Towards End-to-end Text Spotting with Convolutional Recurrent Neural Networks

Hui Li[1*], Peng Wang[2,1*], Chunhua Shen[1]
[1]The University of Adelaide, and Australian Centre for Robotic Vision
[2]Northwestern Polytechnical University, China
e-mail: {hui.li02,chunhua.shen}@adelaide.edu.au, peng.wang@nwpu.edu.cn

Abstract

In this work, we jointly address the problem of text detection and recognition in natural scene images based on convolutional recurrent neural networks. We propose a unified network that simultaneously localizes and recognizes text with a single forward pass, avoiding intermediate processes, such as image cropping, feature re-calculation, word separation, and character grouping. In contrast to existing approaches that consider text detection and recognition as two distinct tasks and tackle them one by one, the proposed framework settles these two tasks concurrently. The whole framework can be trained end-to-end, requiring only images, ground-truth bounding boxes and text labels. The convolutional features are calculated only once and shared by both detection and recognition, which saves processing time. Through multi-task training, the learned features become more informative and improves the overall performance. Our proposed method has achieved competitive performance on several benchmark datasets.

1. Introduction

Text in natural scene images contains rich semantic information and is of great value for image understanding. As an important task in image analysis, scene text spotting, including both text region detection and word recognition, attracts much attention in computer vision field.

Due to the large variation in text patterns and background, spotting text in natural scene is much more challenging than in scanned documents. Although significant progress has been made recently based on Deep Neural Network (DNN) techniques, it is still an open problem [36].

Previous works [28, 2, 12, 11] usually divide text spotting into a sequence of distinct sub-tasks. Text detection is performed firstly with a high recall to get candidate text regions. Then word recognition is performed on cropped bounding boxes using different approaches, followed by word separation or character grouping. A number of techniques are also developed which solely focus on text detection or word recognition. However, the tasks of word detection and recognition are highly correlated. Firstly, the feature information can be shared between them. In addition, these two tasks can complement each other: detecting text regions accurately helps improve recognition performance, and recognition outputs can be used to refine detection results.

To this end, we propose an end-to-end trainable text spotter, which jointly detects and recognizes words in natural scene images. An overview of the network architecture is presented in Figure 1. It consists of several convolutional layers, a region proposal network tailored specifically for text (refer to as Text Proposal Network, TPN), a Recurrent Neural Network (RNN) encoder for embedding proposals of varying sizes to fixed-length vectors, multi-layer perceptrons for detection and bounding box regression, and an attention-based RNN decoder for word recognition. Via this framework, both text bounding boxes and word labels are provided with a single forward evaluation of the network. We do not need to process the intermediate issues such as character grouping [35, 26] or text line separation [32], and thus avoid potential error accumulation. The main contributions are thus three-fold:

(1) An end-to-end trainable DNN is designed to optimize the overall accuracy and share computations. The network integrates both text detection and word recognition. With the simultaneous training on multiple tasks, the learned features are more informative, which can promote the detection results as well as the overall performance. The convolutional features are shared by both detection and recognition, which saves processing time. To our best knowledge, this is the first attempt to integrate text detection and recognition into a single end-to-end trainable network.

(2) We propose a new method for region feature extraction. In previous works [4, 21], Region-of-Interest (RoI)

*The first two authors equally contributed to this work. C. Shen is the corresponding author.

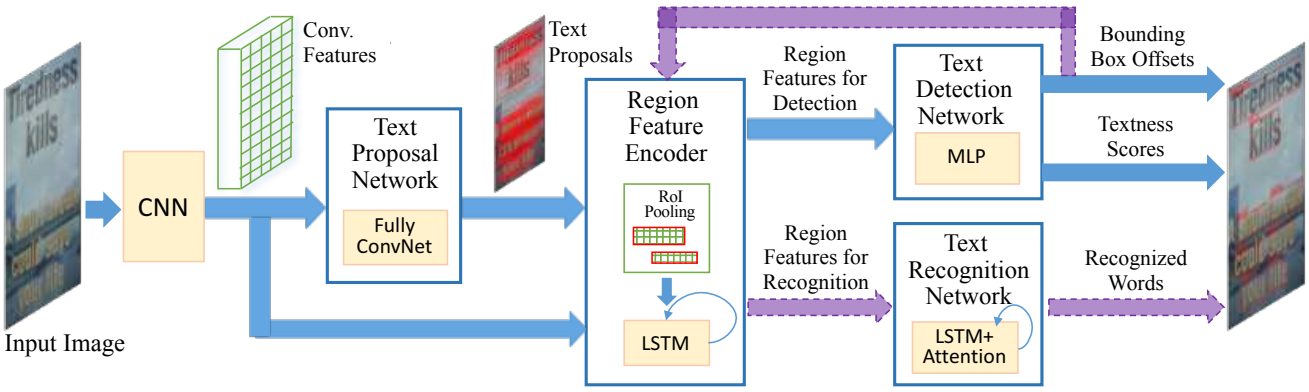

Figure 1. Model overview. The network takes an image as input, and outputs both text bounding boxes and text labels in one forward pass. The whole network is trained end-to-end.

pooling layer converts regions of different sizes and aspect ratios into feature maps with a fixed size. Considering the significant diversity of aspect ratios in text bounding boxes, it is sub-optimal to fix the size after pooling. To accommodate the original aspect ratios and avoid distortion, RoI pooling is tailored to generate feature maps with varying lengths. An RNN encoder is then employed to encode feature maps of different lengths into the same size.

(3) A curriculum learning strategy is designed to train the system with gradually more complex training data. Starting from synthetic images with simple appearance and a large word lexicon, the system learns a character-level language model and finds a good initialization of appearance model. By employing real-world images with a small lexicon later, the system gradually learns how to handle complex appearance patterns. We conduct a set of experiments to explore the capabilities of different model structures. The best model outperforms state-of-the-art results on several standard benchmarks, including ICDAR2011 [22] and ICDAR2015 [14].

2. Related Work

Text spotting essentially includes two tasks: text detection and word recognition. In this section, we present a brief introduction to related works on text detection, word recognition, and text spotting systems that combine both. There are comprehensive surveys for text detection and recognition in [30, 36].

Text Detection Text detection aims to localize text in images and generate bounding boxes for words. Existing approaches can be roughly classified into three categories: character based, text-line based and word based methods.

Character based methods firstly find characters in images, and then group them into words. They can be further divided into sliding window based [12, 29, 35, 26] and Connected Components (CC) based [9, 20, 3] methods. Sliding window based approaches use a trained classifier to detect characters across the image in a multi-scale sliding window fashion. CC based methods segment pixels with consistent region properties (*i.e.*, color, stroke width, density, *etc.*) into characters. The detected characters are further grouped into text regions by morphological operations, conditional random fields or other graph models.

Text-line based methods detect text lines firstly and then separate each line into multiple words. The motivation is that people usually distinguish text regions initially even if characters are not recognized. Based on the observation that a text region usually exhibits high self-similarity to itself and strong contrast to its local background, Zhang *et al*. [32] propose to extract text lines by exploiting symmetry property. Zhang *et al*. [33] localize text lines via salient maps that are calculated by fully convolutional networks. Post-processing techniques are also proposed in [33] to extract text lines in multiple orientations.

More recently, a number of approaches are proposed to detect words directly using DNN based techniques, such as Faster R-CNN [21], YOLO [13], SSD [18]. By extending Faster R-CNN, Zhong *et al*. [34] design a text detector with a multi-scale Region Proposal Network (RPN) and a multi-level RoI pooling layer. Tian *et al*. [27] develop a vertical anchor mechanism, and propose a Connectionist Text Proposal Network (CTPN) to accurately localize text lines in images. Gupta *et al*. [6] use a Fully-Convolutional Regression Network (FCRN) for efficient text detection and bounding box regression, motivated by YOLO. Similar to SSD, Liao *et al*. [17] propose "TextBoxes" by combining predictions from multiple feature maps with different resolutions, and achieve the best-reported text detection performance on datasets in [14, 28].

Text Recognition Traditional approaches to text recognition usually perform in a bottom-up fashion, which recognize individual characters firstly and then integrate them

into words by means of beam search [2], dynamic programming [12], *etc*. In contrast, Jaderberg *et al*. [10] consider word recognition as a multi-class classification problem, and categorize each word over a large dictionary (about 90K words) using a deep convolutional neural network (CNN).

With the success of RNNs on handwriting recognition [5], He *et al*. [7] and Shi *et al*. [23] treat word recognition as a sequence labeling problem. RNNs are employed to generate sequential labels of arbitrary length without character segmentation, and Connectionist Temporal Classification (CTC) is adopted to decode the sequence. Lee and Osindero [16] and Shi *et al*. [24] propose to recognize text using an attention-based sequence-to-sequence learning structure. In this manner, RNNs automatically learn the character-level language model presented in word strings from the training data. The soft-attention mechanism allows the model to selectively exploit local image features. These networks can be trained end-to-end with cropped word patches as inputs. Moreover, Shi *et al*. [24] insert a Spatial Transformer Network (STN) to handle words with irregular shapes.

Text Spotting Systems Text spotting needs to handle both text detection and word recognition. Wang *et al*. [28] take the locations and scores of detected characters as input and try to find an optimal configuration of a particular word in a given lexicon, based on a pictorial structures formulation. Neumann and Matas [20] use a CC based method for character detection. These characters are then agglomerated into text lines based on heuristic rules. Optimal sequences are finally found in each text line using dynamic programming, which are the recognized words. These recognition-based pipelines lack explicit word detection.

Some text spotting systems firstly generate text proposals with a high recall and a low precision, and then refine them using a separate recognition model. It is expected that a strong recognizer can reject false positives, especially when a lexicon is given. Jaderberg *et al*. [11] use an ensemble model to generate text proposals, and then adopt the word classifier in [10] for recognition. Gupta *et al*. [6] employ FCRN for text detection and the word classifier in [10] for recognition. Liao *et al*. [17] combine "TextBoxes" and "CRNN" [23], which yield state-of-the-art text spotting performance on datasets in [14, 28].

3. Model

Our goal is to design an end-to-end trainable network, which simultaneously detects and recognizes all words in images. Our model is motivated by recent progresses in DNN models such as Faster R-CNN [21] and sequence-to-sequence learning [24, 16], but we take the special characteristics of text into consideration. In this section, we present a detailed description of the whole system.

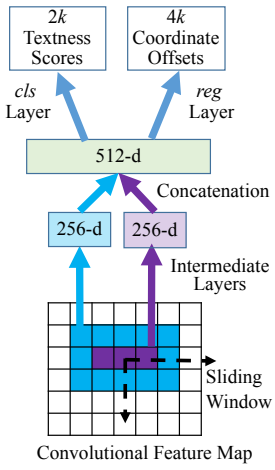

Figure 2. Text Proposal Network (TPN). We apply multiple scale sliding windows over the convolutional feature maps. Both local and contextual information are retained in order to propose high quality text bounding boxes. The concatenated local and contextual features are further fed into the *cls* layer for computing textness scores and the *reg* layer to calculate coordinate offsets, with respect to k anchors at each position.

Notation All bold capital letters represent matrices and all bold lower-case letters denote column vectors. $[\mathbf{a}; \mathbf{b}]$ concatenates the vectors $\mathbf{a}$ and $\mathbf{b}$ vertically, while $[\mathbf{a}, \mathbf{b}]$ stacks $\mathbf{a}$ and $\mathbf{b}$ horizontally (column wise). In the following, the bias terms in neural networks are omitted.

3.1. Overall Architecture

The whole system architecture is illustrated in Figure 1. Firstly, the input image is fed into a CNN that is modified from the VGG-16 net [25]. The original VGG-16 net consists of 13 layers of 3×3 convolutions followed by Rectified Linear Unit (ReLU), 5 layers of 2×2 max-pooling, and Fully-Connected (FC) layers. Here we remove FC layers. As long as text in images can be relatively small, we only keep the 1st, 2nd and 4th max-pooling layers, so that the down-sampling ratio is increased from $1/32$ to $1/8$.

Given the computed convolutional features, TPN provides a list of text region proposals (bounding boxes). Then, Region Feature Encoder (RFE) converts the convolutional features of proposals into fixed-length representations. These representations are further fed into Text Detection Network (TDN) to calculate their textness scores and bounding box offsets. Next, RFE is applied again to compute fixed-length representations of text bounding boxes provided by TDN (see purple paths in Figure 1). Finally, Text Recognition Network (TRN) recognizes words in the detected bounding boxes based on their encoded representations.

3.2. Text Proposal Network

Text proposal network (TPN) is inspired from RPN [21, 34], which can be regarded as a fully convolutional network. As presented in Figures 2, it takes convolutional features as input, and outputs a set of bounding boxes accompanied with "textness" scores and coordinate offsets, which indicate scale-invariant translations and log-space height/width shifts relative to pre-defined anchors as in [21].

Considering that word bounding boxes usually have larger aspect ratios (W/H) and varying scales, we designed $k = 24$ anchors with 4 scales (with box areas of 16^2, 32^2, 64^2, 80^2) and 6 aspect ratios ($W/H = 1, 2, 3, 5, 7, 10$).

Inspired by [34], we apply two 256-d rectangle convolutional filters of different sizes ($W = 5, H = 3$ and $W = 3, H = 1$) on feature maps to extract both local and contextual information. The rectangle filters lead to wider receptive fields, which is more suitable for word bounding boxes with large aspect ratios. The resulting features are further concatenated to 512-d vectors and fed into two sibling layers for text/non-text classification and bounding box regression.

3.3. Region Feature Encoder

To process RoIs of different scales and aspect ratios in a unified way, most existing works re-sample regions into *fixed-size* feature maps via pooling [4]. However, for text, this approach may lead to significant distortion due to the large variation of word lengths. For example, it may be unreasonable to encode short words like "Dr" and long words like "congratulations" into feature maps of the same size. In this work, we propose to re-sample regions according to their respective aspect ratios, and then use RNNs to encode the resulting feature maps of different lengths into fixed length vectors. The whole region feature encoding process is illustrated in Figure 3.

For an RoI of size $h \times w$, we perform spatial max-pooling with a resulting size of

$$H \times \min(W_{max}, 2Hw/h), \tag{1}$$

where the expected height H is fixed and the width is adjusted to keep the aspect ratio as $2w/h$ (twice the original aspect ratio) unless it exceeds the maximum length W_{max}. Note that here we employ a pooling window with an aspect ratio of $1:2$, which benefits the recognition of narrow shaped characters, like 'i', 'l', etc., as stated in [23].

Next, the re-sampled feature maps are considered as a sequence and fed into RNNs for encoding. Here we use Long-Short Term Memory (LSTM) [8] instead of vanilla RNN to overcome the shortcoming of gradient vanishing or exploding. The feature maps after the above varying-size RoI pooling are denoted as $\mathbf{Q} \in \mathbb{R}^{C \times H \times W}$, where $W = \min(W_{max}, 2Hw/h)$ is the number of columns and C is the channel size. We flatten the features in each column, and obtain a sequence $\mathbf{q}_1, \ldots, \mathbf{q}_W \in \mathbb{R}^{C \times H}$ which are fed into LSTMs one by one. Each time LSTM units receive one column of feature $\mathbf{q}_t$, and update their hidden state $\mathbf{h}_t$ by a non-linear function: $\mathbf{h}_t = \mathrm{f}(\mathbf{q}_t, \mathbf{h}_{t-1})$. In this recurrent fashion, the final hidden state $\mathbf{h}_W$ (with size $R = 1024$) captures the holistic information of $\mathbf{Q}$ and is used as an RoI representation with fixed dimension.

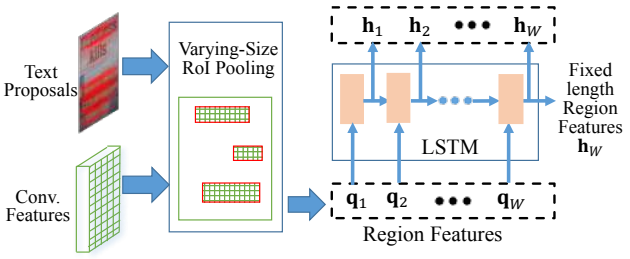

Figure 3. Region Features Encoder (RFE). Region features after RoI pooling are not required to be of the same size. In contrast, they are calculated according to the specific aspect ratios of bounding boxes, with height normalized. LSTM is then employed to encode region features of varying length into fixed-size representations.

3.4. Text Detection and Recognition

Text Detection Network (TDN) aims to judge whether the proposed RoIs are text or not and refine the coordinates of bounding boxes once again, based on the extracted region features $\mathbf{h}_W$. Two fully-connected layers with 2048 neurons are applied on $\mathbf{h}_W$, followed by two parallel layers for classification and bounding box regression respectively.

The classification and regression layers used in TDN are similar to those used in TPN. Note that the whole system refines the coordinates of text bounding boxes twice: once in TPN and then in TDN. Although RFE is employed twice to calculate features for proposals produced by TPN and later the detected bounding boxes provided by TDN, the convolutional features only need to be computed once.

Text Recognition Network (TRN) aims to predict the text in the detected bounding boxes based on the extracted region features. As shown in Figure 4, we adopt LSTMs with attention mechanism [19, 24] to decode the sequential features into words.

Firstly, hidden states at all steps $\mathbf{h}_1, \ldots, \mathbf{h}_W$ from RFE are fed into an additional layer of LSTM encoder with 1024 units. We record the hidden state at each time step and form a sequence of $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_W] \in \mathbb{R}^{R \times W}$. It includes local information at each time step and works as the context for the attention model.

As for decoder LSTMs, the ground-truth word label is adopted as input during training. It can be regarded as a sequence of tokens $\mathbf{s} = \{s_0, s_1, \ldots, s_{T+1}\}$ where s_0 and s_{T+1} represent the special tokens START and END respectively. We feed decoder LSTMs with $T + 2$ vectors: $\mathbf{x}_0$, $\mathbf{x}_1, \ldots, \mathbf{x}_{T+1}$, where $\mathbf{x}_0 = [\mathbf{v}_W; \mathrm{Atten}(\mathbf{V}, \mathbf{0})]$ is the concatenation of the encoder's last hidden state $\mathbf{v}_W$ and the attention output with the guidance signal equals to zero; and $\mathbf{x}_i = [\psi(s_{i-1}); \mathrm{Atten}(\mathbf{V}, \mathbf{h}'_{i-1})]$, for $i = 1, \ldots, T + 1$, is made up of the embedding $\psi()$ of the $(i-1)$-th token s_{i-1} and the attention output guided by the hidden state of decoder LSTMs in the previous time-step $\mathbf{h}'_{i-1}$. The embedding function $\psi()$ is defined as a linear layer followed

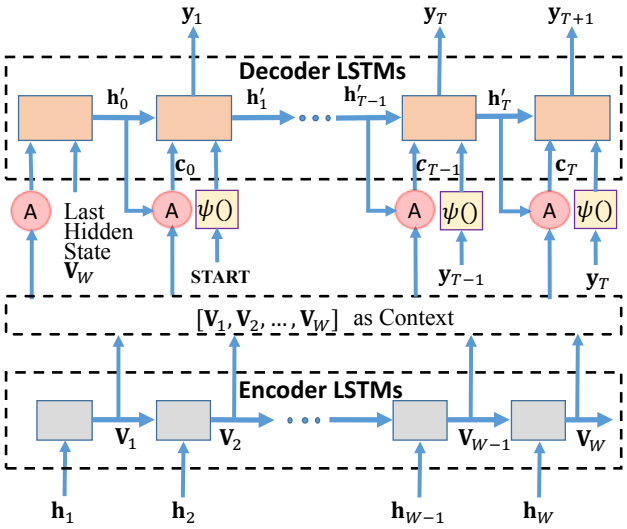

Figure 4. Text Recognition Network (TRN). The region features are encoded by one layer of LSTMs, and then decoded in an attention based sequence to sequence manner. Hidden states of encoder at all time steps are reserved and used as context for attention model.

by a tanh non-linearity.

The attention function $\mathbf{c} = \mathrm{Atten}(\mathbf{V}, \mathbf{h})$ is defined as follows:

$$\begin{cases} \mathbf{g}_j = \tanh(\mathbf{W}_v \mathbf{v}_j + \mathbf{W}_h \mathbf{h}), \ j = 1, \ldots, W, \\ \boldsymbol{\alpha} = \mathrm{softmax}(\mathbf{w}_g^\top \cdot [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_W]), \\ \mathbf{c} = \sum_{j=1}^{W} \alpha_j \mathbf{v}_j, \end{cases} \tag{2}$$

where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_W]$ is the varying-length sequence of features to be attended (context), $\mathbf{h}$ is the guidance signal, $\mathbf{W}_v$ and $\mathbf{W}_h$ are linear embedding weights to be learned, $\boldsymbol{\alpha}$ is the attention weights of size W, and the attended feature $\mathbf{c}$ is a weighted sum of input features.

At each time-step $t = 0, 1, \ldots, T+1$, the decoder LSTMs compute their hidden state $\mathbf{h}'_t$ and output vector $\mathbf{y}_t$ as follows:

$$\begin{cases} \mathbf{h}'_t = \mathrm{f}(\mathbf{x}_t, \mathbf{h}'_{t-1}), \\ \mathbf{y}_t = \varphi(\mathbf{h}'_t) = \mathrm{softmax}(\mathbf{W}_o \mathbf{h}'_t) \end{cases} \tag{3}$$

where the LSTM [8] is used for the recurrence formula $\mathrm{f}()$, and $\mathbf{W}_o$ linearly transforms hidden states to the output space of size 38, including 26 case-insensitive characters, 10 digits, a token representing all punctuations like "!" and "?", and a special END token.

At test time, the token with the highest probability in previous output $\mathbf{y}_t$ is selected as the input token at step $t+1$, instead of the ground-truth tokens $s_1, \ldots, s_T$. The process is started with the START token, and repeated until we get the special END token.

3.5. Loss Functions and Training

Loss Functions As we demonstrate above, our system takes as input an image, word bounding boxes and their labels during training. Both TPN and TDN employ the binary logistic loss L_{cls} for classification, and smooth L_1 loss L_{reg} [21] for regression. So the loss for training TPN is

$$L_{TPN} = \frac{1}{N} \sum_{i=1}^{N} L_{cls}(p_i, p_i^\star) + \frac{1}{N_+} \sum_{i=1}^{N_+} L_{reg}(\mathbf{d}_i, \mathbf{d}_i^\star), \tag{4}$$

where N is the number of randomly sampled anchors in a mini-batch and N_+ is the number of positive anchors in this batch (the range of positive anchor indices is from 1 to N_+). The mini-batch sampling and training process of TPN are similar to that used in [21]. An anchor is considered as positive if its Intersection-over-Union (IoU) ratio with any ground-truth is greater than 0.7 and considered as negative if its IoUs with all ground-truth are smaller than 0.3. In this paper, N is set to 256 and N_+ is at most 128. p_i denotes the predicted probability of anchor i being text and $p_i^\star$ is the corresponding ground-truth label (1 for text, 0 for non-text). $\mathbf{d}_i$ is the predicted coordinate offsets $(\mathrm{dx}_i, \mathrm{dy}_i, \mathrm{dw}_i, \mathrm{dh}_i)$ for anchor i, and $\mathbf{d}_i^\star$ is the associated offsets for anchor i relative to the ground-truth. Bounding box regression only applies to positive anchors.

For the final outputs of the whole system, we apply a multi-task loss for both detection and recognition:

$$\begin{aligned} L_{DRN} = &\frac{1}{\hat{N}} \sum_{i=1}^{\hat{N}} L_{cls}(\hat{p}_i, \hat{p}_i^\star) + \frac{1}{\hat{N}_+} \sum_{i=1}^{\hat{N}_+} L_{reg}(\hat{\mathbf{d}}_i, \hat{\mathbf{d}}_i^\star) \\ &+ \frac{1}{\hat{N}_+} \sum_{i=1}^{\hat{N}_+} L_{rec}(\mathbf{Y}^{(i)}, \mathbf{s}^{(i)}) \end{aligned} \tag{5}$$

where $\hat{N} = 128$ is the number of text proposals sampled from the output of TPN, and $\hat{N}_+ \leq 64$ is the number of positive ones. The IoU ratio thresholds for positive and negative anchors are set to 0.6 and 0.4 respectively, which are less strict than those used for training TPN. In order to mine hard negatives, we first apply TDN on 1000 randomly sampled negatives and select those with higher textness scores. $\hat{p}_i$ and $\hat{\mathbf{d}}_i$ are the outputs of TDN. $\mathbf{s}^{(i)}$ is the ground-truth tokens for sample i and $\mathbf{Y}^{(i)} = \{\mathbf{y}_0^{(i)}, \mathbf{y}_1^{(i)}, \ldots, \mathbf{y}_{T+1}^{(i)}\}$ is the corresponding output sequence of decoder LSTMs. $L_{rec}(\mathbf{Y}, \mathbf{s}) = -\sum_{t=1}^{T+1} \log \mathbf{y}_t(s_t)$ denotes the cross entropy loss on $\mathbf{y}_1, \ldots, \mathbf{y}_{T+1}$, where $\mathbf{y}_t(s_t)$ represents the predicted probability of the output being s_t at time-step t and the loss on $\mathbf{y}_0$ is ignored.

Following [21], we use an approximate joint training process to minimize the above two losses together (ADAM [15] is adopted), ignoring the derivatives with respect to the proposed boxes' coordinates.

Data Augmentation We sample one image per iteration in the training phase. Training images are resized to shorter side of 600 pixels and longer side of at most 1200 pixels. Data augmentation is also implemented to improve the robustness of our model, which includes:

1) randomly rescaling the width of the image by ratio 1 or 0.8 without changing its height, so that the bounding boxes have more variable aspect ratios;

2) randomly cropping a sub-image which includes all text in the original image, padding with 100 pixels on each side, and resizing to 600 pixels on shorter side.

Curriculum Learning In order to improve generalization and accelerate the convergence speed, we design a curriculum learning [1] paradigm to train the model from gradually more complex data.

1) We generate 48k images containing words in the "Generic" lexicon [11] of size 90k by using the synthetic engine proposed in [6]. The words are randomly placed on simple *pure color backgrounds* (10 words per image on average). We lock TRN initially, and train the rest parts of our proposed model on these synthetic images in the first 30k iterations, with convolutional layers initialized from the trained VGG-16 model and other parameters randomly initialized according to Gaussian distribution. For efficiency, the first four convolutional layers are fixed during the entire training process. The learning rate is set to 10^{-5} for parameters in the rest of convolutional layers and 10^{-3} for randomly initialized parameters.

2) In the next 30k iterations, TRN is added and trained with a learning rate of 10^{-3}, together with other parts in which the learning rate for randomly initialized parameters is halved to 5×10^{-4}. We still use the 48k synthetic images as they contain a comprehensive 90k word vocabulary. With this synthetic dataset, a character-level language model can be learned by TRN.

3) In the next 50k iterations, the training examples are randomly selected from the "Synth800k" [6] dataset, which consists of 800k images with averagely 10 synthetic words placed on each *real scene background*. The learning rate for convolutional layers remains at 10^{-5}, but that for others is halved to 10^{-4}.

4) Totally 2044 *real-world* training images from ICDAR2015 [14], SVT [28] and AddF2k [34] datasets are employed for another 20k iterations. In this stage, all the convolutional layers are fixed and the learning rate for others is further halved to 10^{-5}. These real images contain much less words than synthetic ones, but their appearance patterns are much more complex.

4. Experiments

In this section, we perform experiments to verify the effectiveness of the proposed method. All experiments are implemented on an NVIDIA Tesla M40 GPU with 24GB memory. We rescale the input image into multiple sizes during test phase in order to cover the large range of bounding box scales, and sample 300 proposals with the highest textness scores produced by TPN. The detected bounding boxes are then merged via Non-Maximum Suppression (NMS) according to their textness scores and fed into TRN for recognition.

Criteria We follow the evaluation protocols used in ICDAR2015 Robust Reading Competition [14]: a bounding box is considered as correct if its IoU ratio with any ground-truth is greater than 0.5 and the recognized word also matches, ignoring case. The words that contain alphanumeric characters and no longer than three characters are ignored. There are two evaluation protocols used in the task of scene text spotting: "End-to-End" and "Word Spotting". "End-to-End" protocol requires that all words in the image are to be recognized, with independence of whether the string exists or not in the provided contextualized lexicon, while "Word Spotting" on the other hand, only looks at the words that actually exist in the lexicon provided, ignoring all the rest that do not appear in the lexicon.

Datasets The commonly used datasets for scene text spotting include ICDAR2015 [14], ICDAR2011 [22] and Street View Text (SVT) [28]. The dataset for the task of "Focused Scene Text" in ICDAR2015 Robust Reading Competition, consists of 229 images for training and 233 images for test. In addition, it provides 3 specific lists of words as lexicons for reference in the test phase, *i.e.*, "Strong", "Weak" and "Generic". "Strong" lexicon provides 100 words per-image including all words appeared in the image. "Weak" lexicon contains all words appeared in the entire dataset, and "Generic" lexicon is a 90k word vocabulary proposed by [11]. ICDAR2011 does not provide any lexicon, so we only use the 90k vocabulary. SVT dataset consists of 100 images for training and 249 images for test. These images are harvested from Google Street View and often have a low resolution. It also provides a "Strong" lexicon with 50 words per-image. As there are unlabeled words in SVT, we only evaluate the "Word-Spotting" performance on this dataset.

4.1. Evaluation under Different Model Settings

In order to show the effectiveness of our proposed varying-size RoI pooling (see Section 3.3) and the attention mechanism (see Section 3.4), we examine the performance of our model with different settings in this subsection. With the fixed RoI pooling size of 4×20, we denote the models with and without the attention mechanism as "Ours Atten+Fixed" and "Ours NoAtten+Fixed" respectively. The model with both attention and varying-size RoI pooling is denoted as "Ours Atten+Vary", in which the size of feature maps after pooling is calculated by Equ. (1) with $H = 4$ and $W_{max} = 35$.

Table 1. Text spotting results on different benchmarks. We present the F-measure here in percentage. "Ours Two-stage" uses separate models for detection and recognition, while other "Ours" models are end-to-end trained. "Ours Atten+Vary" achieves the best performance on almost all datasets.

Method	ICDAR2015 Word-Spotting			ICDAR2015 End-to-End			ICDAR2011 Word-Spotting	SVT Word-Spotting	
	Strong	Weak	Generic	Strong	Weak	Generic	Generic	Strong	Generic
Deep2Text II+ [31]	84.84	83.43	78.90	81.81	79.47	76.99	–	–	–
Jaderberg *et al*. [11]	90.49	–	76	86.35	–	–	76	76	53
FCRNall+multi-filt [6]	–	–	84.7	–	–	–	84.3	67.7	55.7
TextBoxes [17]	93.90	91.95	85.92	**91.57**	89.65	83.89	87	84	64
YunosRobot1.0	86.78	–	86.78	84.20	–	84.20	–	–	–
Ours Two-stage	92.94	90.54	84.24	88.20	86.06	81.97	82.86	82.19	62.35
Ours NoAtten+Fixed	92.70	90.37	83.83	87.73	85.53	79.18	81.70	79.49	58.70
Ours Atten+Fixed	93.33	91.66	87.73	90.72	87.86	83.98	83.81	81.80	64.50
Ours Atten+Vary	**94.16**	**92.42**	**88.20**	91.08	**89.81**	**84.59**	**87.70**	**84.91**	**66.18**

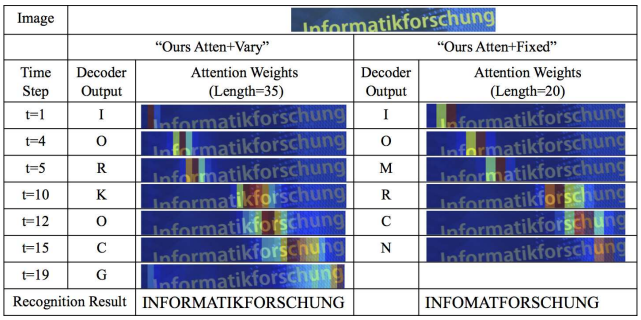

Figure 5. The sequence decoding process performed by "Ours Atten+Vary" and "Ours Atten+Fixed". The heat maps show that at each time step, the position of the character being decoded has a higher attention weight, so that the corresponding local features will be extracted and assist text recognition. However, if we use the fixed size RoI pooling ("Ours Atten+Fixed"), information may be lost during pooling, especially for a long word, which can lead to an incorrect recognition result. In contrast, the varying-size RoI pooling ("Ours Atten+Vary") preserves more information and leads to a correct result.

Although the last hidden state of LSTMs encodes the holistic information of RoI image patch, it still lacks details. Particularly for a long word image patch, the initial information may be lost during the recurrent encoding process. Thus, we keep the hidden states of encoder LSTMs at each time step as context. The attention model can choose the corresponding local features for each character during decoding process, as illustrated in Figure 5. From Table 1, we can see that the model with attention mechanism, namely "Ours Atten+Fixed", achieves higher F-measures on all evaluated data than "Ours NoAtten+Fixed" which does not use attention.

One contribution of this work is a new region feature encoder, which is composed of a varying-size RoI pooling mechanism and an LSTM sequence encoder. To validate its effectiveness, we compare the performance of models "Ours Atten+Vary" and "Ours Atten+Fixed". Experiments shows that varying-size RoI pooling performs significantly better for long words. For example, "Informatikforschung" can be recognized correctly by "Ours Atten+Vary", but not by "Ours Atten+Fixed" (as shown in Figure 5), because a large amount of information for long words is lost by fixed-size RoI pooling. As illustrated in Table 1, adopting varying-size RoI pooling ("Ours Atten+Vary") instead of fixed-size pooling ("Ours Atten+Fixed") improves F-measure by around 1 percentage point for ICDAR2015, 4 points for ICDAR2011 and 3 points for SVT with strong lexicon used.

4.2. Joint Training vs. Separate Training

Previous works [11, 6, 17] on text spotting typically perform in a two-stage manner, where detection and recognition are trained and processed separately. The text bounding boxes detected by a model need to be cropped from the image and then recognized by another model. In contrast, our proposed model is trained jointly for both detection and recognition. By sharing convolutional features and RoI encoder, the knowledge learned from the correlated detection and recognition tasks can be transferred between each other and results in better performance for both tasks.

To compare with the model "Ours Atten+Vary" which is jointly trained, we build a two-stage system (denoted as "Ours Two-stage") in which detection and recognition models are trained separately. For fair comparison, the detector in "Ours Two-stage" is built by removing the recognition part from model "Ours Atten+Vary" and trained only with the detection loss (denoted as "Ours DetOnly"). As to recognition, we employ CRNN [23] that produces state-of-the-art performance on text recognition. Model "Ours Two-stage" firstly adopts "Ours DetOnly" to detect text with the same multi-scale inputs. CRNN is then applied to recognize the detected bounding boxes. We can see from Table 1 that model "Ours Two-stage" performs worse than "Ours Atten+Vary" on all the evaluated datasets.

Figure 6. Examples of text spotting results by "Ours Atten+Vary". The first two columns are images from ICDAR2015, and the rest are images from SVT. Red bounding boxes are both detected and recognized correctly. Green bounding boxes are missed words, and yellow bounding boxes are false positives. The results show that our model is able to detect and recognize words of different aspect ratios. Most missed words have small bounding boxes.

Table 2. Text detection results on different datasets. Precision (P) and Recall (R) at maximum F-measure (F) are reported in percentage. The jointly trained model ("Ours Atten+Vary") gives better detection results than the one trained with detection loss only ("Ours DetOnly").

Method	ICDAR2015			ICDAR2011		
	R	P	F	R	P	F
Jaderberg *et al*. [11]	68.0	86.7	76.2	69.2	87.5	77.2
FCRNall+multi-filt [6]	76.4	**93.8**	84.2	76.9	**94.3**	84.7
Ours DetOnly	78.5	88.9	83.4	80.0	87.5	83.5
Ours Atten+Vary	**80.5**	91.4	**85.6**	**81.7**	89.2	**85.1**

Furthermore, we also compare the detection-only performance of these two systems. Note that "Ours DetOnly" and the detection part of "Ours Atten+Vary" share the same architecture, but they are trained with different strategies: "Ours DetOnly" is optimized with only the detection loss, while "Ours Atten+Vary" is trained with a multi-task loss for both detection and recognition. In consistent with the "End-to-End" evaluation criterion, a detected bounding box is considered to be correct if its IoU ratio with any ground-truth is greater than 0.5. The detection results are presented in Table 2. Without any lexicon used, "Ours Atten+Vary" produces a detection performance with F-measures of 85.6% on ICDAR2015 and 85.1% on ICDAR2011, which are higher than "Ours DetOnly" by 2 percentage points in average. This result illustrates that detector performance can be improved via joint training.

4.3. Comparison with Other Methods

In this part, we compare the text spotting performance of "Ours Atten+Vary" with state-of-the-art approaches. As shown in Table 1, "Ours Atten+Vary" outperforms the compared methods on most of the evaluated datasets. In particular, our method shows an significant superiority when using a generic lexicon. It leads to an averagely 1.5 percentage point higher recall than the state-of-the-art TextBoxes [17], using only 3 input scales compared with 5 scales used by TextBoxes. Several text spotting examples are presented in Figure 6.

4.4. Speed

Using an M40 GPU, model "Ours Atten+Vary" takes approximately $0.9s$ to process an input image of 600×800 pixels. It takes nearly $0.45s$ to compute the convolutional features, $0.02s$ for text proposal calculation, $0.25s$ for RoI encoding, $0.01s$ for text detection and $0.15s$ for word recognition. On the other hand, model "Ours Two-stage" spends around $0.45s$ for word recognition on the same detected bounding boxes, as it needs to crop the word patches, and re-calculate the convolutional features during recognition.

5. Conclusion

In this paper we have presented an unified end-to-end trainable DNN for simultaneous text detection and recognition in natural scene images. A novel RoI encoding method has been proposed, considering the large diversity of aspect ratios of word bounding boxes. With this framework, scene text spotting can be performed efficiently and accurately in a single forward pass.

For future works, one potential direction is extending the proposed model using 2D RNNs and 2D attention mechanisms to handle oriented texts. Furthermore, small texts can be better spotted using features from multiple convolutional layers, as in [18, 17].

References

[1] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proc. Int. Conf. Mach. Learn.*, 2009.

[2] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2013.

[3] M. Busta, L. Neumann, and J. Matas. Fastext: Efficient unconstrained scene text detector. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015.

[4] R. Girshick. Fast R-CNN. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015.

[5] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proc. Int. Conf. Mach. Learn.*, 2006.

[6] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[7] P. He, W. Huang, Y. Qiao, C. C. Loy, and X. Tang. Reading scene text in deep convolutional sequences. In *Proc. National Conf. Artificial Intell.*, 2016.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[9] W. Huang, Z. Lin, J. Yang, and J. Wang. Text localization in natural images using stroke feature transform and text covariance descriptors. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2013.

[10] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *Proc. Adv. Neural Inf. Process. Syst.*, 2014.

[11] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *Int. J. Comp. Vis.*, 116(1):1–20, 2015.

[12] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *Proc. Eur. Conf. Comp. Vis.*, 2014.

[13] J.Redmon, S. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[14] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. ICDAR 2015 robust reading competition. In *Proc. Int. Conf. Doc. Anal. Recog.*, 2015.

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learn. Representations*, 2014.

[16] C.-Y. Lee and S. Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[17] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. Textboxes: A fast text detector with a single deep neural network. In *Proc. National Conf. Artificial Intell.*, 2017.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *Proc. Eur. Conf. Comp. Vis.*, 2016.

[19] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proc. Conf. Empirical Methods in Natural Language Processing*, 2015.

[20] L. Neumann and J. Matas. Scene text localization and recognition with oriented stroke detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2013.

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. Adv. Neural Inf. Process. Syst.*, 2015.

[22] A. Shahab, F. Shafait, and A. Dengel. ICDAR 2011 robust reading competition challenge 2: Reading text in scene images. In *Proc. Int. Conf. Doc. Anal. Recog.*, 2011.

[23] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2016.

[24] B. Shi, X. Wang, P. Lv, C. Yao, and X. Bai. Robust scene text recognition with automatic rectification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *Proc. Int. Conf. Learn. Representations*, 2015.

[26] S. Tian, Y. Pan, C. Huang, S. Lu, K. Yu, and C. L. Tan. Text flow: A unified text detection system in natural scene images. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015.

[27] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *Proc. Eur. Conf. Comp. Vis.*, 2016.

[28] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2011.

[29] T. Wang, D. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *Proc. IEEE Int. Conf. Patt. Recogn.*, 2012.

[30] Q. Ye and D. Doermann. Text detection and recognition in imagery: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(7):1480–1500, 2015.

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

[32] Z. Zhang, W. Shen, C. Yao, and X. Bai. Symmetry-based text line detection in natural scenes. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[33] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[34] Z. Zhong, L. Jin, S. Zhang, and Z. Feng. Deeptext: A new approach for text proposal generation and text detection in natural images. In *Proc. IEEE Int. Conf. Acoustics, Speech & Signal Processing*, 2017.

[35] S. Zhu and R. Zanibbi. A text detection system for natural scenes with convolutional feature learning and cascaded classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[36] Y. Zhu, C. Yao, and X. Bai. Scene text detection and recognition: recent advances and future trends. *Frontiers of Computer Science*, 10(1):19–36, 2016.