ASTER: An Attentional Scene Text Recognizer with Flexible Rectification

Baoguang Shi, Mingkun Yang, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai

Abstract—A challenging aspect of scene text recognition is to handle text with distortions or irregular layout. In particular, perspective text and curved text are common in natural scenes and are difficult to recognize. In this work, we introduce ASTER, an end-to-end neural network model that comprises a rectification network and a recognition network. The rectification network adaptively transforms an input image into a new one, rectifying the text in it. It is powered by a flexible Thin-Plate Spline transformation which handles a variety of text irregularities and is trained without human annotations. The recognition network is an attentional sequence-to-sequence model that predicts a character sequence directly from the rectified image. The whole model is trained end to end, requiring only images and their groundtruth text. Through extensive experiments, we verify the effectiveness of the rectification and demonstrate the state-of-the-art recognition performance of ASTER. Furthermore, we demonstrate that ASTER is a powerful component in end-to-end recognition systems, for its ability to enhance the detector.

Index Terms—Scene Text Recognition, Thin-Plate Spline, Image Transformation, Sequence-to-Sequence Learning

1 Introduction

Scene text recognition has attracted great interest from the academia and the industry in recent years owing to its importance in a wide range of applications. Despite the maturity of Optical Character Recognition (OCR) systems dedicated to document text, scene text recognition remains a challenging problem. The large variations in background, appearance, and layout pose significant challenges, which the traditional OCR methods cannot handle effectively.

Recent advances in scene text recognition are driven by the success of deep learning-based recognition models. Among them are methods that recognize text by characters using convolutional neural networks (CNN), methods that classify words with CNNs [24], [26], and methods that recognize character sequences using a combination of a CNN and a recurrent neural network (RNN) [54]. In spite of their success, these methods do not explicitly address the problem of *irregular text*, which is text that is not horizontal and frontal, has curved layout, *etc.* Instances of irregular text frequently appear in natural scenes. As exemplified in Figure 1, typical cases include oriented text, *perspective text* [49], and curved text. Designed without the invariance to such irregularities, previous methods often struggle in recognizing such text instances.

This work introduces ASTER, which stands for **A**ttentional **S**cene **TE**xt Recognizer with Flexible **R**ectification, for scene text recognition. ASTER tackles the irregular text problem with an explicit rectification mechanism. As depicted in Figure 2, the model comprises two parts: the rectification network and the recognition network. Given an input image, the rectification network transforms the image to rectify the text in it. The transformation is parameterized Thin-Plate Spline [8] (TPS), which is a very flexible transformation that can handle a variety of text irregularities.

Fig. 1. Examples of irregular text.

During inference, the rectification network first predicts the TPS parameters from the image, then applies them to the transformation. Based on the Spatial Transformer Networks (STN) framework proposed by [28], the rectification network can be trained purely by the gradients back propagated by the recognition network, hence requiring no human annotations.

The recognition network predicts a character sequence from the rectified image in an attentional sequence-to-sequence manner. Built on the attention mechanism proposed in [3], [13], the recognition network effectively encapsulates character detection, character recognition, and language modeling into a single model, achieving accurate recognition. Furthermore, we extend the traditional unidirectional decoder into a bidirectional one. The bidirectional decoder consists of two decoders with opposite decoding directions. It merges the outputs of both decoders, leveraging the dependencies in both directions.

Through an extensive set of experiments on a number of standard datasets, we demonstrate the superior performance of ASTER on both regular and irregular text. Furthermore, when used with a text detector, ASTER shows the ability to enhance text detectors by filtering and refining their detected boxes. In particular, ASTER enables a horizontal text detector to detect oriented text. These advantages make ASTER a powerful component for end-to-end text recognition systems.

- *B. Shi, M. Yang, X. Wang, P. Lyu, and X. Bai are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, 430074, China.*
- *C. Yao is with Megvii (Face++) Inc., Beijing, 100190, China.*

Fig. 2. Overview of the proposed model. Dashed lines show the flow of gradients.

In summary, the contributions of this paper are three folded. First, we tackle the problem of irregular text recognition with an explicit rectification mechanism, which significantly improves recognition performance without extra annotations. Second, we introduce attentional sequence-to-sequence models to the problem of scene text recognition and extend it with a bidirectional decoder. Third, we propose a method for enhancing text detectors using ASTER's abilities in text rectification and recognition.

This paper surpasses its conference version [55] with three major extensions. 1) We achieve a breakthrough in the rectification performance by revising the rectification network architecture. First, images of different resolutions are used for control points prediction and for sampling, avoiding the issue of degraded resolution in the original STN framework. Second, we drop the non-linear activation in the localization network, preserving the back-propagated gradients and therefore accelerate convergence during training. Consequently, we observe significant improvements in terms of accuracy, rectified image quality, and sensitivity to initialization; 2) We extend the original recognition decoder into a bidirectional one, in order to leverage the dependencies in both directions; 3) We explore the application of ASTER in end-to-end text recognition and demonstrate its advantages. With these extensions, ASTER outperforms [55] by a large margin and shows broader applicability and advantages.

2 Related Work

2.1 Text Recognition

A rich body of literature concerning scene text recognition has been published in recent years. Comprehensive surveys can be found in [68], [72].

Early work mainly focuses on document text. Since documents usually have clean backgrounds, binarization methods [10] are often adopted for segmenting characters. However, when applied to scene text, these methods fail to handle the large variation in text appearance and the noise in natural images.

In recent decades, a prevalent approach is to localize individual characters and group them into words. In this line of works, many choose to filter a redundant set of character proposals and recognize them with a classifier. Representative proposal extraction methods include Maximally Stable Extremal Regions (MSER) [43], [46], Extremal Regions (ER) [47], and Stroke Width Transform (SWT) [14], [65], [66]. These methods exploit the characteristics of text in either texture or shape. They are effective in localizing characters in images with moderate noises.

Others take a learning-based approach and localize characters using the sliding window technique [35], [44], [45], [60], [61]. In these methods, local regions are densely classified by a binary or multi-category (each category is a letter) classifier to get the locations and scores of characters. Then, a graph-based inference is performed to find words from the detected characters.

With the success of deep neural networks in various computer vision tasks [33], [50], many started to adopt deep neural networks for text recognition. Some methods use convolutional neural networks [33], [34] (CNNs) to localize and recognize characters [29], [62], [70]. [62] created two CNNs to localize and recognize characters respectively. [29] follows a similar approach but used only one CNN to perform both tasks simultaneously. Other methods recognize text holistically. For example, Jaderberg *et al.* [24], [29] perform text recognition with a 90k-class CNN, where each class corresponds to an English word.

Text recognition can be also modeled into a structured learning problem. [25] proposed to construct a structured-output CNN for unconstrained text recognition. Su and Lu [56], [57] models text recognition as a sequence recognition problem, and addressed it with recurrent neural networks (RNNs). Their effort was followed by [21], [54], where CNN and RNN are integrated for text recognition. Wang *et al.* [59] extends this line of work by exploiting gated recurrent CNN.

Our method falls into the category of structured learning. It is based on the attentional sequence-to-sequence learning models [3], [13]. Such a model learns to predict an output sequence from an input sequence and was originally proposed for machine translation and speech recognition tasks. The conference version of this paper [55] was the first, in parallel with [36], to explore such models in the task of scene text recognition. This work further extends [55] with the bidirectional decoder.

2.2 Text Rectification

The rectification of irregular text has been previously studied for document text images [30], [40], [41]. These works rectify (or *flatten*) document images based on morphological analysis or registration techniques, which are applied to images containing multiple lines of text. Scene text, on the other hand, is usually recognized in the form of individual words, to which these methods do not generalize well.

On scene text, the irregular text problem was studied in [49], where Phan *et al.* presented a recognition method robust to perspective text distortions. Recently, Yang *et al.* [64] addressed the irregular text problem with an auxiliary character detection model and an alignment loss, which help precise character localization. Bartz *et al.* [4], [5] addressed this issue with an integrated rectification-recognition network. Compared to [64], our method is conceptually simpler, as it does not need explicit character detection. Moreover, our method requires no extra character-level annotations, in contrast to [64] and [11].

2.3 Text Detection and End-to-End Recognition

The problems of text detection and end-to-end recognition are both under very active research. Many recent text detectors are based on modern object detection or segmentation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2848939, IEEE Transactions on Pattern Analysis and Machine Intelligence

methods. For example, TextBoxes [38] adapts the SSD detector [39] to text detection. EAST [71] uses FCN [53] for text segmentation. Both methods have achieved fast and accurate text detection.

A common end-to-end recognition system comprises a detector and a recognizer in a sequential manner. Previous methods such as Weinman *et al.* [63] and Jaderberg *et al.* [27] first generate text proposals, then recognize them with an independent word recognition model. Recently, some methods try to integrate detection and recognition within an end-to-end neural network. Deep TextSpotter [9] combines an FCN-based detector and a CTC-based recognizer into an end-to-end trainable framework. Similarly, Li *et al.* [37] integrated a text proposal network and an attentional sequence recognition network.

Although this paper's focus is text recognition, we show that ASTER helps achieve state-of-the-art end-to-end performance, even without the strongest detector. Also, we demonstrate that ASTER can strengthen a detector via its rectification mechanism. These properties make ASTER an appealing recognizer in an end-to-end recognition system.

3 Model

The proposed model comprises two parts, the text rectification network and the text recognition network. In the following sections, we first describe the two parts in Section 3.1 and 3.2 respectively. Then, we describe the training strategy in Section 3.3.

3.1 Rectification Network

The rectification network rectifies an input image with a predicted 2D transformation. We adopt Thin-Plate-Spline [8] (TPS) as the transformation. TPS has found broad applications in image transformation and matching, *e.g.* [6]. It is more flexible compared to other simpler 2D transformations, *e.g.* affine and projective. TPS performs non-rigid deformation on images, handling a variety of distortions. Figure 3 exemplifies some typical rectifications. In particular, TPS can rectify both perspective and curved text, two typical types of irregular text.

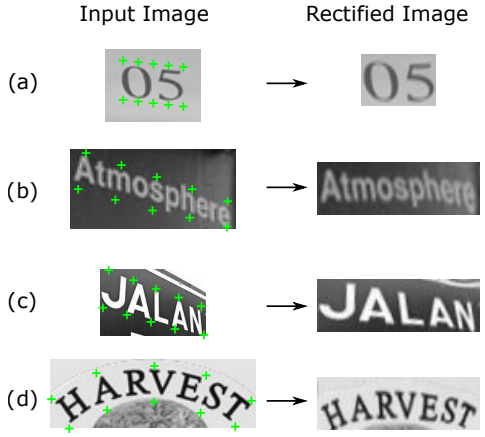


Fig. 3. The TPS transformation can rectify various types of irregular text, including but not limited to, loosely bounded a), oriented or perspectively distorted (b)(c), and curved text (d).

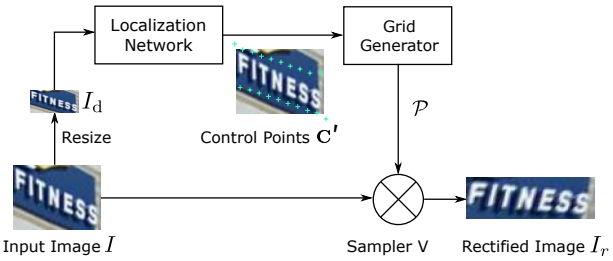


Fig. 4. Structure of the rectification network.

The rectification network is based on the Spatial Transformer Network [28] (STN). The central idea of STN is to model spatial transformation as a learnable network layer. Figure 4 depicts the structure of the rectification network. The network first predicts a set of *control points* via its localization network. Then, a TPS transformation is calculated from the control points and passed to the grid generator and the sampler to generate the rectified image I_r. Since the control points are predicted from I, the rectification network takes no extra inputs other than the input image.

3.1.1 Localization Network

We start by illustrating how TPS rectifies text in Figure 5. A TPS transformation is determined by two sets of control points of equal size, denoted by K. The control points on the output image are placed at fixed locations along the top and bottom image borders with equal spacings. As a result, when the control points on the input image are predicted along the upper and lower text edges, the resulting TPS transformation outputs a rectified image with regular text.

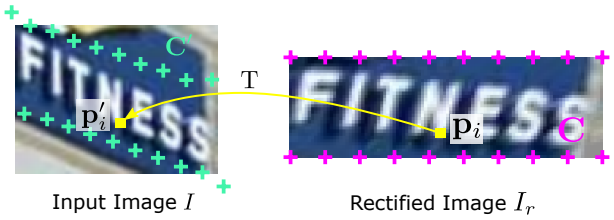


Fig. 5. Text rectification with TPS transformation. Crosses are control points. The yellow arrow represents the transformation T, connecting a point $\mathbf{p}_i$ and its corresponding point $\mathbf{p}'_i$.

Therefore, the problem of text rectification boils down to predicting the control points on the input image. The prediction is performed by a convolutional neural network. Assuming K control points on both I and I_r, their coordinates denoted by $\mathbf{C}'$ and $\mathbf{C}$, respectively. Here, $\mathbf{C} = [\mathbf{c}_1, \dots, \mathbf{c}_K] \in \Re^{2\times K}$ is the concatenation of K control points, where $\mathbf{c}_k = [x_k, y_k]^\intercal$ is the x, y coordinates of the k-th point. Likewise, $\mathbf{C}' = [\mathbf{c}'_1, \dots, \mathbf{c}'_K]$.

The localization network regresses $\mathbf{C}'$ directly from I_d, which is downsampled from I. The network consists of a few convolutional layers, with max-pooling layers inserted between them. The output layer is a fully-connected layer whose output size is $2K$. Its output vector is reshaped to $\mathbf{C} \in \Re^{2\times K}$. Values of $\mathbf{C}'$ and $\mathbf{C}$ are normalized image coordinates, where $(0, 0)$ is top left corner and $(1, 1)$ the bottom right.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2848939, IEEE Transactions on Pattern Analysis and Machine Intelligence

As we will further specify later, all modules in the rectification network are differentiable. As a result, during training, the localization network is trained purely by the back-propagated gradients, requiring no manual annotations on the control points.

3.1.2 Grid Generator

The grid generator computes a transformation and applies it to every pixel locations in I_r, generating a sampling grid $\mathcal{P} = \{\mathbf{p}_i\}$ on I. A 2D TPS transformation is parameterized by a $2 \times (K+3)$ matrix:

$$\mathbf{T} = \begin{bmatrix} a_0 & a_1 & a_2 & \mathbf{u} \\ b_0 & b_1 & b_2 & \mathbf{v} \end{bmatrix}$$

where $\mathbf{u}, \mathbf{v} \in \Re^{1\times K}$. Given a 2D point $\mathbf{p} = [x_p, y_p]^\intercal$, TPS finds its corresponding point $\mathbf{p}'$ by linearly projecting the lifted vector of $\mathbf{p}$:

$$\mathbf{p}' = \mathbf{T} \begin{bmatrix} 1 \\ \mathbf{p} \\ \phi(\|\mathbf{p} - \mathbf{c}_1\|) \\ \vdots \\ \phi(\|\mathbf{p} - \mathbf{c}_K\|) \end{bmatrix} \tag{1}$$

where $\phi(r) = r^2 \log(r)$ is the radial basis kernel applied to the Euclidean distance between $\mathbf{p}$ and the control point $\mathbf{c}_k$.

The coefficients of TPS are found by solving a linear system involving K correspondences between $\mathbf{C}$ and $\mathbf{C}'$:

$$\mathbf{c}'_i = \mathbf{T} \begin{bmatrix} 1 \\ \mathbf{c}_i \\ \phi(\|\mathbf{c}_i - \mathbf{c}_1\|) \\ \vdots \\ \phi(\|\mathbf{c}_i - \mathbf{c}_K\|) \end{bmatrix}, \; i = 1, \ldots, K,$$

subject to the following boundary conditions ($\mathbf{C}_x$ and $\mathbf{C}_y$ are respectively the x and y coordinates of $\mathbf{C}$):

$$\begin{aligned} 0 &= \mathbf{u}\mathbf{1} \\ 0 &= \mathbf{v}\mathbf{1} \\ 0 &= \mathbf{u}\mathbf{C}_x^\intercal \\ 0 &= \mathbf{v}\mathbf{C}_y^\intercal \end{aligned}$$

Combined in matrix form, the linear system is represented by

$$\begin{aligned} \mathbf{T}\boldsymbol{\Delta}_\mathbf{C} &= \begin{bmatrix} \mathbf{C}' & \mathbf{0}^{2\times 3} \end{bmatrix}, \\ \boldsymbol{\Delta}_\mathbf{C} &= \begin{bmatrix} \mathbf{1}^{1\times K} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{0} & \mathbf{0} \\ \hat{\mathbf{C}} & \mathbf{1}^{K\times 1} & \mathbf{C}^\intercal \end{bmatrix} \end{aligned} \tag{2}$$

where $\hat{\mathbf{C}} \in \Re^{K\times K}$ is a square matrix comprising $\hat{c}_{i,j} = \phi(\|\mathbf{c}_i - \mathbf{c}_j\|)$.

From Equation 2, $\mathbf{T}$ has a closed-form solution:

$$\mathbf{T} = \begin{bmatrix} \mathbf{C}' & \mathbf{0}^{2\times 3} \end{bmatrix} \boldsymbol{\Delta}_\mathbf{C}^{-1} \tag{3}$$

Note that, $\mathbf{C}$ is the predefined control points on the output image and therefore a constant. According to Eq. 2, both $\hat{\mathbf{C}}$ and $\boldsymbol{\Delta}_\mathbf{C}$ are derived purely from $\mathbf{C}$. Therefore, they are also constants and only need to be computed once.

The TPS solving and transformation process can be readily modeled into a neural network module. Figure 6 shows the construction of this module. It takes the predicted control points $\mathbf{C}'$ and every pixel location $\mathbf{p}$ on the rectified image as inputs and outputs $\mathbf{p}'$. All the operators used in the modules are differentiable and can be found in most mainstream deep learning libraries. Besides, given an image of certain resolution, the pixel locations are fixed. As a result, the computation from $\mathbf{p}$ to $\hat{\mathbf{p}}$ in Figure 6 can be cached and reused for images of the same resolution.

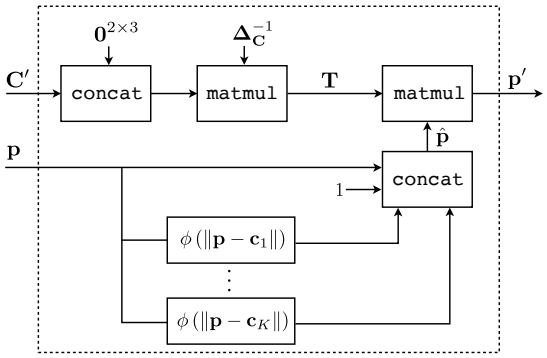


Fig. 6. Construction of the grid generator. **concat** and **matmul** are respectively matrix concatenation and multiplication operators.

3.1.3 Sampler

At the output of the rectification network, the sampler generates the rectified image:

$$I_r = V(\mathcal{P}, I)$$

The sampler computes the value of $\mathbf{p}$ by interpolating the neighbor pixels of $\mathbf{p}'$. Since $\mathbf{p}'$ may fall outside the image, a value clipping is performed before the sampling to restrict the sampling points inside the image border.

The sampler is made differentiable, *i.e.* it can back propagate the gradients on I_r to $\mathcal{P}$. This is achieved by the differentiable image sampling method. We refer the readers to [28] for more details.

3.1.4 Comparison with STN [28] and RARE [55]

Compared to our previous work [55] and the original STN paper [28], this paper introduces two improvements.

Unlike STN [28], we use images of *different* sizes for the localization network and for the sampler. The localization network operates on the smaller image, I_d, which is a downsampled version of I, in order to reduce the number of parameters needed for the prediction. Meanwhile, the sampler operates on the original image (or similarly, the original image resized to a high resolution). Since the rectification network often crops its input images, sampling on a high-resolution image avoids degraded output resolution, therefore preserving the image quality of I_r. In addition, this paper elaborates the use of TPS in STN.

Different than [55], we do not restrict the values of $\mathbf{C}'$ by using a $\tanh$ activation function in the last fully-connected layer. Dropping the non-linear activation function may lead to faster convergence during training, as gradients are well preserved during backpropagation. Without $\tanh$, the control points may fall outside image borders, so value clipping is performed in the sampler to ensure valid sampling. We empirically find that this trick significantly improves the performance and the stability to weight initialization.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2848939, IEEE Transactions on Pattern Analysis and Machine Intelligence

3.2 Recognition Network

The text recognition network predicts a character sequence directly from the rectified image. The network is made to be end-to-end trainable. It is trained with only images and their groundtruth text annotations.

Recent work [54] has demonstrated the effectiveness of modeling text recognition as a sequence recognition problem. At the core of that model is the Connectionist Temporal Classification (CTC) method [17]. CTC provides a differentiable loss function that is insensitive to horizontal character placement and spacing, enabling end-to-end trainable sequence recognition. Despite its effectiveness, CTC does not have a mechanism to model the dependencies among its output characters. Therefore, [54] relies on an external language model, such as a lexicon, to incorporate language priors into its recognition.

We tackle the recognition problem using a sequence-to-sequence model extended by a bidirectional decoder. Since the output of a sequence-to-sequence model is generated by a recurrent neural network, it captures the character dependencies, thus incorporating language modeling into the recognition process. Furthermore, the bidirectional decoder captures the character dependencies in both directions, leveraging richer context and boosting performance. Figure 7 depicts the structure of the unidirectional version. Following the classical sequence-to-sequence model, our model consists of an encoder and a decoder. They are described in the following two sections.

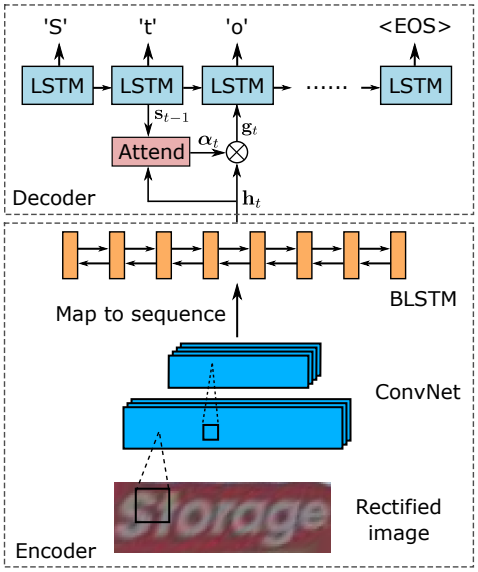


Fig. 7. Structure of the basic text recognition network.

3.2.1 Encoder: Convolutional-Recurrent Neural Network

Rich and discriminative features are of key importance to a recognition model. Ideally, the characters of a word are arranged in a line and are therefore well represented by a feature sequence that describes local image regions arranged from left to right (or likewise, right to left).

As shown in Figure 7, the encoder first extracts a feature map from the input image with a stack of convolutional layers. The convolutional layers ("ConvNet") are designed such that the feature map has the height of 1. Next, the feature map is converted into a feature sequence by being split along its row axis. The shape of the feature map is $h_{\text{conv}} \times w_{\text{conv}} \times d_{\text{conv}}$, respectively its height, width and depth. After the splitting, the feature map is converted into a sequence of w_{conv} vectors, each having $h_{\text{conv}} d_{\text{conv}}$ dimensions.

The ConvNet extracts strong image features, particularly when it consists of many convolutional layers with residual connections [20]. But still, such features are constrained by their *receptive fields*, *i.e.* the image region they capture. To enlarge the feature context, we employ a multi-layer Bidirectional LSTM (BLSTM) network [18] over the feature sequence. The BLSTM network analyzes the feature sequence bidirectionally, capturing long-range dependencies in both directions. It outputs a new feature sequence of the same length, denoted by $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_n]$, where $n = w_{\text{conv}}$.

3.2.2 Decoder: Attentional Sequence-to-Sequence Model

The sequence-to-sequence model translates the feature sequence into a character sequence. It is able to input and output sequences of arbitrary lengths. Such a model is appealing due to its simplicity and powerfulness in sequence modeling and its ability to capture output dependencies.

Sequence-to-sequence models come in various forms, such as [3], [15], [58]. We build our decoder based on the attentional sequence-to-sequence model [3], [13], as it has access to the encoder output at every decoding step and has an intuitive and explainable behavior that allows easier debugging and analysis.

An attentional sequence-to-sequence model is a unidirectional recurrent network. It works iteratively for T steps, producing a symbol sequence of length T, denoted by $(y_1, \ldots, y_T)$.

At step t, the decoder predicts either a character or an end-of-sequence symbol (EOS), based on the encoder output $\mathbf{H}$, the internal state $\mathbf{s}_{t-1}$, and the symbol y_{t-1} predicted in the last step. In this step, the decoder starts by computing a vector of attentional weights, $\boldsymbol{\alpha}_t \in \Re^n$, through its attention mechanism:

$$e_{t,i} = \mathbf{w}^\intercal \tanh\left(\mathbf{W}\mathbf{s}_{t-1} + \mathbf{V}\mathbf{h}_i + b\right) \tag{4}$$
$$\alpha_{t,i} = \exp\left(e_{t,i}\right) / \sum_{i'=1}^{n} \exp(e_{t,i'})$$

where $\mathbf{w}, \mathbf{W}, \mathbf{V}$ are trainable weights.

The attentional weights effectively indicate the importance of every item of the encoder output. Taking the weights as the coefficients, the decoder linearly combines the columns of $\mathbf{H}$ into a vector, which is called a *glimpse*:

$$\mathbf{g}_t = \sum_{i=1}^{n} \alpha_{t,i} \mathbf{h}_i \tag{5}$$

As the name suggests, a glimpse describes a small part of the whole context encoded in $\mathbf{H}$. It is taken as an input to the recurrent cell of the decoder, which produces an output vector and a new state vector:

$$(\mathbf{x}_t, \mathbf{s}_t) = \text{rnn}\left(\mathbf{s}_{t-1}, (\mathbf{g}_t, f(y_{t-1}))\right)$$

where $(\mathbf{g}_t, f(y_{t-1}))$ is the concatenation of $\mathbf{g}_t$ and the one-hot embedding of y_{t-1}. rnn represents the step function of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2848939, IEEE Transactions on Pattern Analysis and Machine Intelligence

any recurrent unit (*e.g.* LSTM [22], GRU [12]), whose output and new state are denoted by $\mathbf{x}_t$ and $\mathbf{s}_t$, respectively. At last, $\mathbf{x}_t$ is taken for predicting the current-step symbol:

$$p(y_t) = \mathrm{softmax}\left(\mathbf{W}_o \mathbf{x}_t + b_o\right) \tag{6}$$

$$y_t \sim p(y_t) \tag{7}$$

Since y_{t-1} is incorporated in the computation, the decoder learns to capture the dependencies between its output characters. This acts as an implicit language model, which assists the recognition with the language priors it learns.

During inference, we can either adopt a greedy decoding scheme by taking the symbol with the highest $\mathrm{softmax}$ score or adopt a beam search approach by maintaining the k candidates with the top accumulative scores at every step. In practice, we use a beam search with $k = 5$. The beam search yields a slight but consistent improvement in accuracy compared with the greedy decoding.

3.2.3 Bidirectional Decoder

Although a sequence-to-sequence decoder captures output dependencies, it captures in only one direction and misses the other. For example, a decoder that recognizes text in the left-to-right order may have difficulty deciding the first letter between upper-case '`I`' and lower-case '`l`' in certain fonts, as they are hard to distinguish visually and the decoder has no memory of past decoded letters. Such words may be easier to recognize by a decoder that works in the right-to-left order, as the rest of the letters hint the first one based on the language prior.

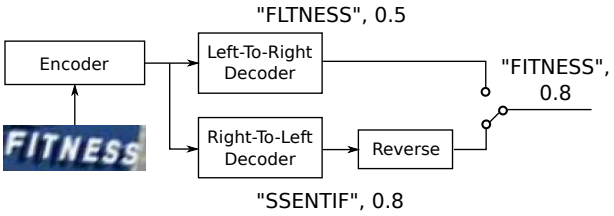


Fig. 8. Bidirectional decoder. "0.5" and "0.8" are recognition scores.

As this example suggests, decoders that work in opposite directions are potentially complementary. To leverage the dependencies in both directions, we propose a bidirectional decoder, which consists of two decoders with opposite directions. As illustrated in Figure 8, one decoder is trained to predict the characters from left to right and the other right to left. Two recognition results are produced after running both decoders. To merge the results, we simply pick the one with the highest recognition score, which is the sum of log-softmax scores of all predicted symbols.

3.3 Training

The model is trained end to end under a multi-task setting, whose objective is

$$L = -\frac{1}{2}\sum_{t=1}^{T}\left(\log p_{\mathrm{ltr}}(y_t|I) + \log p_{\mathrm{rtl}}(y_t|I)\right) \tag{8}$$

where $y_1, \ldots, y_t, \ldots, y_T$ is the groundtruth text represented by a character sequence. The objective is the average of the losses on the left-to-right decoder and the right-to-left decoder, whose predicted distributions are denoted by p_{ltr} and p_{rtl}, respectively.

The weights of all layers in our model are initialized randomly, except for the localization network. Since the TPS transformation is computed from the control points predicted by the localization network, a randomly initialized localization network results in randomly placed control points, which will distort I_r and causes instability during the training process. To address this issue, we initialize the last fully-connected layer (fc2) so that I_r is not distorted at the beginning of the training. Concretely, we set the weights of fc2 to zeros and its biases to values such that $\mathbf{C}'$ is identical to $\mathbf{C}$. We find that this initialization scheme yields a more stable training process.

4 Experiments

We conduct extensive experiments to verify the effectiveness of each part of our model and compare its performance with other state-of-the-art methods. In this section, we begin by specifying the experimental settings in Section 4.1. Then, we conduct a few ablation studies in Section 4.2 and Section 4.3, each targeted at a model part to demonstrate its effectiveness and analyze its behavior. Finally, in Section 4.4 we evaluate ASTER on public datasets and compare it with other state-of-the-art methods.

4.1 Experimental Settings

4.1.1 Datasets

The proposed model is trained on two synthetic datasets without finetuning on other datasets. The model is tested on 5 standard datasets to evaluate its general recognition performance. Besides, we test the model on 2 special datasets of irregular text to demonstrate its rectification ability. Following the standard, we evaluate recognition performance using the case-insensitive word accuracy.

Synth90k is the synthetic text dataset proposed in [24]. The dataset contains 9 million images generated from a set of 90k common English words. Words are rendered onto natural images with random transformations and effects. Every image in Synth90k is annotated with a groundtruth word. All of the images in this dataset are taken for training.

SynthText is the synthetic text dataset proposed in [19]. The generation process is similar to that of [24]. But unlike [24], SynthText is targeted for text detection. Therefore, words are rendered onto full images. We crop the words using the groundtruth word bounding boxes.

IIIT5k-Words (IIIT5k) [44] contains 3000 test images collected from the web. Each image is associated with a short, 50-word lexicon and a long, 1000-word lexicon. A lexicon consists of the groundtruth word and other random words.

Street View Text (SVT) [60] is collected from the Google Street View. The test set contains 647 images of cropped words. Many images in SVT are severely corrupted by noise, blur, and low resolution. Each image is associated with a 50-word lexicon.

ICDAR 2003 (IC03) [42] contains 860 images of cropped word after filtering. Following [60], we discard words that contain non-alphanumeric characters or have less than three characters. Each image has a 50-word lexicon defined in [60].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2848939, IEEE Transactions on Pattern Analysis and Machine Intelligence

ICDAR 2013 (IC13) [32] inherits most images from IC03 and extends it with new images. The dataset is filtered by removing words that contain non-alphanumeric characters. The dataset contains 1015 images. No lexicon is provided.

ICDAR 2015 Incidental Text (IC15) is the Challenge 4 of the ICDAR 2015 Robust Reading Competition [31]. This challenge features incidental text images, which are taken by a pair of Google Glasses without careful positioning and focusing. Consequently, the dataset contains a lot of irregular text. Testing images are obtained by cropping the words using the groundtruth word bounding boxes.

SVT-Perspective (SVTP) is proposed in [49] for evaluating the performance of recognizing perspective text. Images in SVTP are picked from the side-view images in Google Street View. Many of them are heavily distorted by the non-frontal view angle. The dataset consists of 639 cropped images for testing, each with a 50-word lexicon inherited from the SVT dataset.

CUTE80 (CUTE) is proposed in [51]. The dataset focuses on curved text. It contains 80 high-resolution images taken in natural scenes. CUTE80 is originally proposed for detection tasks. We crop the annotated words and get a test set of 288 images. No lexicon is provided.

4.1.2 Text Rectification Network

Images are resized to 64×256 before entering the rectification network. We use a large input size in order to reserve high resolution before the rectification sampling. The sampler outputs images of size 32×100, which is also the input size of the recognition network.

The localization network works on the input image downsampled to 32×64. It consists of 6 convolutional layers with kernel size 3×3. Each of the first 5 layers is followed by a 2×2 max-pooling layer. The number of the output filters are respectively 32, 64, 128, 256, 256 and 256. The convolutional layers are followed by two fully-connected layers. Their number of output units are respectively 512 and $2K$, where K is the number of control points. We set K to 20 throughout the experiments. Other values of K lead to similar results.

4.1.3 Text Recognition Network

The configurations of the recognition network are listed in Table 1. We use a 45-layer residual network [20] as the convolutional feature extractor. Each residual unit comprises a 1×1 convolution followed by 3×3 convolution, as recent work [23] suggests the efficiency of this scheme. Feature maps are downsampled by 2×2-stride convolutions in the first two residual blocks. The stride is changed to 2×1 in the fourth and fifth residual blocks. The 2×1 down-sampling stride reserves more resolution along the horizontal axis, in order to distinguish neighbor characters.

Following the residual network are two layers of Bidirectional LSTM (BiLSTM). Each layer consists of a pair of LSTMs with 256 hidden units. The outputs of the LSTMs are concatenated and linearly projected to 256 dimensions before entering the next layer.

The decoders are attentional LSTMs. The number of attention units and hidden units are both 256. The decoder recognizes 94 character classes, including digits, upper-case and lower-case letters, and 32 ASCII punctuation marks.

TABLE 1
Text recognition network configurations. Each block is a residual network block. 's' stands for stride of the first convolutional layer in a block. '*' means dynamic output length. "Out Size" is feature map size for convolutional layers (height×width) and sequence length for recurrent layers. "Att. LSTM" stands for attentional LSTM decoder. Two decoders are instantiated and work in parallel.

	Layers	Out Size	Configurations
Encoder	Block 0	32×100	3×3 conv, s 1×1
	Block 1	16×50	$\begin{bmatrix} 1 \times 1 \text{ conv, } 32 \\ 3 \times 3 \text{ conv, } 32 \end{bmatrix} \times 3$, s 2×2
	Block 2	8×25	$\begin{bmatrix} 1 \times 1 \text{ conv, } 64 \\ 3 \times 3 \text{ conv, } 64 \end{bmatrix} \times 4$, s 2×2
	Block 3	4×25	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ 3 \times 3 \text{ conv, } 128 \end{bmatrix} \times 6$, s 2×1
	Block 4	2×25	$\begin{bmatrix} 1 \times 1 \text{ conv, } 256 \\ 3 \times 3 \text{ conv, } 256 \end{bmatrix} \times 6$, s 2×1
	Block 5	1×25	$\begin{bmatrix} 1 \times 1 \text{ conv, } 512 \\ 3 \times 3 \text{ conv, } 512 \end{bmatrix} \times 3$, s 2×1
	BiLSTM 1	25	256 hidden units
	BiLSTM 2	25	256 hidden units
Decoder	Att. LSTM	*	256 attention units 256 hidden units
	Att. LSTM	*	256 attention units 256 hidden units

When the evaluation protocol is case insensitive and disregards punctuation, we normalize decoder outputs to lower cases and remove all predicted punctuation marks.

4.1.4 Optimization

The model is trained from scratch. We adopt ADADELTA [69] as the optimizer. A simpler SGD optimizer with momentum will also train the model successfully, but with slightly lower accuracy. The model is trained by batches of 64 examples for a million iterations. Every batch is constructed by 32 examples drawn from Synth90k and the other 32 from SynthText. Models trained in this manner significantly outperform models (*e.g.* [54], [55]) trained solely on Synth90k. The learning rate is set to 1.0 initially and decayed to 0.1 and 0.01 at step 0.6M and 0.8M respectively. Although learning rate in ADADELTA is adaptive, we find the classic learning rate schedule beneficial to the performance.

4.1.5 Implementation

We implement the proposed model using TensorFlow [1]. The model is trained on a single NVIDIA TITAN Xp graphics card with 12GB memory. The training speed is about 6.5 iterations/s, taking less than 2 days to reach convergence. The inference speed is 20ms per image when the test batch size is 1. This speed could be boosted with greater batch size.

4.2 Experiments on Text Rectification

We study two aspects of the text rectification network. First, we study the effect of text rectification in both quantitative

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2848939, IEEE Transactions on Pattern Analysis and Machine Intelligence

and qualitative terms. Second, for the localization network, we study its sensitivity to weight initialization.

4.2.1 Effect of Rectification

To analyze the effect of rectification, we study two variants of the proposed model. The first variant only consists of the recognition network, performing no rectification. To avoid the effect of other model parts, we also change the bidirectional decoder to a single-directional one. The second variant is the first one plus the rectification network. Both variants are trained from scratch, using the same training settings described in Section 4.1. Their performance is evaluated on six of the test datasets, namely IIIT5k, SVT, IC03, IC13, SVTP, and CUTE.

TABLE 2
Recognition accuracies with and without rectification.

Variants	IIIT5k	SVT	IC03	IC13	SVTP	CUTE
Without Rect.	91.93	88.76	93.49	89.75	74.11	73.26
With Rect.	**92.67**	**91.16**	**93.72**	**90.74**	**78.76**	**76.39**

Table 2 lists the results of the two variants. As can be seen, the model with rectification outperforms the one without on all datasets, particularly on SVTP (+4.7%) and CUTE (+3.1%). Since these two datasets both consist of irregular text, the rectification shows a significant effect. Furthermore, we construct a series of datasets with ascending level of irregularities. This is achieved by mixing SVTP+CUTE and IIIT5k (all examples considered regular) with different ratios. All datasets have 933 examples. Fig. 9 plots the rectification improvement (accuracy difference between **With Rect.** and **Without Rect.**) versus portion of irregular examples. As can be seen, the rectification improvement monotonically increases with the level of irregularity, showing the rectification effects on irregular text.

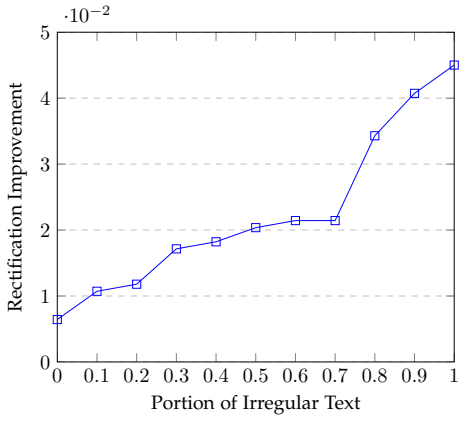
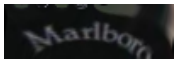
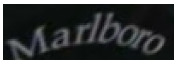


Fig. 9. Rectification Improvement versus Portion of Irregular Text

For a qualitative comparison, Table 4 visualizes the rectification results on some examples from CUTE80 and SVT-Perspective. Even without a direct supervision on control point locations, the rectification network learns to place the control points near the upper and lower edges of text. The points are aligned on a smooth curve with even spacing, causing little distortions or artifacts in the rectified images.

TABLE 3
Rectified images and recognition results by [55] and by ASTER. Recognition errors are marked by red characters.

By [55]	By ASTER	By [55] By ASTER
		window wyndham
		optimute optimum
		coera cobra
		warehouse warehouse
		tallobs tailors
		maribon marlboro
		scotting colls

Seen from Table 4, the rectification network works effectively on the perspective text. Even on images with heavy perspective distortions (*e.g.* "starbucks" and "storage"), the rectification network rectifies the words to regular ones, alleviating recognition difficulty considerably. The rectification network also rectifies curved text. For example, on the words "ronaldo", "optimum", and "grove", control points are predicted in arc shapes, and the text is rectified to have regular shapes. Although the rectified images still have distortions, the recognition network is able to correctly recognize them.

Another interesting phenomenon is that the rectification tends to skew text. This phenomenon is observed in many examples, such as the "academy", "entrance", "museum", and "storage" in Table 4. We conjecture the reason to be that the skew simplifies learning, as it causes adjacent characters to have overlapping spacings, therefore introducing dependencies along the vertical image axis.

Finally, we observe that some control points are placed outside their image borders, for example, those in "city" and "lights". Unlike [55], we do not force control points to be within image borders, as doing so may disturb the sampling grid and causes image distortions. Seen from Table 3, although ASTER's rectification fails occasionally, it generally produces images with much less distortion than those of [55]. This shows that the simple modifications result in significant improvement in image quality.

4.2.2 Sensitivity to Weight Initialization

Proper weight initialization is necessary for training the rectification network smoothly. As mentioned in Section 3.3, we initialize the last fully-connected layer with zero weights and biases of certain values. We name this initialization scheme **identity**.

To demonstrate the effect of weight initialization, Figure 10 compares **identity** to another initialization scheme called **random**, where all model layers are randomly initialized. As can be seen, **identity** leads to a much faster

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2848939, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 4
Selected results on SVT-Perspective and CUTE80. For every two rows, the first row contains the input images (top), the predicted control points (visualized as green crosses), and the rectified images (bottom). The second row contains the recognition results.

convergence and a more stable training process. Meanwhile, a randomly initialized model can still be trained successfully with more training iterations, and it reaches very close accuracies as **identity** in the end. We observe that a randomly initialized model produces garbage rectified images at first, but returns to normal after a few thousands of training iterations.

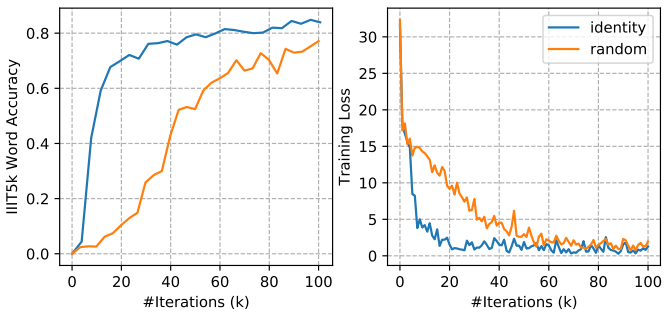


Fig. 10. Word accuracies (left) and training losses (right) with different model initialization schemes.

In [55], some carefully designed weight initialization schemes are necessary to train the model successfully. And the training completely fails with random initialization. In contrast, the model in this paper is much less sensitive to weight initialization. It can be trained successfully even with random initialization.

4.3 Experiments on Text Recognition

In this section, we study several key aspects of the text recognition network, including its attention mechanism, the bidirectional decoder, and the recognition performance versus word length.

4.3.1 Analysis on Attention

The attention mechanism plays a central role in the recognition network. According to Equations 5, local features are weighted combined to recognize a character. This suggests that the attention mechanism performs implicit character detection.

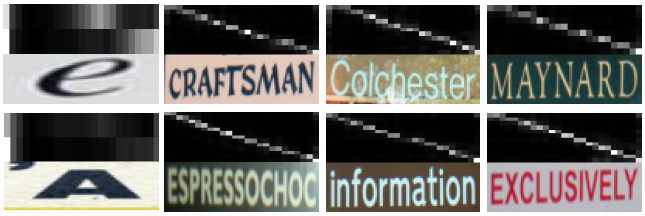


Fig. 11. Attentional weights visualizations.

To understand the behavior of the decoder, we extract the attentional weights, *i.e.* $\alpha_{t,i}$ in Equation 5, and visualize them on several examples shown in Figure 11. Above every image, a matrix of attentional weights is visualized in a 2D map. The t-th row of the map corresponds the attentional weights at t-th decoding step. Except for very short words, we can observe clear alignments between the attentional weights and the characters. This demonstrates the implicit character detection performed by the recognition network.

4.3.2 Bidirectional Decoder

To evaluate the effectiveness of the bidirectional decoder, we create three model variants, namely **L2R**, which recognizes text in the left-to-right order; **R2L**, which recognizes text in the right-to-left order; **Bidirectional**, the bidirectional decoder. These variants are also trained from scratch using the same training settings described in Section 4.1. Table 5 compares their recognition accuracies.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2848939, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 5
Recognition accuracies with different decoders. **L2R** and **R2L** stand for left-to-right and right-to-left, respectively.

Variants	IIIT5k	SVT	IC03	IC13	SVTP	CUTE
L2R	91.93	88.76	93.49	89.75	74.11	73.26
R2L	91.43	89.96	92.79	89.95	73.95	**74.31**
Bidirectional	**92.27**	**91.57**	**93.60**	**90.54**	**74.26**	**74.31**

Overall, **L2R** and **R2L** have similar accuracies. **L2R** outperforms on IIIT5k, IC03, and SVTP, while **R2L** outperforms on the rest. This indicates that the two variants may favor different data distributions. Meanwhile, **Bidirectional** outperforms both variants on all the datasets, with only one exception where **Bidirectional** equals the better variant. In particular, on SVT, **Bidirectional** outperforms the other two variants by 2.8% and 1.6% respectively, verifying the effectiveness of the bidirectional decoder.

4.3.3 Accuracy Versus Word Length

The recognition network takes as input fixed-size images. Although resizing images to a fixed size inevitably causes image distortion, we argue that its effect on performance is minor. The main reason is that the distortion equally affects training and testing data. As such, our model is trained to handle elongated and condensed examples.

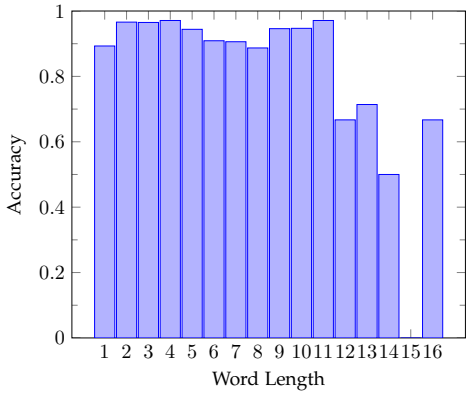


Fig. 12. Recognition accuracies versus word length (tested on IIIT5k)

Figure 12 shows the relationship between recognition accuracy and word length. As can be seen, the recognition accuracy is fairly even on words whose lengths are equal to or less than 11. Beyond this length, a drop in accuracy is observed. However, this is in part due to that long words are inherently harder to predict correctly under the measure of whole-word accuracy.

We have also tried replacing the fixed-size resizing by proportional resizing plus padding, but it results in worse performance in most cases.

4.4 Comparison to State of the Art

Finally, we compare the performance of our model with other state-of-the-art models. Some datasets provide lexicons for constraining recognition outputs. When a lexicon is given, we simply replace the predicted word with the nearest lexicon word under the metric of edit distance.

Table 6 compares the recognition accuracies across a number of methods. Our method achieves 9 best results out of 12. Particularly, on IIIT5k and SVT, our method nearly halve the recognition errors comparing to the previous bests. Our model only falls short on a few results compared to [11] and [26]. However, it is important to notice that 1) [11] uses extra character-level annotations, while ASTER does not; 2) [26] is a constrained-output recognizer. It only recognizes within its 90k dictionary, while ASTER is unconstrained. Considering that the tested datasets cover a wide range of real-world scenarios and that all the results are produced by one model, it is clear that our method works strongly in general.

Table 6 also lists two variants of ASTER, namely ASTER-A and ASTER-B. They differ ASTER only in the ConvNet architecture and the training data. ASTER-A outperforms [55] under all datasets and metrics, with only one exception on IIIT5k, which consists of regular text only. This further verifies the effectiveness of the extensions introduced in this paper. [64], also using VGG, outperforms ASTER-A on some datasets. However, [64] uses a private training dataset with character-level annotations.

We have demonstrated the effectiveness of the rectification and the bidirectional alone. With both the rectification and the bidirectional decoder, ASTER outperforms the two variants, the **With Rect.** in Table 2 and the **Bidirectional** in Table 5, across all the tested datasets. Therefore, we can see that the performance gains brought by the rectification and the bidirectional decoder are additive.

5 End-to-End Recognition

A text recognizer is often used together with a text detector to construct an end-to-end recognition system. Usually, the detector first localizes word bounding boxes. Then, the recognizer recognizes the images cropped by the boxes. The recognition performance is often affected by the quality of the detected boxes since loosely bounded boxes result in imperfect crops, which are difficult to recognize.

ASTER is appealing for end-to-end recognition systems not only for its robust recognition performance but also for its ability to *enhance detection*, which comes in two folds. First, the recognition scores produced by ASTER can be used to filter the detection boxes. Second, ASTER can rectify detection boxes via its rectification network. As we have demonstrated, ASTER tends to predict control points along the upper and lower text edges. From the control points, we can estimate a new, oriented bounding box to replace the original detection box. As a result, ASTER can turn a horizontal detector into an oriented one.

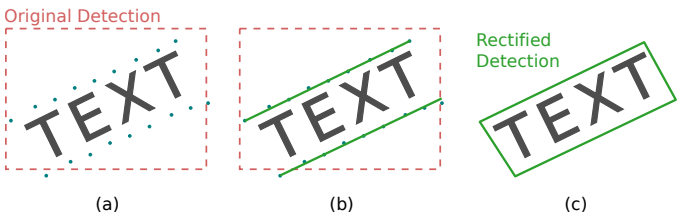


Fig. 13. Detection box rectification process.

TABLE 6
Recognition results comparison. "50", "1k", "Full" are lexicons. "0" means no lexicon. *The conference version of this paper. "90k" and "ST" are the Synth90k and the SynthText datasets, respectively. "ST$^+$" means including character-level annotations. "Private" means private training data.

Methods	ConvNet, Data	IIIT5k			SVT		IC03			IC13	IC15	SVTP	CUTE
		50	1k	0	50	0	50	Full	0	0	0	0	0
Wang *et al.* [60]	-	-	-	-	57.0	-	76.0	62.0	-	-	-	-	-
Mishra *et al.* [44]	-	64.1	57.5	-	73.2	-	81.8	67.8	-	-	-	-	-
Wang *et al.* [62]	-	-	-	-	70.0	-	90.0	84.0	-	-	-	-	-
Bissacco *et al.* [7]	-	-	-	-	-	-	90.4	78.0	-	87.6	-	-	-
Almazan *et al.* [2]	-	91.2	82.1	-	89.2	-	-	-	-	-	-	-	-
Yao *et al.* [67]	-	80.2	69.3	-	75.9	-	88.5	80.3	-	-	-	-	-
Rodríguez-Serrano *et al.* [52]	-	76.1	57.4	-	70.0	-	-	-	-	-	-	-	-
Jaderberg *et al.* [29]	-	-	-	-	86.1	-	96.2	91.5	-	-	-	-	-
Su and Lu [56]	-	-	-	-	83.0	-	92.0	82.0	-	-	-	-	-
Gordo [16]	-	93.3	86.6	-	91.8	-	-	-	-	-	-	-	-
Jaderberg *et al.* [26]	VGG, 90k	97.1	92.7	-	95.4	80.7	98.7	**98.6**	93.1	90.8	-	-	-
Jaderberg *et al.* [25]	VGG, 90k	95.5	89.6	-	93.2	71.7	97.8	97.0	89.6	81.8	-	-	-
Shi *et al.* [54]	VGG, 90k	97.8	95.0	81.2	97.5	82.7	98.7	98.0	91.9	89.6	-	-	-
*Shi *et al.* [55]	VGG, 90k	96.2	93.8	81.9	95.5	81.9	98.3	96.2	90.1	88.6	-	71.8	59.2
Lee *et al.* [36]	VGG, 90k	96.8	94.4	78.4	96.3	80.7	97.9	97.0	88.7	90.0	-	-	-
Yang *et al.* [64]	VGG, Private	97.8	96.1	-	95.2	-	97.7	-	-	-	-	75.8	69.3
Cheng *et al.* [11]	ResNet, 90k+ST$^+$	99.3	97.5	87.4	97.1	85.9	**99.2**	97.3	94.2	**93.3**	70.6	-	-
ASTER-A	VGG, 90k	98.1	95.7	81.7	97.6	85.5	98.7	97.3	92.2	88.6	67.6	73.2	63.9
ASTER-B	ResNet, 90k	98.7	96.3	83.2	**99.2**	87.6	99.1	97.6	92.4	89.7	68.9	75.4	67.4
ASTER	ResNet, 90k+ST	**99.6**	**98.8**	**93.4**	**99.2**	**93.6**	98.8	98.0	**94.5**	91.8	**76.1**	**78.5**	**79.5**

Figure 13 illustrates the detection box rectification process. Assuming the text is detected by a horizontal detector. Given the predicted control points, we estimate two straight lines from the first half and the second half of the points, respectively, using the linear least squares regression method. For each line, two endpoints are found by the minimum and maximum x coordinates of the control points projected onto the line. The four endpoints construct a quadrilateral, which is the rectified detection. Similarly, we can estimate an axis-aligned rectangle or an oriented rectangle, depending on the task.

We first evaluate the end-to-end recognition performance of ASTER. An end-to-end recognition system is constructed using TextBoxes [38] and ASTER. Although TextBoxes is not the strongest text detector at the time of writing, our emphasis is that ASTER can achieve superior end-to-end accuracies even without a strong text detector. The source code and model are obtained from the GitHub repository[1]. We finetune the original model on IC15, which is an oriented text detection dataset. Specifically, we use the SGD optimizer with a learning rate of 10^{-4} to finetune the model for 1500 steps; The batch size is set to 16; The rest of the settings follow that of [38].

Table 8 summarizes the results. Our system achieves state-of-the-art results under all metrics. Note that, compared with Deep TextSpotter [9], our system has a weaker detector, since TextBoxes is horizontal while [9] is oriented. But still, our end-to-end performance beats [9] by a considerable margin.

1. https://github.com/MhLiao/TextBoxes

Furthermore, we demonstrate how ASTER strengthens the detector. Table 7 compares the detection accuracies among the original TextBoxes [9], TextBoxes strengthened by ASTER without rectification, and TextBoxes strengthened by ASTER. Without the rectification, ASTER still strengthens TextBoxes significantly, by re-scoring detection boxes with its recognition scores. The rectification brings further improvement, which comes from both better re-scoring and the detection rectification.

Figure 14 shows the end-to-end and the detection rectification results. For every horizontal box produced by TextBoxes, ASTER generates a quadrilateral as the rectified detection result. The quadrilateral bound the text more tightly, therefore increasing detection accuracies.

TABLE 7
Detection accuracies by different systems.

Detection Methods	IC15 Detection		
	Precision	Recall	HMean
TextBoxes [38]	0.600	0.653	0.626
TextBoxes + ASTER w/o Rect.	0.797	0.621	0.698
TextBoxes + ASTER	**0.863**	**0.692**	**0.768**

6 Conclusion

This paper addresses the irregular text recognition problem with an explicit rectification mechanism based on the STN framework and the TPS transformation. The resulting text recognizer, called ASTER, shows superior performance on cropped text recognition and end-to-end recognition tasks.

TABLE 8
End-to-end results comparison. "End-to-End" and "Word-Spotting" are two different measures. "Strong", "Weak", and "Generic" denote different lexicons.

End-to-End Systems	IC15 End-to-End			IC15 Word-Spotting		
	Strong	Weak	Generic	Strong	Weak	Generic
Stradvision-2 [31]	0.437	-	-	0.459	-	-
TextSpotter [48]	0.350	0.199	0.156	0.37	0.21	0.16
Deep TextSpotter [9]	0.54	0.51	0.47	0.58	0.53	0.51
TextBoxes [38] + ASTER w/o Rect.	0.638	0.609	0.574	0.678	0.645	0.606
TextBoxes [38] + ASTER	**0.706**	**0.673**	**0.640**	**0.752**	**0.713**	**0.676**

Besides, owing to its rectification mechanism, ASTER shows extra merits in terms of strengthening text detectors, even enabling oriented text detection of detectors designed for horizontal text.

In this work, end-to-end text recognition is addressed in a two-stage manner, meaning the detection is performed in another network using separate features. As we have demonstrated, ASTER performs implicit text detection. But this detection capability is limited to a small range near the target text. Expanding this range to the full image would yield a one-stage, end-to-end recognition system, and is a direction worthy of further investigation.

ACKNOWLEDGEMENTS

This work was supported by National Key R&D Program of China No. 2018YFB1004600, NSFC 61733007, and NSFC 61573160.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016.*, pages 265–283, 2016.

[2] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(12):2552–2566, 2014.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[4] C. Bartz, H. Yang, and C. Meinel. SEE: towards semi-supervised end-to-end scene text recognition. *CoRR*, abs/1712.05404, 2017.

[5] C. Bartz, H. Yang, and C. Meinel. STN-OCR: A single neural network for text detection and text recognition. *CoRR*, abs/1707.08831, 2017.

[6] S. J. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[7] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *ICCV*, 2013.

[8] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(6):567–585, 1989.

[9] M. Busta, L. Neumann, and J. Matas. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *ICCV*, pages 2223–2231, 2017.

[10] R. G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(7):690–706, 1996.

[11] Z. Cheng, F. Bai, Y. Xu, G. Zheng, S. Pu, and S. Zhou. Focusing attention: Towards accurate text recognition in natural images. In *ICCV*, pages 5086–5094, 2017.

[12] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[13] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. *CoRR*, abs/1506.07503, 2015.

[14] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 2963–2970, 2010.

[15] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *ICML*, pages 1243–1252, 2017.

[16] A. Gordo. Supervised mid-level features for word image representation. In *CVPR*, 2015.

[17] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006.

[18] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868, 2009.

[19] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *CVPR*, pages 2315–2324, 2016.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[21] P. He, W. Huang, Y. Qiao, C. C. Loy, and X. Tang. Reading scene text in deep convolutional sequences. In *AAAI*, pages 3501–3508, 2016.

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

[23] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269, 2017.

[24] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *NIPS Deep Learning Workshop*, 2014.

[25] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Deep structured output learning for unconstrained text recognition. In *ICLR*, 2015.

[26] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *Int. J. Comput. Vision*, 2015.

[27] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20, 2016.

[28] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015.

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

[30] L. Jagannathan and C. Jawahar. Perspective correction methods for camera based document analysis. In *Proc. First Int. Workshop on Camera-based Document Analysis and Recognition*, pages 148–154, 2005.

[31] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. K. Ghosh, A. D. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. ICDAR 2015 competition on robust reading. In *ICDAR*, pages 1156–1160, 2015.

[32] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. Almazán, and L. de las Heras. ICDAR 2013 robust reading competition. In *ICDAR*, 2013.

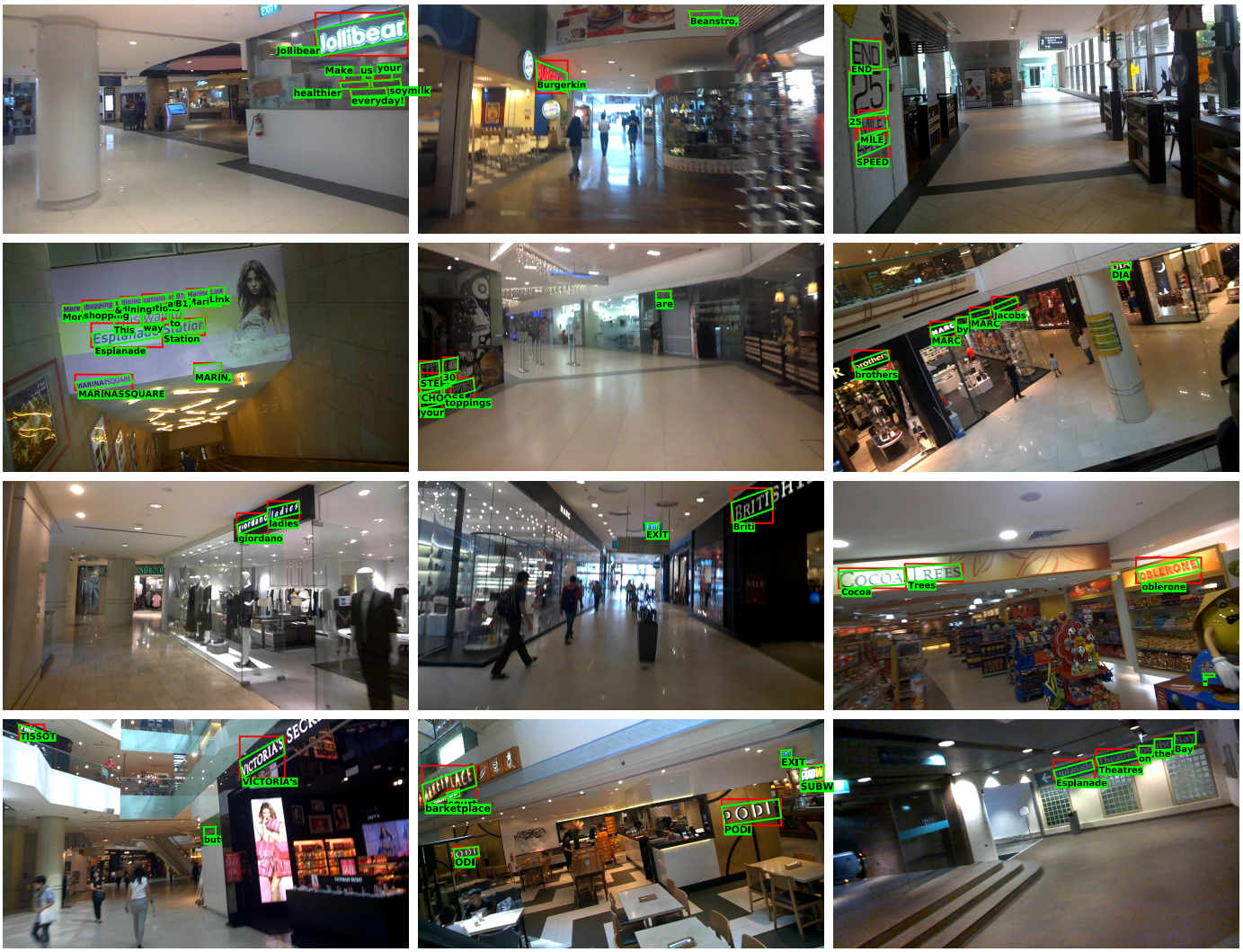


Fig. 14. End-to-end recognition results on IC15. Red boxes are detected by TextBoxes. Green polygons are the rectified detections.

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

[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[35] C. Lee, A. Bhardwaj, W. Di, V. Jagadeesh, and R. Piramuthu. Region-based discriminative feature pooling for scene text recognition. In *CVPR*, pages 4050–4057, 2014.

[36] C. Lee and S. Osindero. Recursive recurrent nets with attention modeling for OCR in the wild. In *CVPR*, pages 2231–2239, 2016.

[37] H. Li, P. Wang, and C. Shen. Towards end-to-end text spotting with convolutional recurrent neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[38] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. Textboxes: A fast text detector with a single deep neural network. In *AAAI*, pages 4161–4167, 2017.

[39] 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.

[40] S. Lu, B. M. Chen, and C. C. Ko. Perspective rectification of document images using fuzzy set and morphological operations. *Image Vision Comput.*, 23(5):541–553, 2005.

[41] S. Lu and C. L. Tan. Document flattening through grid modeling and regularization. In *18th International Conference on Pattern Recognition (ICPR 2006), 20-24 August 2006, Hong Kong, China*, pages 971–974, 2006.

[42] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, R. Young, K. Ashida, H. Nagai, M. Okamoto, H. Yamamoto, H. Miyao, J. Zhu, W. Ou, C. Wolf, J. Jolion, L. Todoran, M. Worring, and X. Lin. ICDAR 2003 robust reading competitions: entries, results, and future directions. *IJDAR*, 7(2-3):105–122, 2005.

[43] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image Vision Comput.*, 22(10):761–767, 2004.

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

[45] A. Mishra, K. Alahari, and C. V. Jawahar. Enhancing energy minimization framework for scene text recognition with top-down cues. *Computer Vision and Image Understanding*, 145:30–42, 2016.

[46] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *ACCV*, pages 770–783, 2010.

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

[48] L. Neumann and J. Matas. Real-time lexicon-free scene text localization and recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(9):1872–1885, 2016.

[49] T. Q. Phan, P. Shivakumara, S. Tian, and C. L. Tan. Recognizing text with perspective distortion in natural scenes. In *ICCV*, 2013.

[50] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*

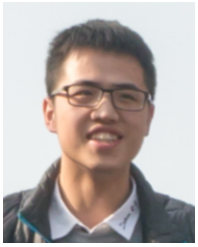
[51] A. Risnumawan, P. Shivakumara, C. S. Chan, and C. L. Tan. A robust arbitrary text detection system for natural scene images. *Expert Syst. Appl.*, 41(18):8027–8048, 2014.

[52] J. A. Rodríguez-Serrano, A. Gordo, and F. Perronnin. Label embedding: A frugal baseline for text recognition. *Int. J. Comput. Vision*, 113(3):193–207, 2015.

[53] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017.

[54] 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.*, 39(11):2298–2304, 2017.
[55] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai. Robust scene text recognition with automatic rectification. In *CVPR*, pages 4168–4176, 2016.
[56] B. Su and S. Lu. Accurate scene text recognition based on recurrent neural network. In *ACCV*, 2014.
[57] B. Su and S. Lu. Accurate recognition of words in scenes without character segmentation using recurrent neural network. *Pattern Recognition*, 63:397–405, 2017.
[58] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.
[59] J. Wang and X. Hu. Gated recurrent convolution neural network for OCR. In *NIPS*, pages 334–343, 2017.
[60] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *ICCV*, 2011.
[61] K. Wang and S. Belongie. Word spotting in the wild. In *ECCV*, 2010.
[62] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *ICPR*, 2012.
[63] J. J. Weinman, Z. Butler, D. Knoll, and J. Feild. Toward integrated scene text reading. *IEEE transactions on pattern analysis and machine intelligence*, 36(2):375–387, 2014.
[64] X. Yang, D. He, Z. Zhou, D. Kifer, and C. L. Giles. Learning to read irregular text with attention mechanisms. In *IJCAI*, pages 3280–3286, 2017.
[65] C. Yao, X. Bai, and W. Liu. A unified framework for multioriented text detection and recognition. *IEEE Transactions on Image Processing*, 23(11):4737–4749, 2014.
[66] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1083–1090. IEEE, 2012.
[67] C. Yao, X. Bai, B. Shi, and W. Liu. Strokelets: A learned multi-scale representation for scene text recognition. In *CVPR*, 2014.
[68] Q. Ye and D. S. Doermann. Text detection and recognition in imagery: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(7):1480–1500, 2015.
[69] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.
[70] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. *arXiv preprint arXiv:1604.04018*, 2016.
[71] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. EAST: an efficient and accurate scene text detector. In *CVPR*, pages 2642–2651, 2017.
[72] 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.

Baoguang Shi received his B.S. degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China in 2012, where he is currently a Ph.D. candidate. He was an intern at Microsoft Research Asia in 2014, and a visiting student at Cornell University from 2016 to 2017. His research interests include scene text detection and recognition, 3D shape recognition, and facial recognition.



Mingkun Yang received his B.S. degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), China in 2016. He is currently a Master student with the School of Electronic Information and Communications, HUST. His main research interests include fine-grained image classification and scene text recognition.

Xinggang Wang is an assistant professor of School of Electronics Information and Communications of HUST. He received his B.S. degree in Communication and Information System and PhD degree in Computer Vision both from HUST. He is a reviewer of IEEE Transaction on Cybernetics, Pattern Recognition, Computer Vision and Image Understanding, Neurocomputing, CVPR, ICCV and ECCV etc. His research interests include computer vision and machine learning.

Pengyuan Lyu received his B.S. degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), China in 2015. He is currently a Master student with the School of Electronic Information and Communications, HUST. His main research interests include scene text detection and recognition.

Cong Yao is currently with Megvii (Face++) Inc., Beijing, China. He received the B.S. and Ph.D. degrees in electronics and information engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2008 and 2014, respectively. He was a research intern at Microsoft Research Asia (MSRA), Beijing, China, from 2011 to 2012. He was a Visiting Research Scholar with Temple University, Philadelphia, PA, USA, in 2013. His research has focused on computer vision and machine learning, in particular, the area of text detection and recognition in natural images.

Xiang Bai received his B.S., M.S., and Ph.D. degrees from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2003, 2005, and 2009, respectively, all in electronics and information engineering. He is currently a Professor with the School of Electronic Information and Communications, HUST. He is also the Vice-director of the National Center of Anti-Counterfeiting Technology, HUST. His research interests include object recognition, shape analysis, scene text recognition and intelligent systems. He serves as an associate editor for Pattern Recognition , Pattern Recognition Letters, Neurocomputing and Frontiers of Computer Science.