Edit Probability for Scene Text Recognition

Fan Bai[1*] Zhanzhan Cheng[2] Yi Niu[2]
Shiliang Pu[2] Shuigeng Zhou[1†]
[1]Shanghai Key Lab of Intelligent Information Processing, and School of Computer Science, Fudan University, Shanghai 200433, China
[2]Hikvision Research Institute, China
{fbai17,sgzhou}@fudan.edu.cn
{chengzhanzhan,niuyi,pushiliang}@hikvision.com

Abstract

We consider the scene text recognition problem under the attention-based encoder-decoder framework, which is the state of the art. The existing methods usually employ a frame-wise maximal likelihood loss to optimize the models. When we train the model, the misalignment between the ground truth strings and the attention's output sequences of probability distribution, which is caused by missing or superfluous characters, will confuse and mislead the training process, and consequently make the training costly and degrade the recognition accuracy. To handle this problem, we propose a novel method called edit probability (EP) for scene text recognition. EP tries to effectively estimate the probability of generating a string from the output sequence of probability distribution conditioned on the input image, while considering the possible occurrences of missing/superfluous characters. The advantage lies in that the training process can focus on the missing, superfluous and unrecognized characters, and thus the impact of the misalignment problem can be alleviated or even overcome. We conduct extensive experiments on standard benchmarks, including the IIIT-5K, Street View Text and ICDAR datasets. Experimental results show that the EP can substantially boost scene text recognition performance.

1. Introduction

Text recognition in natural scene images has recently attracted much research interest of the computer vision community [18, 25, 31]. The sequence-learning-based text recognition techniques, which have been advancing rapidly in recent years [7, 25, 32], generally consist of an encoding module and a decoding module. The encoding module usually encodes the input images to vectors of fixed dimensionality with a certain encoding technique, such as convolution neural network (CNN) [32], or recurrent neural network (RNN) including long short-term memory (LSTM) [16, 31, 34] and gate recurrent neural network (GRU) [4, 8, 32]. While the decoding module decodes the encoded feature vectors into the target strings by exploiting RNN, connectionist temporal classification (CTC) [12, 31] or attention mechanism [3, 5, 32] *etc*.

The state-of-the-art of scene text recognition is the attention-based encoder-decoder framework [7, 25, 32]. It outputs a sequence of *probability distribution* (*pd*) that is expected to be aligned with the characters of the text in the input image. In model training, the probability of the *ground-truth text* (*gt* in short), calculated by the corresponding output *pd* sequence in a frame-wise style, is utilized to estimate the likelihood of model parameters. In this paper, we call this joint probability *frame-wise probability* (FP), and the existing frame-wise loss based methods *FP based methods* in the sequel.

However, in both the training and predicting processes of the attention-based text recognition models, some characters may be missing or superfluous, which results in misalignment between the *gt* and the output sequence of *pd*. In a recent work, Cheng *et al*. [7] considered this phenomenon as a result of attention drift, and solved it by introducing the Focusing Attention Net (FAN). This method achieved the state of the art performance. But the training of FAN requires extra pixel-wise supervising information, which is expensive to provide. And the training process is time-consuming due to the large amount of pixel-wise calculation.

Fig. 1 provides examples to illustrate the phenomenon of missing and superfluous characters in training an attention-based text recognition model on the ground truth "DOVE#". Here, '#' represents the End-Of-Sequence (EOS) symbol,

*Fan Bai did most of this work when he was an intern in Hikvision Research Institute.

†Corresponding author.

which is commonly used in attention-based methods [7, 25, 32]. In Fig. 1 (a) and (b), the model may recognize the inputs as "DVE#" and "DOOVE#" respectively, based on the output pd sequences. Comparing against the gt "DOVE#", it is natural to say that the former misses an 'O' and the latter has a superfluous 'O'.

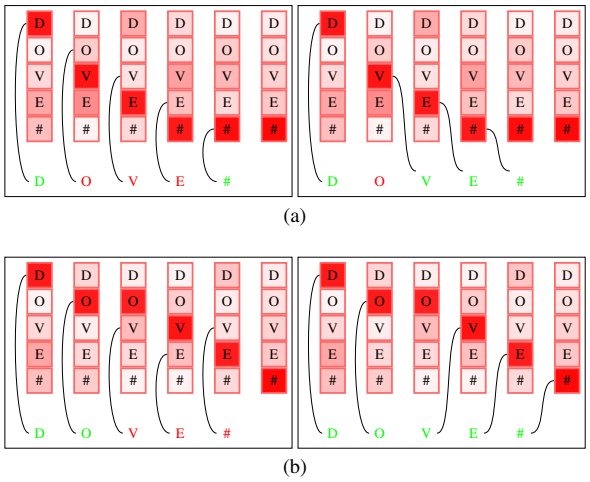

Figure 1. The illustration of misalignment between the output sequences of *pd* and the ground truth "DOVE#" due to (a) missing character and (b) superfluous character. A cell with higher saturation corresponds to a character with higher probability in the *pd*. The green *gt*-characters at the bottom of each subfigure have the highest probability in their *pd*s (linked by curves), while the red characters have lower probability in their *pd*s. In subfigure (a), the left part shows that 'O', 'V' and 'E' have low probability in their corresponding *pd*s. The right part shows if an 'O' is inserted, then 'V' and 'E' are alignable with the probabilities in their *pd*s. Similarly, in subfigure (b), if an 'O' is removed, then the remaining 'O', 'V', 'E' and '#' have matched probabilities in their *pd*s.

By checking the training process of attention-based text recognition models, we can see that the FP-based methods simply train the model by maximizing the probability of each character in the *gt*, based on the corresponding *pd* of the attention's output sequence. However, the misalignments caused by missing or superfluous characters may confuse and mislead the training process, and consequently make the training costly and degrade the recognition accuracy. Concretely, back to Fig. 1, except for 'D' and '#', all the other three characters 'O', 'V' and 'E' have low probability in the corresponding *pd*s (see the left diagrams), thus large error will be back-propagated to these *pd*s for further training. Instead, if the training algorithm realizes that the character 'O' in Fig. 1 (a) is missing and one of the character 'O' in Fig. 1 (b) is superfluous, it will align 'V' and 'E' in Fig. 1 (a), 'V', 'E', and '#' in Fig. 1 (b) to more appropriate *pd*s (see the right diagrams), and the characters will have higher probability under the new alignment, then it needs only to focus on the missing/superfluous character 'O', which will make the training simpler and less costly.

Motivated by the observation above and inspired by the concept of sequence alignment where edit distance is used to measure the dissimilarity of two sequences, in this paper we propose a new method for scene text recognition under the attention-based encoder-decoder framework, which is called **edit probability** (**EP** in short). By treating the misalignment between the gt and the output pd sequence as the result of possible occurrences of missing/superfluous characters, in the training process, EP tries to effectively estimate the probability of a string conditioned on the output sequence of *pd* under certain model parameters while considering possible occurrences of missing/superfluous characters. The merit lies in that the training process can focus on the missing, superfluous and misclassified characters, and the impact of misalignment can be reduced substantially. To validate the proposed method, we conduct extensive experiments on several benchmarks, which show that EP can significantly boost recognition performance.

The rest of this paper is organized as follows: Section 2 reviews related work, Section 3 presents the EP method in detail, Section 4 conducts empirical evaluation, and Section 5 concludes the paper.

2. Related Work

In the past decade, many methods have been proposed for scene text recognition, which roughly fall into three types: 1) traditional methods with handcrafted features, 2) Naïve deep neural-network-based methods, and 3) sequence-based methods.

In early years, traditional methods first extract handcrafted visual features for individual character detection and recognition one by one, then integrate these characters into words based on a set of heuristic rules or a language model. Neumann and Matas [28] recognized characters by training a Support Vector Machine (SVM) classifier with the defined handcrafted features such as aspect ratio, hole area ratio etc. Wang *et al*. [35, 36] first trained a character classifier with the extracted HOG descriptors [38], then recognized characters of the cropped word image by a sliding window one by one. However, due to the low representation capability of handcrafted features, these methods cannot achieve satisfactory recognition performance.

Later, instead of handcrafted features, some deep neural-network-based methods were developed for extracting robust visual features. Bissacco *et al*. [6] adopted a fully connected network (FC) of 5 hidden layers for extracting character features, then applied an n-gram language model to recognize characters. Wang *et al*. [37] and Jaderberg *et al*. [19, 20] first developed CNN-based framework for character feature representation, then applied some heuris-

tic rules for characters generation. These naïve deep neural-network-based methods above usually recognized character sequence based on some pre/post-processing, such as the segmentation of each character or a non-maximum suppression, which may be very challenging because of the complicated background and the inadequate distance between consecutive characters.

Recently, some researchers treated the text recognition task as a sequence learning problem: first encoding a text image into a sequence of features with deep neural network, then directly generating character sequence with sequence recognition techniques. He *et al*. [15] and Shi *et al*. [31] proposed the end-to-end neural networks that first capture visual feature representation by using CNN or RNN, then the CTC [12] loss was combined with the neural network outputs for calculating the conditional probability between the predicted and the target sequences. The state of the art for text recognition is the attention-based methods [7, 25, 32]. These methods first combined CNN and RNN for encoding text images into feature representations, then employed a frame-wise loss to optimize the model. In the training process, the misalignment between the gt sequence and the output pd sequence may mislead the training algorithm and result in poor performance.

Note that the misalignment problem has also been observed in attention training of speech recognition. Kim *et al*. tried to solve the problem by using a joint CTC-attention model within the multi-task learning framework [24]. However, as pointed in [7], the joint CTC-attention model does not work well in scene text recognition. This paper also addresses the scene text recognition problem under the attention-based framework. Different from the existing methods, we propose a novel method EP that tries to estimate the probability of a string conditioned on the input image, by treating the misalignment between the gt text and the output pd sequence as the result of possible occurrences of missing/superfluous characters. EP provides an effective way to handle the misalignment problem, and empirically it outperforms the existing methods.

3. The EP Method

In this section, we present the EP method in detail, including the EP-based attention decoder, the formulation of EP, the EP training process, and EP based prediction with/without a lexicon.

Edit probability is proposed to effectively train attention-based models for accurate scene text recognition. Conceptually, for an image $\mathcal{I}$ and a text string T, $EP(T|\mathcal{I};\theta)$ measures the probability of T conditioned on $\mathcal{I}$ under model parameters θ. It is evaluated by summing the probabilities of all possible edit paths that transform an initially empty string to T based on the *pd* sequence y generated by the model. And each edit path consists of a sequence of edit operations that are detailed in Sec. 3.2.

3.1. EP-based Attention Decoder

The original attention decoder is an RNN that generates the output *pd* y_j on the j-*th* step [4]:

$$\begin{aligned} y_j &= \mathrm{softmax}(v^\top s_j), \\ s_j &= \mathrm{LSTM}(y_{j-1}, c_j, s_{j-1}), \\ c_j &= \textstyle\sum_{k'=1}^{|h|} \alpha_{j,k} h_k, \\ \alpha_{j,k} &= \dfrac{\exp(e_{j,k})}{\sum_{k'=1}^{|h|} \exp(e_{j,k})}, \\ e_{j,k} &= w^\top \tanh(W s_{j-1} + V h_k + b) \end{aligned} \tag{1}$$

where h is a sequence of encoded feature vectors, and s_j, c_j, α_j and e_j represent the LSTM [16] hidden state, the weighted sum of h, the attention weights and the alignment model on the j-*th* step, respectively. w, W, V, b and v are all trainable parameters.

In this work, for EP calculation, the attention decoder also generates R_j and I_j on the j-*th* step:

$$\begin{aligned} R_j = (R_j^C, R_j^I, R_j^D)^\top &= \mathrm{softmax}(W_R^\top s_j), \\ I_j &= \mathrm{softmax}(W_I^\top s_j) \end{aligned} \tag{2}$$

where W_R and W_I are trainable parameters. R_j^C, R_j^I and R_j^D respectively represents the probability of y_j *being correctly aligned*, *a character being missing before* y_j and y_j *being superfluous*. And I_j is the *pd* of characters being missing before y_j conditioned on R_j^I.

3.2. Edit Probability

Formally, with the alphabet (including the EOS) Σ, let $T \in \mathbb{T}_\Sigma$ where $\mathbb{T}_\Sigma$ represents the set of all valid strings (each of which contains one and only one EOS as its end token) on Σ. We define the edit states as tuples $(T_{1:i}, y_{1:j})$ for $0 \le i \le |T|$ and $0 \le j \le |y|$. And the state $(T_{1:i}, y_{1:j})$ indicates generating $T_{1:i}$ from $y_{1:j}$. In particular, $T_{1:0}$ and $y_{1:0}$ represent an empty string (also denoted by "") and an empty *pd* sequence respectively. The edit operations for state $(T_{1:i}, y_{1:j})$ are defined as follows:

- *consumption*: the consumption operation $\varepsilon_{i,j}^C$ transforms state $(T_{1:i-1}, y_{1:j-1})$ to state $(T_{1:i}, y_{1:j})$ if $0 < i \le |T|$ and $0 < j \le |y|$ by regarding the character T_i and the *pd* A_j as being correctly aligned, and appending T_i to $T_{1:i-1}$ by consuming y_j. Formally,

$$\varepsilon_{i,j}^C(T_{1:i-1}, y_{1:j-1}) = (T_{1:i}, y_{1:j}). \tag{3}$$

  The probability of this operation is the joint probability of *doing consumption* and *consuming* T_i *of* y_j. That is,

$$p(\varepsilon_{i,j}^C|\mathcal{I};\theta) = R_j^C y_j(T_i). \tag{4}$$

- *deletion*: the deletion operation $\varepsilon^D_{i,j}$ transforms state $(T_{1:i}, y_{1:j-1})$ to state $(T_{1:i}, y_{1:j})$ if $0 < j \leq |y|$ by regarding the *pd* y_j as being superfluous, and deleting y_j directly. Formally,

$$\varepsilon^D_{i,j}(T_{1:i}, y_{1:j-1}) = (T_{1:i}, y_{1:j}), \tag{5}$$

For $T_i \neq$ '#', the probability of this operation is R^D_j. And for $T_i =$ '#', *deletion* is the only allowed operation on state $(T_{1:i}, y_{1:j-1})$ as any character after the EOS is ignored. So we have

$$p(\varepsilon^D_{i,j}|\mathcal{I};\theta) = \begin{cases} R^D_j & \text{if } T_i \neq \text{'\#'}, \\ 1 & \text{if } T_i = \text{'\#'}. \end{cases} \tag{6}$$

- *insertion*: the insertion operation $\varepsilon^I_{i,j}$ transforms state $(T_{1:i-1}, y_{1:j})$ to state $(T_{1:i}, y_{1:j})$ if $0 < i \leq |T|$ by regarding T_i as a missing character, and appending T_i to $T_{1:i-1}$ directly. Formally,

$$\varepsilon^I_{i,j}(T_{1:i-1}, y_{1:j}) = (T_{1:i}, y_{1:j}). \tag{7}$$

For $j < |y|$, the probability of this operation is the joint probability of *a character is missed from the position just before* y_{j+1} and *the missing character is* T_{i+1}. And for $j = |y|$, *insertion* is the only allowed operation over state $(T_{1:i-1}, y_{1:|y|})$ as there is no more *pd* to delete or consume. So we have

$$p(\varepsilon^I_{i,j}|\mathcal{I};\theta) = \begin{cases} R^I_j I_j(T_i) & \text{if } j < |y|, \\ I_j(T_i) & \text{if } j = |y|. \end{cases} \tag{8}$$

By assuming that the edit operations over T and y are conditional independent, the probability of an edit path E is the joint probability of all the edit operations in E. That is,

$$p(E|\mathcal{I};\theta) = \prod_{t=1}^{|E|} p(E_t|\mathcal{I};\theta), \tag{9}$$

where E_t refers to the *t-th* edit operation in E. In particular, the probability of an empty path is 1.

The edit probability of states $(T_{1:i}, y_{1:j})$ is evaluated by the sum of all the conditional probabilities of edit paths $E_{1:|E|} \in \mathbb{E}^*_\Sigma$ that transform ("", $y_{1:0}$) to $(T_{1:i}, y_{1:j})$ where "" and $y_{1:0}$ represent an empty string and an empty *pd* sequence respectively. Formally,

$$\text{ep}(T_{1:i}, y_{1:j}) = \sum_{\substack{E \in \mathbb{E}^*_\Sigma \\ E(\text{"}\text{"}, y_{1:0}) = (T_{1:i}, y_{1:j})}} p(E|\mathcal{I};\theta) \tag{10}$$

where and $\mathbb{E}_\Sigma$ is the set of all edit operations over T and y. That is,

$$\mathbb{E}_\Sigma \stackrel{\text{def}}{=} \{\varepsilon^C_{i,j}, \varepsilon^D_{i,j}, \varepsilon^I_{i,j} | 0 < i \leq |T|, 0 < j \leq |y|\}. \tag{11}$$

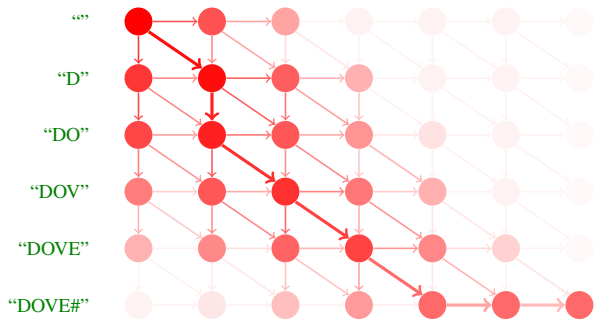

Figure 2. The illustration of edit probability calculation. The red filled circle on the i-th row j-th column (count from 0) indicates the state $(T_{1:i}, y_{1:j})$, in which $T_{1:i}$ is the string prefix (as shown with the green words) and $y_{1:j}$ is the prefix of the *pd* sequence (as shown in Fig. 1 (a)). Given the state $(T_{1:i}, y_{1:j})$: 1) the horizontal arrow below the state indicates the deletion operation $\varepsilon^D_{i,j}$; 2) The vertical arrow right to the state indicates the insertion operation $\varepsilon^I_{i,j}$; 3) The diagonal arrow right below the state indicates the consumption operation $\varepsilon^C_{i,j}$. And the higher saturation of a red filled circle or an arrow refers to the higher probability of the state or the corresponding edit operation, vice versa. The edit path with the highest probability is emphasized with bold arrows, which consumes y_1, y_2, y_3 and y_4 to generate 'D', 'V', 'E' and '#' respectively, inserts an 'O' before y_2, and deletes y_5 and y_6.

However, enumerating all the possible edit paths is prohibitively expensive (if not impossible) as the search space is too large. Fortunately, this can be solved by dynamic programming based on the following equation inferred from Eq. 9:

$$p(\varepsilon \circ E|\mathcal{I};\theta) = p(\varepsilon|\mathcal{I};\theta)p(E|\mathcal{I};\theta), \tag{12}$$

for $\varepsilon \in \mathbb{E}_\Sigma$ and $E \in \mathbb{E}^*_\Sigma$. And $\circ$ represents the concatenation operator for edit operations and edit paths, which is defined as follows:

$$(\varepsilon \circ E)(T_{1:i}, y_{1:j}) = E(\varepsilon(T_{1:i}, y_{1:j})). \tag{13}$$

With Eq. 12, we can rewrite Eq. 10 as follows:

$$\begin{aligned} &\text{ep}(T_{1:i}, y_{1:j}) \\ &= \begin{cases} 1 & \text{if } i = 0, j = 0 \\ p(\varepsilon^D_{i,j}|\mathcal{I};\theta)\,\text{ep}(T_{1:i}, y_{1:j-1})\delta_{j>0} & \text{otherwise} \\ \quad + p(\varepsilon^I_{i,j}|\mathcal{I};\theta)\,\text{ep}(T_{1:i-1}, y_{1:j})\delta_{i>0} & \\ \quad + p(\varepsilon^C_{i,j}|\mathcal{I};\theta)\,\text{ep}(T_{1:i-1}, y_{1:j-1})\delta_{i>0,j>0} & \end{cases} \end{aligned} \tag{14}$$

where the value of $\delta_{condition}$ is 1 if the condition is met, otherwise 0. By recursively applying Eq. 14, $\text{EP}(T|\mathcal{I};\theta) = \text{ep}(T, y)$ can be calculated in $\text{O}(|T| \cdot |y|)$.

Fig. 2 shows the EP calculation process for generating "DOVE#" from the sequence of *pd* displayed in Fig. 1 (a). We can see that the insertion operation, which inserts an 'O' is contained in the maximal probable edit path. This is an expected result.

3.3. EP Training

With the training set $\mathcal{X}$ that consists of pairs of image and *gt* string, the EP training is to find $\hat{\theta}$ that minimizes the negative log-likelihood over $\mathcal{X}$:

$$\hat{\theta} = \arg\min_{\theta} - \sum_{(\mathcal{I},T)\in\mathcal{X}} \ln(\mathrm{EP}(T|\mathcal{I};\theta)) \tag{15}$$

The model can be optimized with standard back-propagation algorithm [30].

3.4. EP Predicting

EP Predicting is to find the string $\hat{T}$ that maximizes $\mathrm{EP}(\hat{T}|\mathcal{I};\hat{\theta})$:

$$\hat{T} = \arg\max_{T\in\mathbb{T}_\Sigma} \mathrm{EP}(T|\mathcal{I};\hat{\theta}). \tag{16}$$

However, looking for the whole answer set $\mathbb{T}_\Sigma$ with Eq. 16 is extremely costly. Therefore, we develop two efficient sequence generation mechanisms for both lexicon-free and lexicon-based prediction.

Predicting without lexicon. By deeply analyzing the prediction problem, we find that in general, the string $\hat{T}$ that maximizes $\mathrm{EP}(\hat{T}|\mathcal{I};\hat{\theta})$ is mostly a prefix (ended by an EOS '#') of the string $\mathcal{T}$ with the most probable edit path that transforms $(\text{“”}, y_{1:0})$ to $(\mathcal{T}, y)$ where '#' $\notin \mathcal{T}$.

Therefore, we firstly find the string $\mathcal{T}$:

$$\mathcal{T} = \arg\max_{T\in(\Sigma\setminus\{\text{'\#'}\})^*} \max_{\substack{E\in\mathbb{E}^*_\Sigma \\ E(\text{“”},y_{1:0})=(T,y)}} p(E|\mathcal{I};\theta) \tag{17}$$

where $\Sigma\setminus\{\text{'\#'}\}$ represents the alphabet without the EOS, then use all the prefixes of $\mathcal{T}$ (each ended by an EOS) as the candidate set $\mathcal{B}$:

$$\mathcal{B} = \{\mathcal{T}_{1:i} \oplus \text{'\#'} | 0 \le i \le |\mathcal{T}|\} \tag{18}$$

where $\oplus$ represents the concatenation operator for two strings, finally select the best $\hat{T} \in \mathcal{B}$ that maximizes $\mathrm{ep}(\hat{T}, A)$:

$$\hat{T} = \arg\max_{T\in\mathcal{B}} \mathrm{EP}(T|\mathcal{I};\hat{\theta}). \tag{19}$$

Note that the edit path with the highest probability should not include an insertion edit operation that inserts a non-EOS character, because we can remove such insertions and get a new edit path whose conditional probability is greater than the previous one.

Therefore, we can generate $\mathcal{T}$ by beginning with the state $(\text{“”}, y_{1:0})$ and performing the most probable deletion or consumption operation (not considering any operation generating the EOS) at each step, till a state $(\mathcal{T}, y)$ is reached. Since all strings in $\mathcal{B}$ share the common prefixes, we can compute all the $\mathrm{ep}(T, y)$ for $T \in \mathcal{B}$ in $O(|\mathcal{T}| \cdot |y|)$ time with Eq. 14.

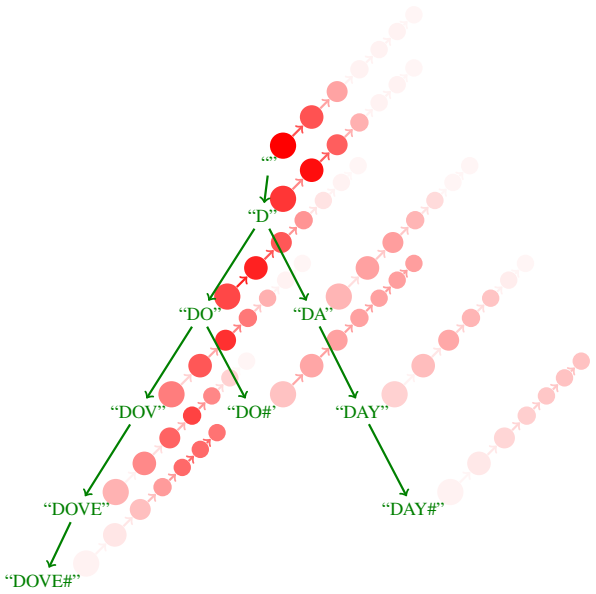

Figure 3. The illustration of edit probability trie. The trie contains a lexicon with three strings: “DOVE#”, “DO#” and “DAY#”. Each node represents a prefix of one or more strings in the lexicon. Upper right to each node with the prefix (say S) is a probability vector where the color saturation of the j-*th* element represents $\mathrm{ep}(S, y_{1:j})$ and y is the output sequence of *pd*. Higher color saturation means higher probability, and vice versa.

Predicting with a lexicon. In constrained cases, we can enumerate the strings in a lexicon $\mathcal{D}$, and find the most probable one. A tunable parameter λ is used to indicate how much we trust the lexicon $\mathcal{D}$ since some target strings may be not contained in the lexicon. Therefore, we actually assess all the possible strings in $\mathcal{B} \cup \mathcal{D}$ by

$$\hat{T} = \arg\max_{T\in\mathcal{B}\cup\mathcal{D}} \begin{cases} \lambda\,\mathrm{EP}(T|\mathcal{I};\hat{\theta}) & \text{if } T \in \mathcal{D} \\ (1-\lambda)\,\mathrm{EP}(T|\mathcal{I};\hat{\theta}) & \text{if } T \notin \mathcal{D} \end{cases} \tag{20}$$

where $0.5 \le \lambda \le 1$. The larger λ is, the more we trust the lexicon, and vice versa. Specifically, $\lambda = 0.5$ means that the lexicon can provide only some additional candidates that are treated equally to those in $\mathcal{B}$, while $\lambda = 1$ means that the generated strings are guaranteed to appear in the lexicon $\mathcal{D}$.

However, with the growing of lexicon size, the above enumeration-based method is extremely time-consuming. To tackle this problem, Shi *et al*. used a prefix tree (Trie) [9, 32] to accelerate the search process since many strings in the lexicon share common prefixes. In this work we develop a data structure called **edit probability Trie (EP-Trie)**, which is a variant of Trie with nodes containing not only a prefix S, but also a vector indicating $\mathrm{ep}(S, y_{1:j})$ for $0 \le j \le |y|$. The vector of a node can be computed from the vector of its parent in O($|y|$) time with Eq. 14. We will demonstrate the effectiveness of EP-Trie in Sec. 4.5. Fig. 3 illustrates EP-Trie.

Method	IIIT5k			SVT		IC03			IC13	IC15
	50	1k	None	50	None	50	Full	None	None	None
ABBYY [35]	24.3	–	–	35.0	–	56.0	55.0	–	–	–
Wang *et al*. [35]	–	–	–	57.0	–	76.0	62.0	–	–	–
Mishra *et al*. [13]	64.1	57.5	–	73.2	–	81.8	67.8	–	–	–
Wang *et al*. [37]	–	–	–	70.0	–	90.0	84.0	–	–	–
Goel *et al*. [10]	–	–	–	77.3	–	89.7	–	–	–	–
Bissacco *et al*. [6]	–	–	–	90.4	78.0	–	–	–	87.6	–
Alsharif and Pineau [2]	–	–	–	74.3	–	93.1	88.6	–	–	–
Almazán *et al*. [1]	91.2	82.1	–	89.2	–	–	–	–	–	–
Yao *et al*. [38]	80.2	69.3	–	75.9	–	88.5	80.3	–	–	–
Rodríguez-Serrano *et al*. [29]	76.1	57.4	–	70.0	–	–	–	–	–	–
Jaderberg *et al*. [19]	–	–	–	86.1	–	96.2	91.5	–	–	–
Su and Lu [33]	–	–	–	83.0	–	92.0	82.0	–	–	–
Gordo [11]	93.3	86.6	–	91.8	–	–	–	–	–	–
Jaderberg *et al*. [20]	97.1	92.7	–	95.4	80.7	98.7	**98.6**	93.1	90.8	–
Jaderberg *et al*. [19]	95.5	89.6	–	93.2	71.7	97.8	97.0	89.6	81.8	–
Shi *et al*. [31]	97.6	94.4	78.2	96.4	80.8	98.7	97.6	89.4	86.7	–
Shi *et al*. [32]	96.2	93.8	81.9	95.5	81.9	98.3	96.2	90.1	88.6	–
Lee *et al*. [25]	96.8	94.4	78.4	96.3	80.7	97.9	97.0	88.7	90.0	–
Cheng *et al*. [7]	99.3	97.5	87.4	**97.1**	85.9	**99.2**	97.3	94.2	93.3	70.6
Shi *et al*. (baseline) [32]	96.5	92.8	79.7	96.1	81.5	97.8	96.4	88.7	87.5	–
Cheng *et al*. (baseline) [7]	98.9	96.8	83.7	95.7	82.2	98.5	96.7	91.5	89.4	63.3
Shi's + EP (ours)	99.1	97.3	85.0	96.3	86.2	98.4	97.0	93.7	93.0	68.1
Cheng's + EP (ours)	**99.5**	**97.9**	**88.3**	96.6	**87.5**	98.7	97.9	**94.6**	**94.4**	**73.9**

Table 1. Results of recognition accuracy on general benchmarks. "50" and "1k" are lexicon sizes, "Full" indicates the combined lexicon of all images in the benchmarks, and "None" means lexicon-free. The results of the baseline methods are directly referenced from the "SRN only" and the "Baseline" in [32] and [7] respectively.

4. Performance Evaluation

We conduct extensive experiments to validate the EP method on several general recognition benchmarks under the attention framework. For a fair and comprehensive comparison, we directly employ the structures of the state-of-the-art works and replace their loss layers with EP. We first evaluate the performance of the EP-based methods, and compare them with the existing methods. Then we demonstrate the advantage of EP training over frame-wise loss based training on some real training data. Finally, we evaluate our method with the Hunspell 50k lexicon [17], and compare EP predicting methods with and without lexicon.

4.1. Datasets

IIIT 5K-Words [27] (IIIT5K) is a dataset collected from the Internet with 3000 cropped word images in its test set. For each of its images, a 50-word lexicon and a 1k-word lexicon are specified, both of which contain the ground truth words as well as other randomly picked words.

Street View Text [35] (SVT) is collected from the Google Street View. Its test set consists of 647 word images, each of which is specified with a 50-word lexicon.

ICDAR 2003 [26] (IC03) contains 251 scene images with text bounding boxes. Each image is associated with a 50-word lexicon defined by Wang *et al*. [35]. A full lexicon that combines all lexicon words is also provided. For fair comparison, as in [35], we discard the images containing non-alphanumeric characters or have less than three characters. The resulting dataset contains 867 cropped images.

ICDAR 2013 [23] (IC13) is the successor of IC03, so most of its data are inherited from IC03. It contains 1015 cropped text images, but no lexicon is associated.

ICDAR 2015 [22] (IC15) contains 2077 cropped images. For fair comparison, we discard the images containing non-alphanumeric characters, and eventually obtain 1811 images in total. No lexicon is associated.

4.2. Implementation Details

Network Structure: The attention-based encoder-decoder framework is the-state-of-the-art technique for text recognition, which consists of two major steps: 1) Obtaining visual feature representation with a CNN-based feature extractor, such as 7-Conv-based by Shi *et al*. [32] and ResNet-based by Cheng *et al*. [7]; 2) Generating the output sequence of probability distribution with the attention

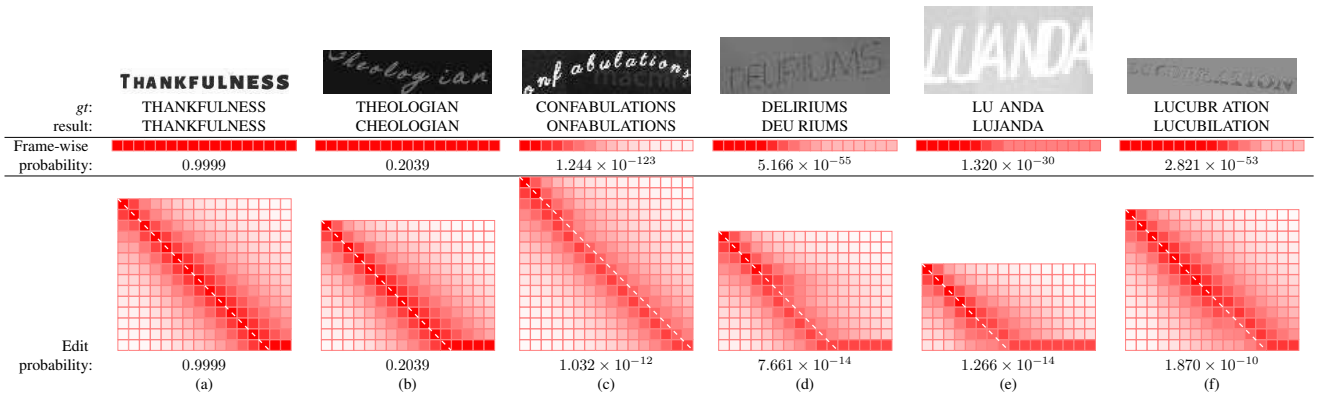

Figure 4. The visual comparison of EP and FP on real training data. In each subfigure, the characters shown in the 3rd raw are those having the largest probability in the corresponding *pd*. A vector of probability is shown in the 4th row, where the red saturation of the j-*th* element (labeled from 0) of the vector indicates the FP value of the first j characters in the *gt*, calculated by the first j *pd*s in the output sequence. A matrix of probability is given in the 6th row, where the red saturation of the (i, j) element of the matrix indicates the EP value of generating the first i characters in the *gt* from the first j *pd*s in the output sequence. In subfigure (a) and (b), no misalignment occurs. The maximal possible edit path appears on the diagonal line of the EP matrix because no insertion/deletion operation is likely to be performed. In subfigure (c) and (d), some misalignments occur because some characters are missing. The maximal possible edit path appears below the diagonal line of the EP matrix because some insertions are likely to be performed near by the places of the missing characters. In subfigure (e) and (f), some misalignments occur because some characters are superfluous. The maximal possible edit path appears to the right of the diagonal line of the EP matrix because some deletions are likely to be performed near by the places of the superfluous characters.

model. In this work, we evaluate the proposed method by replacing the FP-based training/predicting in Shi's and Cheng's structures with the EP-based training/predicting.

Model Training: Our model is trained on 8-million synthetic data released by Jaderberg *et al*. [18] and 4-million synthetic data (excluding the images that contain non-alphanumeric characters) cropped from 800-thousand images released by Gupta *et al*. [14] by the ADADELTA [39] optimization method.

Running Environment: Our method is implemented under the Caffe framework [21]. In our implementation, most modules in our model can be GPU accelerated as the CUDA backend is extensively used. All experiments are conducted on a workstation equipped with an Intel Xeon(R) E5-2650 2.30GHz CPU, an NVIDIA Tesla P40 GPU and 128GB RAM.

4.3. Comparison with Existing Methods

Tab. 1 shows the performance results of two EP-based methods, two FP-based (baseline) methods and previous methods. With Shi's and Cheng's structures, the EP-based methods significantly outperform the baseline methods on all benchmarks. In comparison with the existing methods, we consider both constrained and unconstrained cases. In the unconstrained cases, the EP-based method (Cheng's + EP) outperforms all existing methods; While in the constrained cases, our method (Cheng's + EP) performs better than all existing methods on IIIT5K, and achieves comparable results to that of the method proposed by Cheng *et al*. (FAN) [7] on SVT and IC03 datasets. However, it should be pointed out that FAN is trained with both word-level and character-level bounding box annotations, which is expensive and time consuming. On the contrary, our method is trained with only word-level ground truth. We also note that the method proposed by Jaderberg *et al*. [18] achieves the best result on IC03 with the full lexicon, but it cannot recognize out-of-training-set words.

4.4. Performance of EP Training

To demonstrate the advantage of EP in training stage, we compare the calculation of the FP and the proposed EP on some real training data. The input images, the ground truths and the recognition results are displayed in the 1st, 2nd and 3rd rows in the upper part of Fig. 4, respectively. As for the recognition results, they are actually the *pd* sequences. For demonstration convenience, we just display the characters that dominate the corresponding *pd*.

For the FP calculation, we show a vector of probability for each image on the 4th row and the FP value on the 5th row in Fig. 4. The value of the j-*th* element in the vector represents the joint conditional probability of generating the first j characters in *gt* from the first j *pd*s in the output *pd* sequence, and the probabilities after the EOS are ignored (regarded as 1). We have the following observations on FP: 1) when the output *pd* sequence is well aligned to the *gt* (see Fig. 4 (a) and (b)), the FP declines only at the place where the character is misclassified; 2) when the output *pd* sequence is misaligned to the *gt* (see Fig. 4 (c),

(d), (e) and (f)), the FP continues to decline after any occurrence of missing/superfluous character even some characters are correctly recognized. As a result, in the cases of misalignment, the error will be back-propagated to the correctly recognized *pd*s following the missing/superfluous character, which may confuse the model training.

For the EP calculation, we display a matrix of probability for each image on the 6th row and the EP values on the 7th row. The value of the (i, j) element represents $\mathrm{ep}(T_{1:i}, y_{1:j})$ where T is the *gt* and y is the output sequence of *pd* conditioned on the input image $\mathcal{I}$. We have the following observations on EP: 1) When the output sequence of *pd* is well aligned to the *gt* (see Fig. 4 (a) and (b)), no matter whether the classification result is correct, the EP value is almost equal to the FP value and the most probable edit path appears on the matrix diagonal line, that is, generating every character in *gt* by consuming the corresponding *pd*; 2) When some characters are missing/superfluous (see Fig. 4 (c) and (d) for missing character cases, (e) and (f) for superfluous character cases), the EP value is much larger than the FP value, and the most probable edit path appears under or to the right of the matrix diagonal line after the occurrence of the missing/superfluous characters. Different from the FP, the EP is indicative of inserting/deleting the missing/superfluous character and generating others by consuming the aligned *pd*. As a result, in the cases of misalignment, much error will be back-propagated only to the place where the missing/superfluous character occurs, and little error back-propagated to the correctly recognized *pd*s before or after the occurrence of the missing/superfluous characters, which makes the model training process focus on the missing/superfluous characters, instead of the misaligned characters.

As EP-based methods theoretically require more calculation than baselines, we measure the time cost for training. The result shows that Shi's/Cheng's baselines cost 170.5ms/536.0ms per iteration, and EP based methods cost only 6.8ms/7.0ms more (batch size is set to 75).

4.5. Performance of EP Prediction

Here, we evaluate the performance of EP prediction with/without lexicon. In real world text recognition tasks, it is not easy to get the ground-truth-related lexicons. Therefore, following previous works [2, 19, 20, 31, 32], we also test our methods on a public ground-truth-unrelated lexicon Hunspell [17], which contains more than 50k words. As mentioned in Sec. 3.4, λ is a tunable super-parameter in the predicting stage. We conduct lexicon-based prediction on all datasets by varying λ from 0.5 to 1. The results are shown in Fig. 5. We can see that 1) the accuracy increases when λ varies from 0.5 to 0.98, but decreases rapidly when λ approaches 1 due to over-correction; 2) When λ is set to 0.5, the accuracy of lexicon-based prediction is exactly

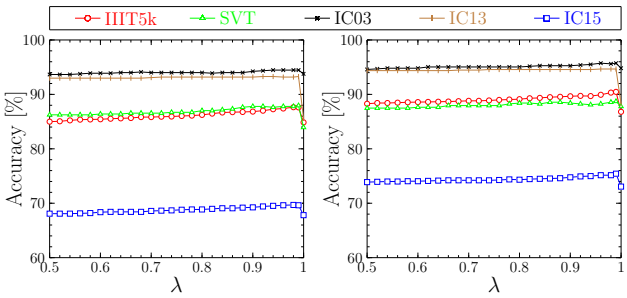

Figure 5. The accuracy on general recognition datasets when predict with the Hunspell 50k lexicon and λ varies from 0.5 to 1. The left and right figures are the results of the Shi's+EP method and Cheng's+EP method, respectively.

the same as that of lexicon-free prediction, which demonstrates the effectiveness of the proposed lexicon-free prediction method; 3) The ground-truth-unrelated lexicon is also helpful for improving text recognition performance if λ falls in a proper range (from 0.9 to 0.98), which validates the effectiveness of the lexicon-based prediction method.

Besides, we also test the enumeration-based and EP-Trie-based methods in terms of recognition accuracy and time cost per image while using the 50k lexicon. Our experiments show that 1) the EP-Trie-based method predicts the same result as the enumeration-based method's, and 2) the former costs 0.11 second per image while the latter costs 2.566 seconds per image, which demonstrates the excellent efficiency of EP-Trie.

5. Conclusion

In this work, we propose a new method called edit probability for accurate scene text recognition. The new method can effectively handle the misalignment problem between the training text and the output probability distribution sequence caused by missing or superfluous characters. We conduct extensive experiments over several benchmarks to validate the proposed method, and experimental results show that EP can significantly improve recognition performance. In the future, we plan to apply the EP idea to machine translation, speech recognition, image/video caption and other related tasks.

6. Acknowledgment

Fan Bai and Shuigeng Zhou were partially supported by the Science and Technology Innovation Action Program of Science and Technology Commission of Shanghai Municipality (STCSM) under grant No. 17511105204.

References

[1] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word Spotting and Recognition with Embedded Attributes. *TPAMI*, 36(12):2552–2566, 2014.

[2] O. Alsharif and J. Pineau. End-to-end text recognition with hybrid hmm maxout models. In *ICLR*, 2014.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*, 2015.

[4] D. Bahdanau, J. Chorowski, D. Serdyuk, Y. Bengio, et al. End-to-end attention-based large vocabulary speech recognition. In *ICASSP*, pages 4945–4949, 2016.

[5] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio. End-to-end attention-based large vocabulary speech recognition. In *ICASSP*, pages 4945–4949, 2016.

[6] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. PhotoOCR: Reading Text in Uncontrolled Conditions. In *ICCV*, pages 785–792, 2013.

[7] 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 5076–5084, 2017.

[8] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[9] R. De La Briandais. File searching using variable length keys. In *Western Joint Computer Conference*, pages 295–298, 1959.

[10] V. Goel, A. Mishra, K. Alahari, and C. V. Jawahar. Whole is Greater than Sum of Parts: Recognizing Scene Text Words. In *ICDAR*, pages 398–402, 2013.

[11] A. Gordo. Supervised mid-level features for word image representation. In *CVPR*, pages 2956–2964, 2015.

[12] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*, pages 369–376, 2006.

[13] A. Graves, A. r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649, 2013.

[14] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic Data for Text Localisation in Natural Images. In *CVPR*, pages 2315–2324, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016.

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

[17] Hunspell. `http://hunspell.github.io`.

[18] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. *arXiv preprint arXiv:1406.2227*, 2014.

[19] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Structured Output Learning for Unconstrained Text Recognition. In *ICLR*, 2015.

[20] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading Text in the Wild with Convolutional Neural Networks. *IJCV*, 116(1):1–20, 2016.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *ACM MM*, pages 675–678, 2014.

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

[23] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i. Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazàn, and L. P. de las Heras. ICDAR 2013 Robust Reading Competition. In *ICDAR*, pages 1484–1493, 2013.

[24] S. Kim, T. Hori, and S. Watanabe. Joint CTC-Attention based End-to-End Speech Recognition using Multi-task Learning. In *ICASSP*, pages 4835–4839, 2017.

[25] C. Y. Lee and S. Osindero. Recursive Recurrent Nets with Attention Modeling for OCR in the Wild. In *CVPR*, pages 2231–2239, 2016.

[26] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 robust reading competitions. In *ICDAR*, pages 682–687, 2003.

[27] A. Mishra, K. Alahari, and C. V. Jawahar. Scene Text Recognition using Higher Order Language Priors. In *BMVC*, pages 1–11, 2012.

[28] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *CVPR*, pages 3538–3545, 2012.

[29] J. A. Rodriguez-Serrano, A. Gordo, and F. Perronnin. Label Embedding: A Frugal Baseline for Text Recognition. *IJCV*, 113(3):193–207, 2015.

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1, 1988.

[31] 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 TPAMI*, 39(11):2298–2304, 2017.

[32] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai. Robust Scene Text Recognition with Automatic Rectification. In *CVPR*, pages 4168–4176, 2016.

[33] B. Su and S. Lu. Accurate Scene Text Recognition Based on Recurrent Neural Network. In *ACCV*, pages 35–48, 2015.

[34] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS*, pages 3104–3112, 2014.

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

[36] K. Wang and S. Belongie. Word Spotting in the Wild. In *ECCV*, pages 591–604. Springer, 2010.

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

[38] C. Yao, X. Bai, B. Shi, and W. Liu. Strokelets: A Learned Multi-scale Representation for Scene Text Recognition. In *CVPR*, pages 4042–4049, 2014.

[39] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701, 2012.