Attention and Language Ensemble for Scene Text Recognition with Convolutional Sequence Modeling

Shancheng Fang[1,2], Hongtao Xie[3]*, Zheng-Jun Zha[3]
Nannan Sun[1,2], Jianlong Tan[1,2],Yongdong Zhang[3]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]School of Information Science and Technology, University of Science and Technology of China, Hefei, China
{fangshancheng,sunnannan,tanjianlong}@iie.ac.cn,{htxie,zhazj,zhyd73}@ustc.edu.cn

ABSTRACT

Recent dominant approaches for scene text recognition are mainly based on convolutional neural network (CNN) and recurrent neural network (RNN), where the CNN processes images and the RNN generates character sequences. Different from these methods, we propose an attention-based architecture[1] which is completely based on CNNs. The distinctive characteristics of our method include: (1) the method follows encoder-decoder architecture, in which the encoder is a two-dimensional residual CNN and the decoder is a deep one-dimensional CNN. (2) An attention module that captures visual cues, and a language module that models linguistic rules are designed equally in the decoder. Therefore the attention and language can be viewed as an ensemble to boost predictions jointly. (3) Instead of using a single loss from language aspect, multiple losses from attention and language are accumulated for training the networks in an end-to-end way. We conduct experiments on standard datasets for scene text recognition, including *Street View Text*, *IIIT5K* and *ICDAR* datasets. The experimental results show our CNN-based method has achieved state-of-the-art performance on several benchmark datasets, even without the use of RNN.

KEYWORDS

Text recognition; convolutional neural networks; multi-level supervised information; attention model

ACM Reference Format:
Shancheng Fang, Hongtao Xie, Zheng-Jun Zha, Nannan Sun, Jianlong Tan, Yongdong Zhang. 2018. Attention and Language Ensemble for Scene Text Recognition with Convolutional Sequence Modeling. In *2018 ACM Multimedia Conference (MM '18), October 22–26, 2018, Seoul, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240508.3240571

1 INTRODUCTION

Text in natural scene images usually contains rich and valuable semantic information. Thus recognizing scene text is crucial to many real-world applications, such as content-based image retrieval [45, 46], text translation from photos and autonomous car, etc. Despite remarkable progress on scene text recognition, it remains a challenging task due to complex image background, variation of text styles and low image quality, etc.

Recently, deep learning technique has been used for scene text recognition with encouraging performance [24, 25]. Most prevalent approaches are based on the combination of convolutional neural network (CNN) and recurrent neural network (RNN) [6, 11, 13, 29, 39, 44]. The CNN is regarded as a feature extractor or an encoder to characterize visual cues within images. The RNN, such as long short-term memory (LSTM) [21] and gated recurrent unit (GRU) [7], generates character sequences on the basis of CNN features.

Although RNNs possess good capacity in modeling character sequences, some inherent limitations of RNNs hinder their practical application on scene text recognition task. For example, due to the sophisticated structure and expensive computation of RNNs, stacking RNNs into a deep model for better language modeling is impractical [9, 29]. Moreover, the gradient vanishing/exploding problem obstructs the optimal training of RNNs sometimes [5]. In contrast, modeling language sequences using CNNs has its own superiority. By stacking convolutional layers [8, 12], the hierarchical representations can also effectively deal with long sequences (e.g. the sequence of character features in text recognition). Besides, convolutional model can be parallelized efficiently and better exploit GPU resource as compared to recurrent model [8].

Attention mechanism usually follows the encoder-decoder architecture, which has been adopted by state-of-the-art methods with impressive performance [6, 29, 39]. By replacing the RNN encoder with CNN encoder to extract image features, these methods directly apply the vanilla attention designed for neural machine translation (NMT) [4, 31] to scene text recognition. Thus attention in the decoder can focus on interest regions in images, offering visual cues for modeling character sequences. In scene text recognition, text should be recognized taking into account both visual cues and linguistic rules in the same time. However, under the vanilla attention mechanism, the visual cues become auxiliary information for language modeling, as character predictions uniquely come from language modeling (Figure 3(a)). Besides, for the reason that language modeling lies in the core position in NMT, only the loss for measuring linguistic rules is considered for training the networks. Hence, there is a practical demand for joint exploration of visual cues and linguistic rules for boosting scene text recognition.

Motivated by the above observations, we propose a novel architecture, which is completely based on CNNs, to fully make use of

*Corresponding Author
[1]The source code is available at https://github.com/FangShancheng/conv-ensemble-str

MM '18, October 22–26, 2018, Seoul, Republic of Korea

ACM ISBN 978-1-4503-5665-7/18/10…$15.00
https://doi.org/10.1145/3240508.3240571

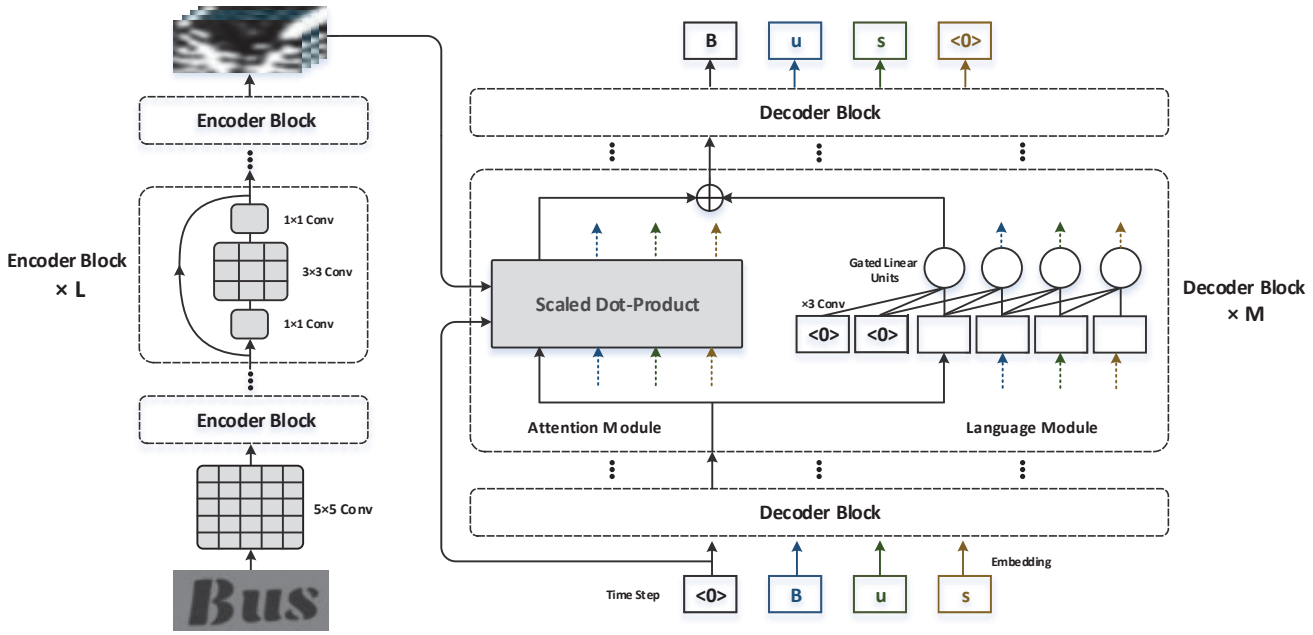

Figure 1: The architecture of our method during training. In the side of encoder (left), we apply residual convolutions to encode images. After a base convolutional layer, several residual blocks are stacked in the encoder. In the side of decoder (right), stacked decoder blocks predict the output sequences after embedding input symbols. Inside each decoder block, an attention module whose main operations are scaled dot-product, and a language module based on gated convolutions are designed equally as an ensemble to boost predictions.

visual cues for scene text recognition. Figure 1 shows the architecture of our method. The proposed architecture consists of an encoder that extracts abstractive image representation, and a decoder that generates character sequences. The encoder is a residual convolutional network. The decoder, different from typical RNN-based works [6, 29, 39], is also based on the deep convolutional network. Inside each decoder block, an attention module focuses on significant visual cues for each character from encoder feature maps, and a language module aims to model the linguistic rules in character level. Due to the superiority of CNN-based decoder, we can easily stack attention and language modules into a deep model, providing abstractive representations in higher level. Besides, different from the vanilla attention that underestimates the role of visual cues when making predictions, visual cues and linguistic rules are of the same importance in our method. Therefore, the attention and language modules are designed equally as an ensemble to boost predictions. Furthermore, we introduce multiple losses from visual cues and linguistic rules to train our networks, which furnish stronger supervised information than only using a single loss from language aspect [6, 29, 38, 39].

In summary, the contributions presented in the paper are as follows:

- We introduce an architecture based entirely on CNN for scene text recognition. Different from the RNN-based architectures, our method can easily stack the attention and language modules layer by layer to deal with long-term dependencies of character sequences.
- A novel attention and language ensemble method is proposed to boost predictions jointly, which recognizes characters considering both visual cues and linguistic rules. Based on the ensemble, we design multiple losses from attention module and language module to train our networks in an end-to-end way.
- We experimentally validate the effectiveness of our method for unconstrained and constrained text recognition. The proposed architecture achieves state-of-the-art or highly-competitive performance on several challenging datasets.

The rest of this paper is organized as follows. In Section 2, we briefly analyze the related works. Section 3 describes our method from the aspects of encoder and decoder. We also introduce attention and language ensemble in this section. The evaluation datasets and extensive experiments are elaborated in Section 4. Finally, we give the conclusion and further discussion in Section 5.

2 RELATED WORKS

Traditionally, methods for scene text recognition are based on either character recognition or word recognition. Character based methods typically localize individual characters using a sliding window [26, 43] or connected components [34] in advance. Then character classifiers combining different feature descriptors are applied, such as CNN [26], HOG features with random ferns [43] and mid-level features, strokelets, with random forest [47]. After that, characters are integrated to generate a full word. Usually, a fixed lexicon is used to group the characters into words in this step. Word based methods, however, adopt a holistic representation over the input

images [10, 25, 35]. For example, based on aggregated Fisher Vectors [35] and a structured SVM framework, Rodríguez-Serrano and Perronnin [36] propose a method that jointly creates word image and text embedding. Jaderberg et al. [25] apply a CNN to generate the fixed representation for an image. In this case, text recognition is regarded as a 90k-class problem. Though these methods report impressive results, they are limited in some cases since their outputs are constrained in a fixed lexicon.

Recently, a group of studies pay their attention on unconstrained text recognition [24]. Unconstrained recognition is more challenging than constrained recognition since there is no pre-defined lexicon, and the predictions can be previously unseen or meaning-less text. A typical work is that, Jaderberg et al. [24] propose a system which incorporates CNNs with Conditional Random Field (CRF). The system is composed of two separate CNNs. One CNN provides unigram sequences and the other gives n-gram language statistics for the CRF.

To effectively model language sequences, RNN is introduced for text recognition. The advantages of RNN include: 1) No need to know the explicit locations of characters in the images, which means that it transcribes the character sequences without character segmentation or sliding windows, since they are error accumulated and computation intensive. 2) No need to calculate the expensive n-gram language statistic. To integrate CNN and RNN for text recognition task, the Connectionist Temporal Classification (CTC) loss [16] is introduced. For example, Shi et al. [38] propose a network architecture that utilizes CNN to extract features and LSTM to translate sequences into text, which is trained through CTC loss in an end-to-end way.

Another way to combine CNN and RNN is adopting attention models [4, 31]. Attention models follow the encoder-decoder architecture. At each output step, the decoder focuses on a specific part of the encoder. In the field of text recognition, encoder is generally a CNN and decoder is an RNN [6, 13, 29, 39, 44]. Lee and Osindero [29] design a convolutional neural network with recursive recurrent connections to generate image features. Then, an attention model is used to selectively exploit the image features, which filters irrelevant features by the attention scores in each output step. Shi et al. [39] use a multi-layer CNN and a two-layer bidirectional LSTM as the encoder, then GRU decoder with attention mechanism produces final character sequences. Ghosh et al. [13] show visual attention can effectively boost the performance in a LSTM-based text recognition model. Wojna et al. [44] propose spatial attention for the recognition of French street view image. Under the spatial attention mechanism, the LSTM can be aware of the locations of the convolutional features. Cheng et al. [6] design a focusing network to adjust attention by evaluating whether an attention model pays attention properly on the target areas in the images. These works above validate the superiority of the attention model, while all of them rely on RNN-based language modeling. Our method inherits the advantages of RNN and applies a more flexible CNN as the decoder to model character sequences.

More recent works in language modeling [8] and NMT [12, 42] have shown that other forms of neural network can also outperform the strong RNN on language modeling. Dauphin et al. [8] introduce gated linear units (GLU) for convolutional networks to ease gradient propagation. The gated convolution not only performs better than LSTM, but also has simper architecture, which can be parallelized easily during training. Vaswani et al. [42] further extend the work [8] to sequence-to-sequence learning. In their work, convolutional networks are equipped with GLU, and attention is applied in each decoder layer. Besides, Gehring et al. [12] also report a state-of-the-art translation network which is with strong attention in both encoder and decoder. Surprisingly it dispenses with recurrence and convolution entirely. Our work is inspired by the method of Gehring et al. [12]. However, we view attention focusing and language modeling quite different from previous works [12, 42]. In this paper, visual cues and linguistic rules are of the same importance. We show that CNN-based language model has a flexible structure allowing attention and language ensemble, which is competitive to RNN-based method in the field of scene text recognition.

3 THE PROPOSED METHOD

3.1 Overview

We introduce the math tokens and formulate the proposed method in the following way. Given a cropped text image I, $s \cdot H$ and $s \cdot W$ are the height and width of I respectively, and s is the stride of the encoder. During the training phase, character labels $\mathbf{l} = (l_1, \ldots, l_n)$ of the image are also given and n is the length of labels. In our method, a CNN as the encoder firstly processes image I into feature map $\mathbf{X} \in \mathbb{R}^{H \times W \times d}$, where d is the dimension of the element $\mathbf{x}_{i,j}$ in $\mathbf{X}$, and i, j are the indices from 1 to H, W. The decoder, also CNN-based but one-dimensional, takes $\mathbf{X}$ and generates the predicted character sequence $\mathbf{y} = (y_1, \ldots, y_N)$, in which the distribution of character sequence $P(\mathbf{y}) = p(y_1, \ldots, y_N) = \prod_{k=1}^{N} p(y_k | y_{k-1}, \ldots, y_1)$ and N is the predicted length of the character sequence.

Figure 1 presents the architecture of our method during training phase. Overall, in the hand of encoder, we process images following the ideas of residual networks [20], which is widely used in image recognition. In the other hand of decoder, we model the character sequences by stacking attention module and language module, which jointly make decision from visual and linguistic perspectives.

Concretely, the first layer of the encoder is a base convolutional layer which converts images to feature maps in specific distribution. Then a deep encoder is built by stacking residual blocks [20]. In this paper, the two terms block and layer are used interchangeably. Also, there is a similar structure in the decoder. After embedding input symbols, the stacked decoder blocks predict the output sequences. Inside each decoder block, an attention module and a language module are designed equally as an ensemble. The attention module focuses on interest region from encoder feature maps, whose main operations are scaled dot-product [42]. The language module, based on gated convolutional layers [8], aims to model the language sequences in character level. In addition, we use batch normalization [22] in the encoder and layer normalization [3] in the decoder to keep variances stable across the main nodes of the networks. In the following sections, we will detail the design of our method from the aspects of encoder and decoder.

3.2 Residual Encoder

The encoder converts images to abstract feature maps, which will be used in attention module of the decoder (Section 3.3.3). Detailed architecture of the encoder is given in Table 1. In the beginning

Table 1: The architecture of encoder.

layer name	conv1	conv2_x	conv3_x	conv4_x	conv5_x
input size	32 × 100	16 × 50	8 × 25	8 × 25	8 × 25
block	5×5, 16, stride 2 BN, ReLU	[1×1, 16 BN, ReLU 3×3, 16 BN, ReLU 1×1, 64 Shortcut BN, ReLU] ×n_1	[1×1, 32 BN, ReLU 3×3, 32 BN, ReLU 1×1, 128 Shortcut BN, ReLU] ×n_2	[1×1, 64 BN, ReLU 3×3, 64 BN, ReLU 1×1, 256 Shortcut BN, ReLU] ×n_3	[1×1, 128 BN, ReLU 3×3, 128 BN, ReLU 1×1, 512 Shortcut BN, [ReLU]] ×n_4

Note: BN is the batch normalization [22]. n_1, n_2, n_3 and n_4 are the numbers of the repetitive blocks.

Table 2: The architecture of decoder.

layer name	embedding	conv_x	logit
dimension	d	d	f
block	Embedding	[Conv1D,GLU,Shortcut,LN Attention, Shortcut, LN Add] ×n_5	Linear Softmax

Note: LN is layer normalization [3]. d and f are the output dimensions of input vectors, and f is equal to the characters number. n_5 is the number of the repetitive blocks.

we normalize images to height 32 and width 100. The first layer is a base convolutional layer followed by batch normalization with activation ReLU [33], which normalizes image to a stable variance. Then several blocks that have a bottleneck structure [19] are stacked. Downsampling is performed after conv1 and conv2_x with a stride of 2. Thus the output shape of the feature map $\mathbf{X}$ is 8×25. Note that only batch normalization (i.e. no ReLU) is applied in the last block, we do this to keep the variance around 1 as the feature maps will be then fed to decoder for attention.

We adopt the augmentation strategies as the work [44] during training: images are first cropped to contain at least 0.8 area of the original image. The cropped area of the image has an aspect ratio between 0.8 and 1.2. Then the cropped area is resized to the size 32×100 with random interpolation procedure. Finally, we randomly change the contras, hue, brightness and saturation of the images.

3.3 Ensemble Decoder

Table 2 gives the structure of the decoder. In the bottom of the decoder is an embedding layer. Following that are the decoder blocks consisting of language module and attention module. Finally, we use linear transform to make predictions over all the possible characters.

When training our model, the inputs to decoder (source labels) are the concatenation of a start token and character labels $\mathbf{l}$, and ground truth labels (target labels) are the concatenation of $\mathbf{l}$ and an end token, which follows the ideas of auto-regressive [15]. Both the start token and end token are $\langle 0 \rangle$ in our method, namely the source labels is $\widetilde{\mathbf{l}} = (\langle 0 \rangle, l_1, \ldots, l_n)$ and target labels is $\widehat{\mathbf{l}} = (l_1, \ldots, l_n, \langle 0 \rangle)$. An example is shown in Figure 1.

3.3.1 ***Label and Position Embeddings.*** As the common recurrent based model, source labels are embedded in the beginning of the decoder. That is to say for a source label l_k in $\widetilde{\mathbf{l}}$, we use learned embeddings to convert it to a vector $\mathbf{v}_k \in \mathbb{R}^d$. To make use of position information, the absolute position of l_k in $\widetilde{\mathbf{l}}$ is embedded to $\mathbf{u}_k \in \mathbb{R}^d$. Then the final embedding vector $\mathbf{s}_k^0 \in \mathbb{R}^d$ is obtained by combining $\mathbf{u}_k$ and $\mathbf{v}_k$ through element-wise addition.

3.3.2 ***Convolutional Language Modeling.*** The key component of language module is one-dimensional convolutions followed by GLU activation, which aims to predict the next character given the source label. The GLU controls information flow by selecting features through sigmoid function, which is demonstrated to be useful for language modeling [8, 12]. We illustrate the employment of convolution and GLU in Figure 2. Given an input vector $\mathbf{s}_k^m$ in the m-th decoder block[2], output vector $\mathbf{t}_k^m$ is obtained by the following formula:

$$\mathbf{Y} = [\mathbf{s}_{k-2}, \mathbf{s}_{k-1}, \mathbf{s}_k],$$
$$\mathbf{t}_k = \sigma(\mathbf{Y} * \mathbf{W}^a + \mathbf{b}^a) \otimes (\mathbf{Y} * \mathbf{W}^b + \mathbf{b}^b),$$

where $\mathbf{s}_{k-2}, \mathbf{s}_{k-1}$ are corresponding history vectors for $\mathbf{s}_k$, which are the representations of source labels l_{k-2} and l_{k-1} in the previous time steps. As the size of convolution kernel we used in this work is 3, the input matrix $\mathbf{Y}$ has dimension of $d \times 3$. $\mathbf{W}^a \in \mathbb{R}^{3 \times d \times d}$ and $\mathbf{W}^b \in \mathbb{R}^{3 \times d \times d}$ are the convolutional weights which keep the input dimension and output dimension unchangeable. $\mathbf{b}^a \in \mathbb{R}^d$ and $\mathbf{b}^b \in \mathbb{R}^d$ are the learned biases. $*$ is the operation of one-dimensional convolution. $\otimes$ is the element-wise product and σ is the sigmoid function.

After that, shortcut connection and layer normalization are applied to obtain the output language vector:

$$\mathbf{t}_k = \text{norm}(\mathbf{t}_k + \mathbf{s}_k).$$

Note that to avoid accessing future information in the subsequent positions, the convolutions take history vectors from $k-2$ to k as input. Specially, when $k <= 2$, zero-vectors are used as the history vectors, as presented in Figure 1. Besides, we apply layer normalization rather batch normalization in the decoder, which not only prevents attending future information but also preserves the variances of activation. The layer normalization is placed in the main nodes of the network, as we try to ensure the variance around 1 throughout the network. Although we have only described the procedure from source label l_k to $\mathbf{t}_k$, all the source labels $\widetilde{\mathbf{l}} = (\langle 0 \rangle, \ldots, l_k, \ldots, l_n)$ can be processed like l_k in parallel during training using the means of auto-regressive.

[2] $m = 1, \ldots, M$. To simplify expressions, m is omitted in the following descriptions unless necessary.

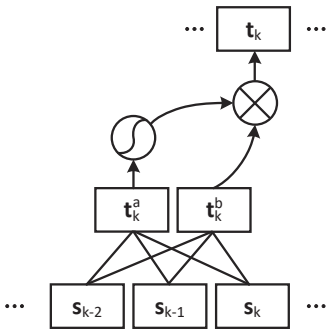

Figure 2: Illustration of one-dimensional convolution with gated linear unit for an input vector s_k, and the output vector is t_k. Two history vectors s_{k-1}, s_{k-2} are used as the size of convolutional kernel is 3. All the vectors in this figure have the same dimension d.

3.3.3 ***Image Focusing with Attention Mechanism***. The attention module predicts the next characters by focusing on a significant part from image feature maps. An attention function can be interpreted as mapping a query vector to an output vector from a set of key-value pairs [42]. In our work, the query vector is obtained as follows:

$$\mathbf{q}_k = \text{linear}(\mathbf{s}_k + \mathbf{s}_k^0),$$

where $\mathbf{s}_k$ is the input vector of the m-th decoder block, $\mathbf{s}_k^0$ is the embedding vector from the first embedding layer introduced in Section 3.3.1, and linear is a linear transform function that tries to project $\mathbf{s}_k + \mathbf{s}_k^0$ from language space to image space. Besides, in this paper both the key vector and the value vector are the element $\mathbf{x}_{i,j}$ of $\mathbf{X}$.

To compute attention scores and attention outputs, we apply scaled dot-product between the query vector and the key vector. The scaled dot-product is demonstrated to be a simple but effective method [12, 42]. Each element of the attention scores is firstly calculated as follows:

$$b_{i,j,k} = \frac{\mathbf{x}_{i,j}^T \cdot \mathbf{q}_k}{\sqrt{d}},$$

where $\sqrt{d}$ is a scaling factor. Then softmax is applied and the attention score is obtained:

$$a_{i,j,k} = \frac{\exp(b_{i,j,k})}{\sum_{o=1,p=1}^{H,W} \exp(b_{o,p,k})}.$$

After that, the attention output for $\mathbf{s}_k$ is a weighted sum over the value vectors:

$$\mathbf{c}_k = \sum_{i=1,j=1}^{H,W} a_{i,j,k} \cdot \mathbf{x}_{i,j}.$$

Finally, another linear transform is used to project $\mathbf{c}_k$ back to language space. After shortcut connection and layer normalization, the output attention vector is obtained:

$$\mathbf{c}_k = \text{norm}(\text{linear}(\mathbf{c}_k) + \mathbf{s}_k).$$

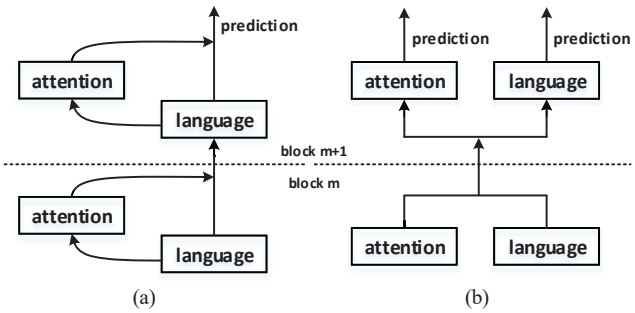

Figure 3: (a) Vanilla attention. (b) Our attention and language ensemble method.

3.3.4 ***Attention and Language Ensemble with Multiple Losses***. For $m = 1, \ldots, M-1$, output vector of the m-th block is the combination of attention vector $\mathbf{c}_k^m$ and language vector $\mathbf{t}_k^m$:

$$\mathbf{s}_k^{m+1} = \mathbf{c}_k^m + \mathbf{t}_k^m.$$

On the top of the decoder are a linear transform and a softmax function that convert the attention vector and the language vector to predicted characters separately:

$$p_a(y_k|y_{k-1}, \ldots, y_1) = \text{softmax}(\mathbf{W}^c \mathbf{c}_k^M + \mathbf{b}^c),$$
$$p_l(y_k|y_{k-1}, \ldots, y_1) = \text{softmax}(\mathbf{W}^t \mathbf{t}_k^M + \mathbf{b}^t),$$

where $\mathbf{W}^c, \mathbf{W}^t \in \mathbb{R}^{d \times f}$ and $\mathbf{b}^c, \mathbf{b}^t \in \mathbb{R}^f$.

Following that, we ensemble the predictions from attention module and language module by simply applying element-wise addition:

$$p(y_k|y_{k-1}, \ldots, y_1) = p_a + p_l.$$

Then beam search is applied to get the final predictions based on this distribution.

During training, the cross entropy losses of attention module and language module are considered together:

$$\mathcal{L} = \mathcal{L}_a + \mathcal{L}_l,$$
$$\mathcal{L}_a = \sum_{t=1}^{N} \log p_a(y_t|y_{t-1}, \ldots, y_1),$$
$$\mathcal{L}_l = \sum_{t=1}^{N} \log p_l(y_t|y_{t-1}, \ldots, y_1),$$

where y_t is predicted score of ground truth label in location t. We do not average the loss across predicted length N as we find it hurt the performance.

Different from the common attention-based methods [6, 12, 42], in which the input of attention focusing is the output of language modeling (usually implemented by RNN or other kinds of neural networks), our method shares the same input for attention focusing and language modeling, and their outputs are regarded as an ensemble to obtain an enhanced decision. See Figure 3 for intuitive comparison about vanilla attention (introduced in [12]) and our method. In general, vanilla attention computes attention module and language module unequally. However, our method adopts an equal and parallel strategy. The advantages of our method include:

1) As the attention module and language module lie in the same position in a decoder block, multiple losses from visual cues (attention module) and linguistic rules (language module) can be computed accumulatively, which provide additional supervised information than only one kind of loss.

2) The equal and parallel strategy essentially reduces the depth of attention layers[3], which mitigates the vanishing gradient problem caused by a deep architecture.

3) Our method has better interpretability, as visual cues and linguistic rules are contributed to predictions equally and separately. In contrast, vanilla attention just stacks attention and language in a meaningless way.

4 EXPERIMENTS

4.1 Datasets and Evaluation Protocols

The datasets we use for training and evaluation are six standard benchmark datasets: Synth90k [23], Street View Text [43], IIIT 5K-Word [32], ICDAR 2003 [30], ICDAR 2013 [28] and ICDAR 2015 [27]. Unless additional statement, we follow the experiment settings in [24]. Thus the networks are trained on Synth90k dataset and evaluated on other five datasets. In addition, another training dataset SynthText is used only for additional comparison. The summaries of these datasets are as follows:

Synth90k: The dataset has 9 million cropped word images which are generated synthetically. Among them about 7.2 million images are training images, and 0.9 million images are validation images. Only the training images are used to train the network and parameters are selected via validation set.

Street View Text (SVT): There are 647 word images cropped from 250 full scene test images, which are download from Google Street View. Each word image contains a 50-word lexicon defined by [43].

IIIT 5K-Word: The test dataset contains 3000 cropped word images for testing, which are collected from Google image search. Each word image is associated with a 50-word lexicon and a 1k-word lexicon.

ICDAR 2003 (IC03): The test dataset contains 251 full images. We follow [43] to crop 860 word images and construct the lexicons. More specifically, images that either contain non-alphanumeric characters or have less than three characters are ignored. Also, a 50-word lexicon and a full lexicon that includes all ground truth words are built for constrained evaluation.

ICDAR 2013 (IC13): The test dataset is constructed using the same strategies as ICDAR 2003 dataset. It contains 1015 cropped word images totally.

ICDAR 2015 (IC15): There are 2077 cropped images in test dataset. By filtering non-alphanumeric characters or irregular text, 1811 images are finally kept for fair comparison [6].

SynthText: This is a synthetic dataset released by Gupta et al. [17] that text is naturally blended in 80 thousand scene images. We follow [6] to obtain 4 million cropped word images which only contain alphanumeric characters with length greater than 3.

We adopt the standard evaluation protocols as most of the previous works [13, 24, 29, 39, 43]. That is, we perform text recognition on the words that contain only alphanumeric characters and at least three characters.

[3]From the perspective of network depth, the depth of an unequal strategy is 2 times that of an equal one.

Table 3: Recognition accuracy (%) of variants with different convolutional layers.

Encoder					Decoder	SVT	IIIT5K	IC03	IC13
total	n_1	n_2	n_3	n_4	n_5				
37	2	2	2	6	5	84.1	81.0	91.4	90.2
25	2	2	2	2	5	81.8	79.5	89.5	88.8
37	3	3	3	3	5	83.3	81.2	90.5	89.7
49	3	3	3	7	5	82.8	80.7	90.8	89.3
37	2	2	2	6	3	82.5	80.4	89.7	89.1
37	2	2	2	6	4	83.3	80.8	90.5	89.6
37	2	2	2	6	6	83.9	**82.0**	**91.9**	**90.7**
37	2	2	2	6	7	**84.4**	81.5	91.5	90.2

Table 4: Recognition accuracy (%) of variants with different components.

Methods	SVT	IIIT5K	IC03	IC13
Baseline	84.1	**81.0**	91.4	**90.2**
Without position embedding	83.5	80.4	**91.6**	90.0
ReLU activation	82.4	79.8	90.6	89.5
Single loss	82.7	80.3	90.1	88.0
Vanilla attention	**84.3**	80.6	91.0	89.8

4.2 Implementation Details

We use Kaiming weight initialization [18] for both the encoder and decoder. Besides, weight normalization [37] is employed for all decoder blocks except embedding matrices. The dimension d of decoder is 512 (Section 3.1), and the characters number f is 63 (62 alphanumeric characters and an end token $\langle 0 \rangle$. Section 3.3). We train our model using Nesterov's [41] accelerated gradient method with a momentum value of 0.9 and batch size 128. The initial learning rate is 0.01 and is reduced by a factor of 0.1 after the loss stops decreasing. We also clip the gradients at a magnitude of 20. Beam search uses a width of 1 (greedy) during training and of 5 during inference.

4.3 Depth of Convolutional Layers

In this section, we explore the impact on accuracy caused by the depth of convolutional layers both in encoder and decoder. Table 3 gives the configurations of different convolutional layers. A variant with 37 encoder layers and 5 decoder layers is used as the baseline. Based on it we increase or decrease the number of encoder layers while fixing the number of decoder layers, and so does the decoder. All the other configurations are the same except the network depth. Note that we do not test shallow encoder (less than 25 layers) and decoder (less than 3 layers), as we all know a deep structure boosts the accuracy.

Table 3 shows the performances when changing the number of layers in the encoder and the decoder. As we can see, the accuracy benefits from increasing number of layers both in encoder and decoder. However, it would be saturate when networks get deep

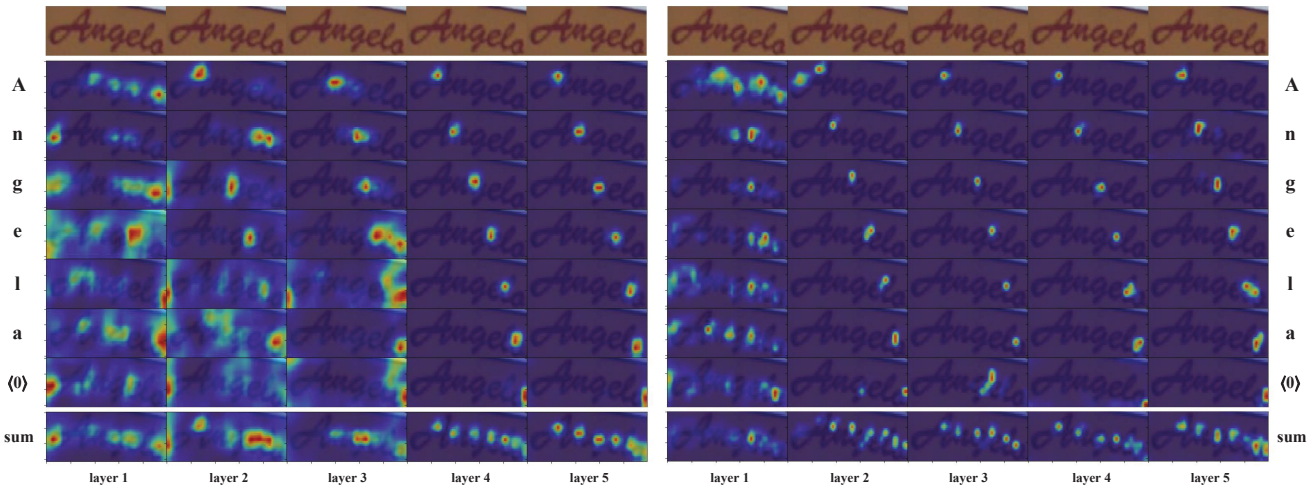

Figure 4: Visualization of attention scores for two five-layer decoders. (a) Vanilla attention. (b) Our attention and language ensemble method.

enough. We roughly stack encoder layers and observe $\sim$ 37 layers have best performance. Then we adjust distribution of n_1, n_2, n_3, n_4 and find a bigger n_4 tends to perform better. More importantly, it reduces computation operations. As for the decoder, $\sim$ 6 layers get promising accuracy and speed. In summary, a variant with 37 layers encoder and 6 layers decoder is chosen as our final configuration.

Notice that deeper networks have bigger receptive fields, we conjecture this point heavily influences the accuracy. An encoder with 25 layers only owns a receptive field of about 50 pixels, while a 37 layers or deeper encoder has covered the original image (width 100), which fits the accuracy curve boosted by the network depth. Similarly, as the lengths of common words are less than 13, a decoder with 6 layers can see all the characters in a word. If we continue to add layers, it brings negligible benefit and increases the overhead.

4.4 Ablation Studies

In this section, we conduct experiments to evaluate the contributions brought by different components. Experimental results are presented in Table 4. The baseline in Table 4 is the same as the one introduced in Section 4.3. To validate the effectiveness of position embedding (Section 3.3.1), we train a counterpart model that only label embedding is included. As the result shown in the second row of Table 4, position embedding indeed improves the accuracy on most of the datasets except IC03 dataset, but the gap on IC03 dataset is small. The position embeddings force our model to learn absolute positions, thus our model is sensitive to features in specific location, which may decrease error predictions.

To evaluate the effectiveness of gated linear unit (Section 3.3.2), we replace all the GLU activation with ReLU activation. The third row in Table 4 shows that an ReLU variant performs worse than our baseline model on all the datasets. We also conjecture the benefit may come from parameters number, since the parameters number of a convolution layer equipped with GLU is $2 \times 3d^2$ while with ReLU is $3d^2$.

The fourth row in Table 4 is the result of a single loss variant. That is to say only $\mathcal{L}_l$ is used as the final loss function rather than $\mathcal{L}$ (Section 3.3.4). We find combining language loss and image loss together significantly improves the accuracy, which demonstrates the superiority of our ensemble method.

We continue to validate the ensemble strategy thoroughly. The last row in Table 4 gives the accuracy of a variant using the vanilla attention mechanism, which is introduced in [12]. This variant applies attention module and language module unequally (Figure 3(a)). From the comparison we can see, our baseline achieves better performance than the counterpart in most cases except SVT dataset. Furthermore, we visualize attention scores of the vanilla attention and the proposed method in Figure 4. Both the two decoders are composed of 5 layers. For the one with vanilla attention (Figure 4(a)), we observe that attention scores in the bottom layers (mainly refer to layer 1,2,3) capture very rough locations of all the characters. However, the proposed method obtains clearer attention scores from the second layer to the last layer (Figure 4(b)). Our equal and parallel structure eases gradient propagation and this may be a main reason why our method outperforms the vanilla attention.

In addition, from the visualization (Figure 4(b)) we can understand how attention module works: 1) the bottom layers tend to capture rough context for each character step, while the top layers focus on accurate location of the decoded character. 2) From the sum of the attention scores in each time step, we notice that attention scores can capture spatial location for each character, which is not limited to horizontal or vertical directions. Thus the proposed method can also handle curved text robustly. 3) When the attention looks at the end of a word image, the decoder generates stop token and the word is finally recognized.

4.5 Comparison with Other Methods

Recent works for text recognition pay their attention in unconstrained text recognition [24], which is more challenging than constrained text recognition, since the latter limits outputs to a

Table 5: Comparisons with the state-of-the-art methods on general recognition benchmarks.

Type	Method	SVT		IIIT5K			IC03			IC13	IC15
		50	None	50	1k	None	50	Full	None	None	None
I	Wang et al. [43] (ABBYY)	35.0	-	24.3	-	-	56.0	55.0	-	-	-
	Wang et al. [43]	57.0	-	-	-	-	76.0	62.0	-	-	-
	Mishra et al. [32]	73.2	-	64.1	57.5	-	81.8	67.8	-	-	-
	Alsharif and Pineau [2]	74.3	-	-	-	-	93.1	88.6	-	-	-
	Almazán et al. [1]	89.2	-	91.2	82.1	-	-	-	-	-	-
	Yao et al. [47]	75.9	-	80.2	69.3	-	88.5	80.3	-	-	-
	Jaderberg et al. [26]	86.1	-	-	-	-	96.2	91.5	-	-	-
	Su and Lu [40]	83.0	-	-	-	-	92.0	82.0	-	-	-
	Gordo [14]	91.8	-	93.3	86.6	-	-	-	-	-	-
	Jaderberg et al. [25] (DICT)	95.4	80.7	97.1	92.7	-	98.7	98.6	93.1	90.8	-
II	Jaderberg et al. [24]	93.2	71.7	95.5	89.6	-	97.8	97.0	89.6	81.8	-
	Shi et al. [38]	96.4	80.8	97.6	94.4	78.2	**98.7**	97.6	89.4	86.7	-
	Shi et al. [39]	95.5	81.9	96.2	93.8	81.9	98.3	96.2	90.1	88.6	-
	Lee and Osindero [29]	96.3	80.7	96.8	94.4	78.4	97.9	97.0	88.7	90.0	-
	Ghosh et al. [13]	95.2	80.4	-	-	-	95.7	94.1	**92.6**	-	-
	Ghosh et al. [13] (without ELM)	91.7	75.1	-	-	-	93.4	91.0	89.3	-	-
	Cheng et al. [6]	95.7	82.2	**98.9**	**96.8**	**83.7**	98.5	96.7	91.5	89.4	63.3
	Our method	**96.6**	**83.9**	97.5	94.8	82.0	98.5	**97.8**	91.9	**90.7**	**64.1**
III	Cheng et al. [6]	97.1	85.9	**99.3**	**97.5**	**87.4**	99.2	97.3	94.2	93.3	70.6
	Our method	**97.8**	**86.7**	98.5	96.8	86.7	**99.3**	**98.4**	**94.8**	**93.5**	**71.2**

Note: 50, 1k and Full are lexicon sizes. None denotes no lexicon used. Type I is constrained methods. Type II is unconstrained methods trained on Synth90k dataset. Type III is unconstrained methods trained on Synth90k and SynthText datasets. ELM means explicit language model.

specific lexicon. Our work also focuses on unconstrained text recognition. Therefore, we conduct two type comparisons according to different training datasets. Firstly, we purely train our model on Synth90k dataset, which follows most of the previous works [13, 24, 29, 38, 39]. See the results of Type II in Table 5. Our method has highly-competitive performance than the state-of-the-art methods, even without the use of RNN. Specifically, we outperform other methods by a large margin of 1.7% on SVT dataset, 0.7% on ICDAR 2013 dataset, and 0.8% on ICDAR 2015 dataset. On IIIT5K dataset, we are in the second place only lower than the method of Cheng et al. [6]. On ICDAR 2003 dataset, our accuracy is better than the other methods except the method of Ghosh et al. [13]. However, Ghosh et al. [13] improve their accuracy significantly by using an explicit language model, which is model-independent and computationally expensive. Different from the RNN-based methods [13, 29, 38, 39], our method uses a novel convolutional decoder and can still achieve impressive performance.

In addition, to fairly compare our method with [6], we continue training our model on both Synth90k and SynthText datasets. As can be seen from Type III in Table 5, our method outperforms [6] on most of the datasets except IIIT5K dataset. Note that both [6] and our method apply a deep residual convolutional network as the encoder, which indicates the superiority of our convolutional decoder with attention and language ensemble on extremely large training dataset.

To fully validate the performance of our method, we also have experiments on constrained text recognition. Given a pre-defined lexicon, after obtaining the conditional probability distributions from unconstrained text recognition, we calculate the probabilities over all lexicon words. Then the one with the highest probability is selected as constrained output. As the results shown in Table 5, our method outperforms other methods on SVT-50 and IC03-FULL datasets due to the remarkable performance in unconstrained recognition. In addition, our accuracy is comparable to state-of-the-art methods on IIIT5K and IC03-50 datasets.

5 CONCLUSION AND FUTURE WORK

In this article, we design an architecture for scene text recognition that both the encoder and the decoder are convolutional layers. Firstly, two-dimensional convolution is used in the encoder to abstract the word images. Then, the decoder equally employs an attention module to capture visual cues, and a language module to model linguistic rules, which can be regarded as an ensemble to make predictions jointly. The attention module is based on scaled dot-product, and the language module is built on one-dimensional convolution followed by GLU. Finally, multiple losses from attention and language are accumulated for training the networks in an end-to-end way. Extensive experiments demonstrate that our method can achieve highly-competitive or state-of-the-art results on common scene text recognition datasets. In the future, we will extend this work to text detection, and build a model for end-to-end scene text recognition with simpler structure.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (2017YFC0820600), the National Nature Science Foundation of China (61525206, 61771468), the Youth Innovation Promotion Association Chinese Academy of Sciences (2017209).

REFERENCES

[1] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. 2014. Word Spotting and Recognition with Embedded Attributes. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 12 (2014), 2552–2566.

[2] Ouais Alsharif and Joelle Pineau. 2014. End-to-End Text Recognition with Hybrid HMM Maxout Models. (2014).

[3] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv preprint arXiv:1607.06450* (2016).

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. (2015).

[5] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks* 5, 2 (1994), 157–166.

[6] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. 2017. Focusing Attention: Towards Accurate Text Recognition in Natural Images. In *ICCV*. IEEE Computer Society, 5086–5094.

[7] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *SSST@EMNLP*. Association for Computational Linguistics, 103–111.

[8] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *ICML (Proceedings of Machine Learning Research)*, Vol. 70. PMLR, 933–941.

[9] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*. IEEE Computer Society, 2625–2634.

[10] Shancheng Fang, Hongtao Xie, Zhineng Chen, Yizhi Liu, and Yan Li. 2017. Uyghur text matching in graphic images for biomedical semantic analysis. *Neuroinformatics* (2017), 1–11.

[11] Shancheng Fang, Hongtao Xie, Zhineng Chen, Shiai Zhu, Xiaoyan Gu, and Xingyu Gao. 2017. Detecting Uyghur text in complex background images with convolutional neural network. *Multimedia Tools Appl.* 76, 13 (2017), 15083–15103.

[12] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *ICML (Proceedings of Machine Learning Research)*, Vol. 70. PMLR, 1243–1252.

[13] Suman K. Ghosh, Ernest Valveny, and Andrew D. Bagdanov. 2017. Visual Attention Models for Scene Text Recognition. In *ICDAR*. IEEE, 943–948.

[14] Albert Gordo. 2015. Supervised mid-level features for word image representation. In *CVPR*. IEEE Computer Society, 2956–2964.

[15] Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850* (2013).

[16] Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML (ACM International Conference Proceeding Series)*, Vol. 148. ACM, 369–376.

[17] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. 2016. Synthetic Data for Text Localisation in Natural Images. In *CVPR*. IEEE Computer Society, 2315–2324.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *ICCV*. IEEE Computer Society, 1026–1034.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. IEEE Computer Society, 770–778.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *ECCV (4) (Lecture Notes in Computer Science)*, Vol. 9908. Springer, 630–645.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[22] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML (JMLR Workshop and Conference Proceedings)*, Vol. 37. JMLR.org, 448–456.

[23] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. *arXiv preprint arXiv:1406.2227* (2014).

[24] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep Structured Output Learning for Unconstrained Text Recognition. In *International Conference on Learning Representations (ICLR)*.

[25] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2016. Reading Text in the Wild with Convolutional Neural Networks. *International Journal of Computer Vision* 116, 1 (2016), 1–20.

[26] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep Features for Text Spotting. In *ECCV (4) (Lecture Notes in Computer Science)*, Vol. 8692. Springer, 512–528.

[27] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman K. Ghosh, Andrew D. Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, Faisal Shafait, Seiichi Uchida, and Ernest Valveny. 2015. ICDAR 2015 competition on Robust Reading. In *ICDAR*. IEEE Computer Society, 1156–1160.

[28] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernández Mota, Jon Almazán, and Lluís-Pere de las Heras. 2013. ICDAR 2013 Robust Reading Competition. In *2013 12th International Conference on Document Analysis and Recognition, Washington, DC, USA, August 25-28, 2013*. IEEE Computer Society, 1484–1493.

[29] Chen-Yu Lee and Simon Osindero. 2016. Recursive Recurrent Nets with Attention Modeling for OCR in the Wild. In *CVPR*. IEEE Computer Society, 2231–2239.

[30] Simon M. Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, and Robert Young. 2003. ICDAR 2003 Robust Reading Competitions. In *7th International Conference on Document Analysis and Recognition (ICDAR 2003), 2-Volume Set, 3-6 August 2003, Edinburgh, Scotland, UK*. IEEE Computer Society, 682–687.

[31] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*. The Association for Computational Linguistics, 1412–1421.

[32] Anand Mishra, Karteek Alahari, and C. V. Jawahar. 2012. Scene Text Recognition using Higher Order Language Priors. In *BMVC*. BMVA Press, 1–11.

[33] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*. Omnipress, 807–814.

[34] Lukas Neumann and Jiri Matas. 2012. Real-time scene text localization and recognition. In *CVPR*. IEEE Computer Society, 3538–3545.

[35] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. 2010. Large-scale image retrieval with compressed Fisher vectors. In *CVPR*. IEEE Computer Society, 3384–3391.

[36] José A. Rodríguez-Serrano and Florent Perronnin. 2013. Label embedding for text recognition. In *BMVC*. BMVA Press.

[37] Tim Salimans and Diederik P. Kingma. 2016. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *NIPS*. 901.

[38] Baoguang Shi, Xiang Bai, and Cong Yao. 2015. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *arXiv preprint arXiv:1507.05717* (2015).

[39] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. 2016. Robust Scene Text Recognition with Automatic Rectification. In *CVPR*. IEEE Computer Society, 4168–4176.

[40] Bolan Su and Shijian Lu. 2014. Accurate Scene Text Recognition Based on Recurrent Neural Network. In *ACCV (1) (Lecture Notes in Computer Science)*, Vol. 9003. Springer, 35–48.

[41] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *ICML (3) (JMLR Workshop and Conference Proceedings)*, Vol. 28. JMLR.org, 1139–1147.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 6000–6010.

[43] Kai Wang, Boris Babenko, and Serge J. Belongie. 2011. End-to-end scene text recognition. In *ICCV*. IEEE Computer Society, 1457–1464.

[44] Zbigniew Wojna, Alexander N. Gorban, Dar-Shyang Lee, Kevin Murphy, Qian Yu, Yeqing Li, and Julian Ibarz. 2017. Attention-Based Extraction of Structured Information from Street View Imagery. In *ICDAR*. IEEE, 844–850.

[45] Hongtao Xie, Ke Gao, Yongdong Zhang, Sheng Tang, Jintao Li, and Yizhi Liu. 2011. Efficient Feature Detection and Effective Post-Verification for Large Scale Near-Duplicate Image Search. *IEEE Trans. Multimedia* 13, 6 (2011), 1319–1332.

[46] Hongtao Xie, Yongdong Zhang, Jianlong Tan, Li Guo, and Jintao Li. 2014. Contextual Query Expansion for Image Retrieval. *IEEE Trans. Multimedia* 16, 4 (2014), 1104–1114.

[47] Cong Yao, Xiang Bai, Baoguang Shi, and Wenyu Liu. 2014. Strokelets: A Learned Multi-scale Representation for Scene Text Recognition. In *CVPR*. IEEE Computer Society, 4042–4049.