text localization in natural images using stroke feature transform and text covariance descriptors. In *ICCV*, pages 1241–1248, 2013.

[14] W. Huang, Y. Qiao, and X. Tang. Robust scene text detection with convolution neural network induced mser trees. In *ECCV*, pages 497–511, 2014.

[15] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *ArXiv e-prints*, 2014.

[16] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *IJCV*, 116(1):1–20, 2016.

[17] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *ECCV*, pages 512–528, 2014.

[18] L. Kang, Y. Li, and D. Doermann. Orientation robust text line detection in natural images. In *CVPR*, pages 4034–4041, 2014.

[19] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. Icdar 2015 competition on robust reading. In *ICDAR*, pages 1156–1160. IEEE, 2015.

[20] 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 *ICDAR*, pages 1484–1493. IEEE, 2013.

[21] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: towards accurate region proposal generation and joint object detection. In *CVPR*, pages 845–853, 2016.

[22] Y. Li, W. Jia, C. Shen, and A. van den Hengel. Characterness: An indicator of text in the wild. *IEEE TIP*, 23(4):1666–1677, 2014.

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

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *ECCV*, pages 21–37, 2016.

[25] Y. Liu and L. Jin. Deep matching prior network: Toward tighter multi-oriented text detection. *arXiv preprint arXiv:1703.01425*, 2017.

[26] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[27] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue. Arbitrary-oriented scene text detection via rotation proposals. In *CoRR, abs/1603.09423*, 2017.

[28] S. Mao, A. Rosenfeld, and T. Kanungo. Document structure analysis algorithms: a literature survey. In *Electronic Imaging 2003*, pages 197–207. International Society for Optics and Photonics, 2003.

[29] G. Meng, Z. Huang, Y. Song, S. Xiang, and C. Pan. Extraction of virtual baselines from distorted document images using curvilinear projection. In *ICCV*, pages 3925–3933, 2015.

[30] A. Mishra, K. Alahari, and C. Jawahar. Scene text recognition using higher order language priors. In *BMVC*. BMVA, 2012.

[31] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[32] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[34] H. F. Schantz. *History of OCR, optical character recognition*. Recognition Technologies Users Association, 1982.

[35] B. Shi, X. Bai, and S. Belongie. Detecting oriented text in natural images by linking segments. *arXiv preprint arXiv:1703.06520*, 2017.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[37] S. Tian, Y. Pan, C. Huang, S. Lu, K. Yu, and C. Lim Tan. Text flow: A unified text detection system in natural scene images. In *ICCV*, pages 4651–4659, 2015.

[38] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *ECCV*, pages 56–72, 2016.

[39] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. In *arXiv preprint arXiv:1601.07140*, 2016.

[40] K. Wang, B. Babenko, and S. Belongie. End-to-End scene text recognition. In *ICCV*, pages 1457–1464, 2011.

[41] K. Wang and S. Belongie. Word spotting in the wild. In *ECCV*, pages 591–604, 2010.

[42] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-End text recognition with convolutional neural networks. In *ICPR*, pages 3304–3308, 2012.

[43] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *CVPR*, pages 1083–1090, 2012.

[44] C. Yao, X. Bai, N. Sang, X. Zhou, S. Zhou, and Z. Cao. Scene text detection via holistic, multi-channel prediction. *CoRR, abs/1606.09002*, 2016.

[45] F. Yin and C. Liu. Handwritten chinese text line segmentation by clustering with distance metric learning. *Pattern Recognition*, 42(12):3146–3157, 2009.

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

[47] X.-C. Yin, X. Yin, K. Huang, and H.-W. Hao. Robust text detection in natural scene images. *IEEE TPAMI*, 36(5):970–983, 2014.

[48] Z. Zhang, W. Shen, C. Yao, and X. Bai. Symmetry-based text line detection in natural scenes. In *CVPR*, pages 2558–2567, 2015.

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

[50] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: An efficient and accurate scene text detector. *arXiv preprint arXiv:1704.03155*, 2017.

[51] S. Zhu and R. Zanibbi. A text detection system for natural scenes with convolutional feature learning and cascaded classification. In *CVPR*, pages 625–632, 2016.