Aggregation Cross-Entropy for Sequence Recognition

Zecheng Xie*, Yaoxiong Huang*, Yuanzhi Zhu, Lianwen Jin†, Yuliang Liu, Lele Xie
South China University of Technology
{zcheng.xie,hwang.yaoxiong,lianwen.jin,zzz.yuanzhi,shaxiaoai18,arlog.lele}@gmail.com

Abstract

In this paper, we propose a novel method, aggregation cross-entropy (ACE), for sequence recognition from a brand new perspective. The ACE loss function exhibits competitive performance to CTC and the attention mechanism, with much quicker implementation (as it involves only four fundamental formulas), faster inference\back-propagation (approximately $O(1)$ in parallel), less storage requirement (no parameter and negligible runtime memory), and convenient employment (by replacing CTC with ACE). Furthermore, the proposed ACE loss function exhibits two noteworthy properties: (1) it can be directly applied for 2D prediction by flattening the 2D prediction into 1D prediction as the input and (2) it requires only characters and their numbers in the sequence annotation for supervision, which allows it to advance beyond sequence recognition, e.g., counting problem. The code is publicly available at https://github.com/summerlvsong/Aggregation-Cross-Entropy.

1. Introduction

Sequence recognition, or sequence labelling [13] is to assign sequences of labels, drawn from a fixed alphabet, to sequences of input data, e.g., speech recognition[14, 2], scene text recognition [38, 39], and handwritten text recognition [34, 48], as shown in Fig. 1. The recent advances in deep learning [30, 41, 20] and the new architectures [42, 5, 4, 46] enabled the construction of systems that can handle one-dimensional (1D) [38, 34] and two-dimensional (2D) prediction problems [56, 4]. For 1D prediction problems, the topmost feature maps of the network are collapsed across the vertical dimension to generate 1D prediction [5] because characters in the original images are generally distributed sequentially. Typical examples are regular scene text recognition [38, 54], online/offline handwritten text recognition [12, 34, 48], and speech recognition [14, 2]. For 2D prediction problems, characters in the input image are distributed in a specific spatial structure. For example, there are highly complicated spatial relations between adjacent characters in mathematical expression recognition [56, 57]. In paragraph-level text recognition, characters are generally distributed line by line [4, 46], whereas in irregular scene text recognition, they are generally distributed in a side-view or curved angle pattern [51, 8].

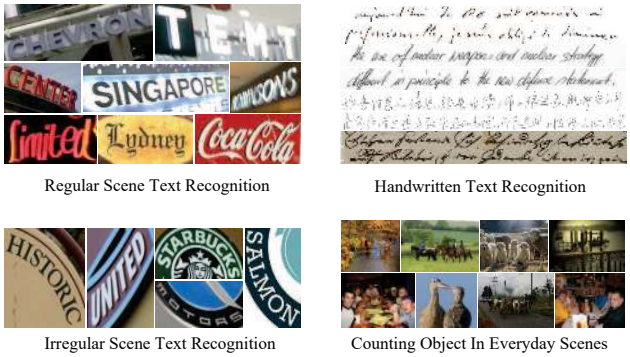

Figure 1. Examples of sequence recognition and counting problems.

For the sequence recognition problem, traditional methods generally require to separate training targets for each segment or time-step in the input sequence, resulting in inconvenient pre-segmentation and post-processing stages [12]. The recent emergence of CTC [13] and attention mechanism [1] significantly alleviate this sequential training problem by circumventing the prior alignment between input image and their corresponding label sequence. However, although CTC-based networks have exhibited remarkable performance in 1D prediction problem, the underlying methodology is sophisticated; moreover, its implementation, the forward-backward algorithm [12], is complicated, resulting in large computation consumption. Besides, CTC can hardly be applied to 2D prediction problems. Meanwhile, the attention mechanism relies on its attention module for label alignment, resulting in additional storage requirement and computation consumption. As pointed out by Bahdanau *et al.* [2], recognition model is difficult to learn from scratch with attention mechanism, due to the

*Zecheng Xie and Yaoxiong Huang make equal contribution.
†Corresponding auther.

misalignment between ground truth strings and attention predictions, especially on longer input sequences [25, 9]. Bai *et al.* [3] also argues that the misalignment problem can confuse and mislead the training process, and consequently make the training costly and degrade recognition accuracy. Although the attention mechanism can be adapted for 2D prediction problem, it turns out to be prohibitive in terms of memory and time consumption, as indicated in [4] and [46].

Compelled by the above observations, we propose a novel aggregation cross-entropy (ACE) loss function for the sequence recognition problem, as detailed in Fig. 2. Given the prediction of the network, the ACE loss consists of three simple stages: (1) aggregation of the probabilities for each category along the time dimension; (2) normalization of the accumulative result and label annotation as probability distributions over all the classes; and (3) comparison between these two probability distributions using cross-entropy. The advantages of the proposed ACE loss function can be summarized as follows:

- Owing to its simplicity, the ACE loss function is much quicker to implement (four fundamental formulas), faster to infer and back-propagate (approximately $O(1)$ in parallel), less memory demanding (no parameter and basic runtime memory), and convenient to use (simply replace CTC with ACE), as compared to CTC and attention mechanism. This is illustrated in Table 5, Section 3.4, and Section 4.4.
- Despite its simplicity, the ACE loss function achieves competitive performance to CTC and the attention mechanism, as established in experiments of regular\irregular scene text recognition and handwritten text recognition problems.
- The ACE loss function can be adapted to the 2D prediction problem by flattening the 2D prediction into 1D prediction, as verified in the experiments of irregular scene text recognition and counting problems.
- The ACE loss function does not require instance order information for supervision, which enable it to advance beyond sequence recognition, e.g., counting problem.

2. Related Work

2.1. Connectionist temporal classification

The advantages of the popular CTC loss were first demonstrated in speech recognition [16, 14] and online handwritten text recognition [15, 12]. Recently, an integrated CNN-LSTM-CTC model was proposed to address the scene text recognition problem [38]. There are also methods that aim to extend CTC in applications; e.g., Zhang *et al.* [55] proposed an extended CTC (ECTC) objective function adapted from CTC to allow RNN-based phoneme recognizers to be trained even when only word-level annotation is available. Hwang *et al.* [21] developed an expectation-maximization-based online CTC algorithm that allows RNNs to be trained with an infinitely long input sequence, without pre-segmentation or external reset. However, the calculation process of CTC is highly complicated and time-consuming, and it require substantial effort to rearrange the feature map and annotation when applied to 2D problems [46, 4].

2.2. Attention mechanism

The attention mechanism was first proposed in machine translation [1, 42] to enable a model to automatically search for parts of a source sentence for prediction. Then, the method rapidly became popular in applications such as (visual) question answering [32, 52], image caption generation [50, 52, 31], speech recognition [2, 25, 32] and scene text recognition [39, 3, 19]. Most importantly, the attention mechanism can also be applied to 2D predictions, such as mathematical expression recognition [56, 57] and paragraph recognition [4, 5, 46]. However, the attention mechanism relies on a complex attention module to fulfill its functionality, resulting in additional network parameters and runtime. Besides, missing or superfluous characters can easily cause misalignment problem, confusing and misleading the training process, and consequently degrading the recognition accuracy [3, 2, 9].

3. Aggregation Cross-Entropy

Formally, given the input image $\mathcal{I}$ and its sequence annotation $\mathcal{S}$ from a training set $\mathcal{Q}$, the general loss function for the sequence recognition problem evaluates the probability of annotation $\mathcal{S}$ of length L conditioned on image $\mathcal{I}$ under model parameter ω as follows:

$$\begin{aligned}\mathcal{L}(\omega) &= -\sum_{(\mathcal{I},\mathcal{S})\in\mathcal{Q}} \log P(\mathcal{S}|\mathcal{I};\omega) \\ &= -\sum_{(\mathcal{I},\mathcal{S})\in\mathcal{Q}} \sum_{l=1}^{L} \log P(\mathcal{S}_l|l,\mathcal{I};\omega) \end{aligned} \tag{1}$$

where $P(\mathcal{S}_l|l,\mathcal{I};\omega)$ represents the probability of predicting character S_l at the l-th position of the predicted sequence. Therefore, the problem is to estimate the general loss function Eq. (1) based on the model prediction $\{y_k^t, t = 1, 2, \cdots, T, k = 1, 2, \cdots, |\mathcal{C}^\epsilon|\}$, where $\mathcal{C}^\epsilon = \mathcal{C} \cup \epsilon$, with $\mathcal{C}$ being the character set and ϵ the blank label. Nevertheless, directly estimating the probability $P(\mathcal{S}|\mathcal{I};\omega)$ was excessively challenging until the emergence of the popular CTC loss function. The CTC loss function elegantly calculates $P(\mathcal{S}|\mathcal{I};\omega)$ using a forward-backward algorithm, which removes the need for pre-segmented data and external post-processing. The attention mechanism provides an alternative solution to estimate the general loss function by directly predicting $P(\mathcal{S}_l|l,\mathcal{I};\omega)$ based on its attention module. However, the forward-backward algorithm of CTC is

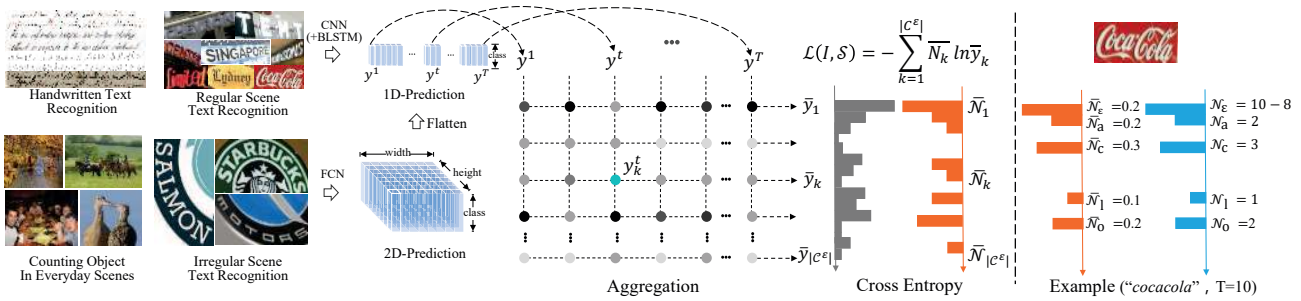

Figure 2. (Left) Illustration of proposed ACE loss function. Generally, the 1D and 2D predictions are generated by integrated CNN-LSTM and FCN model, respectively. For the ACE loss function, the 2D prediction is further flattened to 1D prediction, $\{y_k^t, t = 1, 2, \ldots, T\}$. During aggregation, the 1D predictions at all time-steps are accumulated for each class independently, according to $y_k = \sum_{t=1}^{T} y_k^t$. After normalization, the prediction $\overline{y}$, together with the ground-truth $\overline{\mathcal{N}}$, is utilized for loss estimation based on cross-entropy. (Right) A simple example indicates the generation of annotation for the ACE loss function. $\mathcal{N}_a = 2$ implies that there are two "a" in *cocacola*.

highly complicated and time-consuming whereas the attention mechanism requires extra complex network to ensure the alignment between attention prediction and annotation.

In this paper, we present the ACE loss function to estimate the general loss function based on model prediction y_k^t. In Eq. (1), the general loss function can be minimized by maximizing the predictions at each position of the sequence annotation, i.e., $P(\mathcal{S}_l|l, \mathcal{I}; \omega)$. However, directly calculating $P(\mathcal{S}_l|l, \mathcal{I}; \omega)$ based on y_k^t is challenging because the alignment between the l-th character in the annotation and model prediction y_k^t is unclear. Therefore, rather than precisely estimating the probability $P(\mathcal{S}_l|l, \mathcal{I}; \omega)$, the problem is mitigated by supervising only the accumulative probability of each class; without considering its sequential order in the annotation. For example, if a class appears twice in the annotation, we require its accumulative prediction probability over T time-steps to be exactly two, anticipating that its two corresponding predictions approximate to one. Therefore, we can minimize the general loss function by requiring the network to precisely predict the character number of each class in the annotation as follows:

$$\begin{aligned}\mathcal{L}(\omega) &= -\sum_{(\mathcal{I},\mathcal{S})\in\mathcal{Q}} \sum_{l=1}^{L} \log P(\mathcal{S}_l|l, \mathcal{I}; \omega) \\ &\approx -\sum_{(\mathcal{I},\mathcal{S})\in\mathcal{Q}} \sum_{k=1}^{|\mathcal{C}^\epsilon|} \log P(\mathcal{N}_k|\mathcal{I}; \omega) \end{aligned} \tag{2}$$

where $\mathcal{N}_k$ represents the number of times that character $\mathcal{C}_k^\epsilon$ occurs in the sequence annotation $\mathcal{S}$. Note that this new loss function does not require character order information but only the classes and their number for supervision.

3.1. Regression-Based ACE Loss Function

Now, the problem is to bridge model prediction y_k^t to the number prediction of each class. We propose to calculate the number of each class y_k by summing up the probabilities of the k-th characters for T time-steps, i.e., $y_k = \sum_{t=1}^{T} y_k^t$, as illustrated by *aggregation* in Fig. 2. Note that,

$$\max \sum_{k=1}^{|\mathcal{C}^\epsilon|} \log P(\mathcal{N}_k|\mathcal{I}; \omega) \Leftrightarrow \min \sum_{k=1}^{|\mathcal{C}^\epsilon|} (\mathcal{N}_k - y_k)^2 \tag{3}$$

Therefore, we adapt the loss function (Eq. (2)) from the perspective of regression problem as follows:

$$\mathcal{L}(\omega) = \frac{1}{2} \sum_{(\mathcal{I},\mathcal{S})\in\mathcal{Q}} \sum_{k=1}^{|\mathcal{C}^\epsilon|} (\mathcal{N}_k - y_k)^2. \tag{4}$$

Also note that a total of $(T - |\mathcal{S}|)$ predictions are expected to yield null emission. Therefore, we have $\mathcal{N}_\epsilon = T - |\mathcal{S}|$.

To find the gradient for each example $(\mathcal{I}, \mathcal{S})$, we first differentiate $\mathcal{L}(\mathcal{I}, \mathcal{S})$ with respect to the network output y_k^t:

$$\frac{\partial \mathcal{L}(\mathcal{I}, \mathcal{S})}{\partial y_k^t} = \Delta_k, \tag{5}$$

where $\Delta_k = (y_k - \mathcal{N}_k)$. Recall that for Softmax functions, we have:

$$y_i = \frac{e^{a_i}}{\sum_j e^{a_j}}, \quad \frac{\partial y_i}{\partial a_j} = y_i(\delta_{ij} - y_j), \tag{6}$$

where $\delta_{ij} = 1$ if $i = j$ and zero otherwise. Now, we can differentiate the loss function with respect to a_k^t to back-propagate the gradient through the output layer:

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathcal{I}, \mathcal{S})}{\partial a_k^t} &= \sum_{k'=1}^{|\mathcal{C}^\epsilon|} \frac{\partial \mathcal{L}(\mathcal{I}, \mathcal{S})}{\partial y_{k'}^t} \frac{\partial y_{k'}^t}{\partial a_k^t} = \sum_{k'=1}^{|\mathcal{C}^\epsilon|} \Delta_{k'} \cdot y_{k'}^t(\delta_{kk'} - y_k^t) \\ &= \Delta_{k'} \cdot y_{k'}^t(1 - y_k^t) - \sum_{k'\neq k} \Delta_{k'} \cdot y_{k'}^t y_k^t \end{aligned} \tag{7}$$

3.1.1 Gradient vanishing problem

From Eq. (7), we observe that the regression-based ACE loss (Eq. (4)) is not convenient in term of back-propagation.

In the early training stage, we have $\{y^t_{k'} \approx 1/|\mathcal{C}^\epsilon|, \forall k', t\}$. Therefore, $y^t_{k'}$ will be negligible for large vocabulary sequence recognition problems, where $|\mathcal{C}^\epsilon|$ is large (e.g., 7,357 for the HCTR problem). Although the other terms in Eq. (7) (e.g., $\Delta_{k'}$) have acceptable magnitudes for back-propagation, the gradient would be scaled to a remarkably small size by the term $y^t_{k'}$ and y^t_k, resulting in gradient vanishing problem.

3.2. Cross-Entropy-Based ACE Loss Function

To prevent the gradient-vanishing problem, It is necessary to offset the negative effect of the term $y^t_{k'}$ introduced by the Softmax function in Eq. (7). We borrow the concept of *cross-entropy* from information theory, which is designed to measure the "distance" between two probability distributions. Therefore, we first normalize the accumulative probability of the k-th character y_k to $\overline{y}_k = y_k/T$, and the character numbers $\mathcal{N}_k$ to $\overline{\mathcal{N}}_k = \mathcal{N}_k/T$. Then, the cross-entropy between $\overline{y}$ and $\overline{\mathcal{N}}$ is expressed as:

$$\mathcal{L}(\mathcal{I}, \mathcal{S}) = -\sum_{k=1}^{|\mathcal{C}^\epsilon|} \overline{\mathcal{N}}_k \ln \overline{y}_k \tag{8}$$

The loss function derivatives with respect to a^t_k before the Softmax activation function has the following form:

$$\begin{aligned}
\frac{\partial \mathcal{L}(\mathcal{I}, \mathcal{S})}{\partial a^t_k} &= \sum_{k'=1}^{|\mathcal{C}^\epsilon|} \frac{\partial \mathcal{L}(\mathcal{I}, \mathcal{S})}{\partial \overline{y}_{k'}} \frac{\partial \overline{y}_{k'}}{\partial y^t_{k'}} \frac{\partial y^t_{k'}}{\partial a^t_k} \\
&= \sum_{\mathcal{C}^\epsilon_{k'} \in \mathcal{S}} -\frac{\overline{\mathcal{N}}_{k'}}{\overline{y}_{k'}} \cdot \frac{1}{T} \cdot y^t_{k'}(\delta_{kk'} - y^t_k) \\
&= -\frac{1}{T} \sum_{\mathcal{C}^\epsilon_{k'} \in \mathcal{S}} \overline{\mathcal{N}}_{k'} \frac{y^t_{k'}}{\overline{y}_{k'}} (\delta_{kk'} - y^t_k) \\
&= -\frac{1}{T} \sum_{\mathcal{C}^\epsilon_{k'} \in \mathcal{S}} \overline{\mathcal{N}}_{k'} \frac{y^t_{k'}}{\overline{y}_{k'}} (\delta_{kk'} - y^t_k)
\end{aligned} \tag{9}$$

3.2.1 Discussion

In the following, we explain how the updated loss function solves the gradient vanishing problem:

(1) In the early training stage, $y^t_{k'}$ has an approximately identical order of magnitude at all the time-steps. Thus, the normalized accumulated probability $\overline{y}_{k'}$ is also of an identical order of magnitude as $y^t_{k'}$. That is, $\frac{y^t_{k'}}{\overline{y}_{k'}} \approx 1$; therefore, the gradient through the k'-th class is now $-\frac{1}{T}\overline{\mathcal{N}}_{k'}(\delta_{kk'} - y^t_k)$. Thus, the gradient can straightforwardly back-propagate to a^t_k through the characters that appear in sequence annotation $\mathcal{S}$. Besides, when $k = k'$, i.e., $\mathcal{C}^\epsilon_k \in \mathcal{S}$; the corresponding gradient is approximately $-\frac{1}{T}\overline{\mathcal{N}}_{k'}(1 - y^t_k)$, which will encourage the model to make a larger prediction y^t_k, whereas characters that do not appear in $\mathcal{S}$ become smaller. This was our original intention.

(2) In the later training stage, only a few of the prediction $y^{t^*}_{k^*}$ will be very large, leaving the other predictions small enough to be omitted. In this situation, prediction $y^{t^*}_{k^*}$ will occupy the majority of y_{k^*}, and we have $\frac{y^{t^*}_{k^*}}{\overline{y}_{k^*}} = T \cdot \frac{y^{t^*}_{k^*}}{y_{k^*}}$. Therefore, when $\mathcal{C}^\epsilon_{k^*} \in \mathcal{S}$, the gradient can be straightforwardly back-propagated to the recognition network.

3.3. Two-dimensional Prediction

In some 2D prediction problem like irregular scene text recognition with image level annotations, it is challenging to define the spatial relation between characters. Characters may be arranged in multiple lines, in a curved or sloped direction, or even distributed in a random manner. Fortunately, the proposed ACE loss function can naturally be generalized for the 2D prediction problem, because it does not require character-order information for the sequence-learning process.

Suppose that the output 2D prediction $\boldsymbol{y}$ has height $\mathcal{H}$ and width $\mathcal{W}$, and the prediction at the h-th line and w-th row is denoted as y^{hw}_k. This requires a marginal adaptation of the calculation of $\overline{y}_k$ and $\overline{\mathcal{N}}_k$ as follows, $\overline{y}_k = \frac{y_k}{\mathcal{HW}} = \frac{\sum_{h=1}^{\mathcal{H}} \sum_{w=1}^{\mathcal{W}} y^{hw}_k}{\mathcal{HW}}$, $\overline{\mathcal{N}}_k = \frac{\mathcal{N}_k}{\mathcal{HW}}$. Then, the loss function for the 2D prediction can be transformed as follows:

$$\mathcal{L}(\mathcal{I}, \mathcal{S}) = -\sum_{k=1}^{|\mathcal{C}^\epsilon|} \overline{\mathcal{N}}_k \ln \overline{y}_k = -\sum_{k=1}^{|\mathcal{C}^\epsilon|} \frac{\mathcal{N}_k}{\mathcal{HW}} \ln \frac{y_k}{\mathcal{HW}} \tag{10}$$

In our implementation, we directly flatten the 2D prediction $\{y^{hw}, h = 1, 2, \cdots, \mathcal{H}, w = 1, 2, \cdots, \mathcal{W}\}$ into 1D prediction $\{y^t, t = 1, 2, \cdots, T\}$, where $T = \mathcal{HW}$, and then apply Eq. (8) to calculate the final loss.

3.4. Implementation and Complexity Analysis

Implementation As illustrated in Eq. (2), $\mathcal{N} = \{\mathcal{N}_k | k = 1, 2, \cdots, |\mathcal{C}^\epsilon|\}$ represents the annotation for the ACE loss function; here, $\mathcal{N}_k$ represents the number of times that the character $\mathcal{C}^\epsilon_k$ occurs in the sequence annotation $\mathcal{S}$. A simple example describing the translation of sequence annotation *cocacola* into ACE's annotation is shown in Fig. 2. In conclusion, given the model prediction y^t_k and its annotation $\mathcal{N}$, the key implementation for a cross-entropy-based ACE loss function consists of four fundamental formulas:

- $y_k = \sum_{t=1}^{T} y^t_k$ to calculate the character number of each class by summing up the probabilities of the k-th class for all T time-steps.
- $\overline{y}_k = y_k/T$ to normalize the accumulative probabilities.
- $\overline{\mathcal{N}}_k = \mathcal{N}_k/T$ to normalize the annotation.
- $\mathcal{L}(\mathcal{I}, \mathcal{S}) = -\sum_{k=1}^{|\mathcal{C}^\epsilon|} \overline{\mathcal{N}}_k \ln \overline{y}_k$ to estimate the cross-entropy between $\overline{\mathcal{N}}_k$ and $\overline{y}_k$.

In practical employment, the model prediction y_k^t is generally provided by the integrated CNN-LSTM model (1D prediction) or FCN model (flattened 2D prediction). That is, the input assumption of ACE is identical to that of CTC; therefore, the proposed ACE can be conveniently applied by replacing the CTC layer in the framework.

Complexity Analysis The overall computation of the ACE loss function is implemented based on the abovementioned four formulas that have computation complexities of $O(1)$, $O(|\mathcal{C}^\epsilon|)$, $O(|\mathcal{C}^\epsilon|)$, and $O(|\mathcal{C}^\epsilon|)$, respectively. Therefore, the computation complexity of the ACE loss function is $O(|\mathcal{C}^\epsilon|)$. Note however that the element-wise multiplication, division, and log operation in these four formulas can be implemented in parallel with GPU at $O(1)$. In contrast, the implementation of CTC [12] based on a forward-backward algorithm has a computation complexity of $O(T * |\mathcal{S}|)$. Because the *forward variable* $\alpha(t, u)$ and *backward variable* $\beta(t, u)$ [12] of CTC depend on the previous result (e.g., $\alpha(t-1, u)$ and $\beta(t+1, u)$) to calculate the present output, CTC can hardly be accelerated in parallel in the time dimension. Moreover, the elementary operation $\alpha(t, u)$ of CTC is already very complicated, resulting in larger overall time consumption than that of ACE. With regard to the attention mechanism, its computation complexity is proportional to the times of 'attention'. However, the computation complexity of the attention module at each time already has similar magnitude as that of CTC.

From the perspective of memory consumption, the proposed ACE loss function requires nearly no memory consumption because the ACE loss result can be directly calculated based on the four fundamental formulas. However, CTC requires additional space to preserve the *forward\backward variable* that is proportional to the timestep T and the length of the sequence annotation. Meanwhile, the attention mechanism requires additional module to implement "attention". Thus, its memory consumption is significantly larger than that of CTC and ACE.

In conclusion, the proposed ACE loss function exhibits significant advantages with regard to both computation complexity and memory demand, as compared to CTC and attention.

4. Performance Evaluation

In our experiment, three tasks were employed to evaluate the effectiveness of the proposed ACE loss function, including scene text recognition, offline handwritten Chinese text recognition, and counting objects in everyday scenes. For these tasks, we estimated the ACE loss for 1D and 2D predictions, where 1D implies that the final prediction is a sequence of T predictions and 2D indicates that the final feature map has 2D predictions of shape $\mathcal{H} \times \mathcal{W}$.

4.1. Scene Text Recognition

Scene text recognition often encounter problems owing to the large variations in the background, appearance, resolution, text font, and color, making it a challenging research topic. In this section, we study both 1D and 2D predictions on scene text recognition by utilizing the richness and variety of the testing benchmark in this task.

4.1.1 Dataset

Two types of datasets are used for scene text recognition: regular text datasets, such as IIIT5K-Words [35], Street View Text [43], ICDAR 2003 [33], and ICDAR 2013 [24], and irregular text datasets, such as SVT-Perspective [36], CUTE80 [37], and ICDAR 2015 [23]. The regular datasets were used to study the 1D prediction for the ACE loss function while the irregular text datasets were applied to evaluate the 2D prediction.

IIIT5K-Words (*IIIT5K*) contains 3000 cropped word images for testing.

Street View Text (*SVT*) was collected from Google Street View, including 647 word images. Many of them are severely corrupted by noise and blur, or have very low resolutions.

ICDAR 2003 (*IC03*) contains 251 scene images that are labeled with text bounding boxes. The dataset contains 867 cropped images.

ICDAR 2013 (*IC13*) inherits most of its samples from IC03. It contains 1015 cropped text images.

SVT-Perspective (*SVT-P*) contains 639 cropped images for testing, which are selected from side-view angle snapshots from Google Street View. Therefore, most of images are perspective distorted. Each image is associated with a 50-word lexicon and a full lexicon.

CUTE80 (*CUTE*) contains 80 high-resolution images taken of natural scenes. It was specifically collected for curve text recognition. The dataset contains 288 cropped natural images for testing. No lexicon is associated.

ICDAR 2015 (*IC15*) contains 2077 cropped images including more than 200 irregular text. No lexicon is associated.

4.1.2 Implementation Details

For 1D sequence recognition on regular datasets, our experiments were based on the CRNN [38] network, trained only on 8-million synthetic data released by Jaderberg *et al.* [22]. For 2D sequence recognition on irregular datasets, our experiments were based on the ResNet-101 [18], with conv1 changed to 3×3, stride 1, and conv4_x as output. The training dataset consists of 8-million synthetic data released by Jaderberg *et al.* [22] and 4-million synthetic instances (excluding the images that contain non-alphanumeric characters) cropped from 80-thousand images [17]. The input

Table 1. Comparison between regression and cross-entropy.

Method	IIIT5K	SVT	IC03	IC13
Shi *et al.* [38]	81.2	**82.7**	91.9	89.6
ACE (1D, Regression)	19.4	6.6	12.0	9.3
ACE (1D, Cross Entropy)	**82.3**	82.6	**92.1**	**89.7**

images are normalized to the shape of (96,100) and the final 2D prediction has the shape of (12,13), as shown in Fig. 5. To decode the 2D prediction, we flattened the 2D prediction by concatenating each column in order from left to right and top to bottom and then decoded the flattened 1D prediction following the general procedure.

In our experiment, we observed that directly normalizing the input image to the size of (96,100) overloads the network training process. Therefore, we trained another network to predict the character number in the text image and normalized the text image with respect to the character number to keep the character size within acceptable limits.

4.1.3 Experimental Result

To study the role of regression and cross-entropy for the ACE loss function, we conducted experiments with 1D prediction using regular scene text datasets, as detailed in Table 1 and Fig. 3. Because there are only 37 classes in scene text recognition, the negative effect of the term $y^t_{k'}$ in Eq. (7) is not as significant as that of the HCTR problem (7357 classes). As shown in Fig. 3, with regression-based ACE loss, the network can converge but at a relatively slow rate, probably due to the gradient vanishing problem. With cross-entropy-based ACE loss, the WER and CER evolve at a relatively higher rate and in a smoother manner at the early training stage and attain a significantly better convergence result in the subsequent training stage. Table 1 clearly reveals the superiority of the cross-entropy-based ACE loss function over the regression-based one. Therefore, we use cross-entropy-based ACE loss functions for all the remaining experiments. Moreover, with the same network setting (CRNN) and training set (8-million synthetic data), the proposed ACE loss function exhibits performance comparable

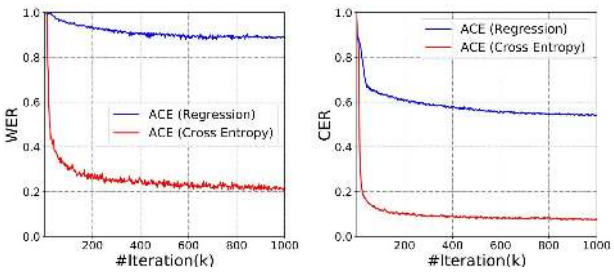

Figure 3. Word error rate (left) and character error rate (right) of ACE loss on validation set under regression and cross entropy perspective.

Table 2. Comparison with previous methods for scene text recognition problem (without rectification)

Method	2D	SVT-P			CUTE	IC15
		50	Full	None	None	None
Shi *et al.* [38]		92.6	72.6	66.8	54.9	-
Liu *et al.* [28]		94.3	83.6	73.5	-	-
Yang *et al.* [51]	✓	93.0	80.2	**75.8**	69.3	-
Cheng *et al.* [7]	✓	92.6	81.6	71.5	63.9	66.2
Cheng *et al.* [8]	✓	94.0	83.7	73.0	76.8	68.2
Liu *et al.* [29]		–	–	73.9	62.5	–
Shi *et al.* [39]		–	–	74.1	73.3	–
ACE (2D)	✓	**94.9**	**87.8**	70.1	**82.6**	**68.9**

with that of previous work [38] with CTC.

To validate the independence of the proposed ACE loss to character order, we conduct experiments with ACE, CTC, and attention on four datasets; the character order of annotation is randomly shuffled at different ratios, as shown in Fig. 4. It is observed that the performance of attention and CTC on all the datasets degrades as the shuffle ratio increases. Specifically, attention is more sensitive than CTC because misalignment problem can easily misleads the training process of attention [3]. In contrast, the proposed ACE loss function exhibits similar recognition results for all the settings of the shuffle ratio, this is because it only requires classes and their number for supervision, completely omitting character order information.

For irregular scene text recognition, we conducted text recognition experiments with 2D prediction. In Table 2, we provide a comparison with previous methods that considered only recognition model and no rectification for fair comparison. As illustrated in Table 2, the proposed ACE loss function exhibits superior performance on the datasets CUTE and IC15, particularly on CUTE with an absolute error reduction of 5.8%. This is because the dataset CUTE was specifically collected for curved text recognition and therefore, fully demonstrates the advantages of the ACE loss function. For the dataset SVT-P, our naive decoding result is less effective than that of Yang *et al.* [51]. This is because numerous images in the dataset SVT-P have

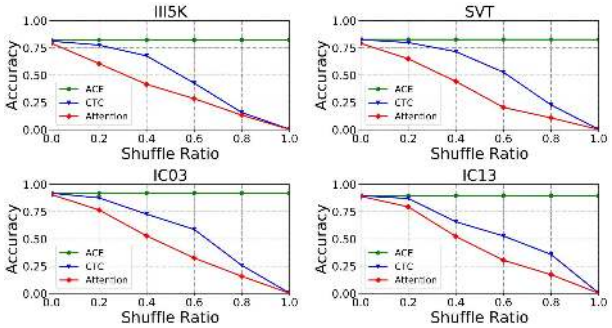

Figure 4. Performance of ACE, CTC, and attention under different shuffle ratios and datasets.

very low resolutions, which creates a very high requirement for semantic context modeling. However, our network is based only on CNN, with neither LSTM/MDLSTM nor attention mechanism to leverage the high-level semantic context. Nevertheless, it is noteworthy that our recognition model achieved the highest result when using lexicons, with which semantic context is accessible. This again validates the robustness and effectiveness of the proposed ACE loss function.

In Fig. 5, we provide a few real images processed by a recognition model using the ACE loss function. The original text images were first normalized and placed in the center of a blank image of shape (96, 100). We observe that after recognition, the 2D prediction exhibits a spatial distribution highly similar to that of the characters in the original text image, which implies the effectiveness of the proposed ACE loss function.

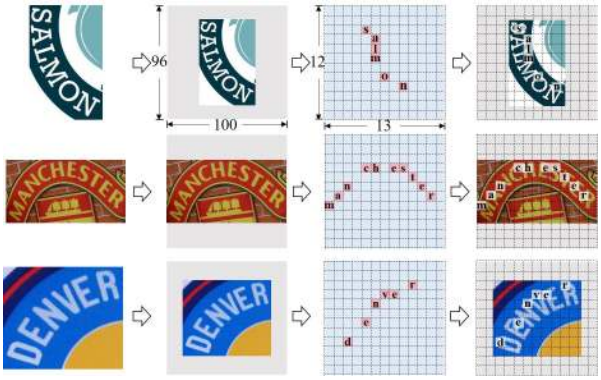

Figure 5. Real images processed by recognition model using ACE loss function. The left two columns represent original text images and their normalized versions within the shape of (96, 100). The third column shows the 2D prediction of the recognition model for the text images. In the right column, we overlap the input and the prediction images, and observe similar character distribution in the 2D space.

4.2. Offline Handwritten Chinese Text Recognition

Owing to its large character set (7,357 classes), diverse writing style, and character-touching problem, the offline HCTR problem is highly complicated and challenging to solve. Therefore, it is a favorable testbed to evaluate the robustness and effectiveness of the ACE loss in 1D predictions.

4.2.1 Implementation Details

For the offline HCTR problem, our model was trained using the CASIA-HWDB [26] datasets and tested with the standard benchmark ICDAR 2013 competition dataset [53].

For the HCTR problem, our network architecture with a prediction sequence of length 70 is specified as follows:

$(126, 576)Input - 8C3 - MP2 - 32C3 - MP2 - 128C3 - MP2 - 5{*}256C3 - MP2 - 512C3 - 512C3 - MP2 - 512C2 - 3 * 512ResLSTM - 7357FC - Output$,

where xCy represents a convolutional layer with kernel number of x and kernel size of $y * y$, MPy denotes a max pooling layer with kernel size of y, and xFC is a fully connected layer with kernel number of x, and ResLSTM is residual LSTM proposed in [49]. The evaluation criteria for the HCTR problem are correct rate (CR) and accuracy rate (AR) specified by ICDAR2013 competition [53].

4.2.2 Experimental Result

In Table 3, we provide the comparison between ACE loss and previous methods. It is evident that the proposed ACE loss function exhibits higher performance than previous methods, including MDLSTM-based models [34, 47], HMM-based model [10], and over-segmentation methods [27, 44, 45, 48] with and without language model (LM). Compared to scene text recognition, handwritten Chinese text recognition problem possesses its unique challenges, such as large character set (7357 classes) and character-touching problem. Therefore, the superior performance of ACE loss function over previous methods can properly verify its robustness and generality for sequence recognition problems.

Table 3. Comparison with previous methods for HCTR.

Method	w.o LM		with LM	
	CR	AR	CR	AR
HIT-2 [27]	–	–	88.76	86.73
Wang *et al.* [44]	–	–	91.39	90.75
Messina *et al.* [34]	–	83.50	–	89.40
Wu *et al.* [47]	87.43	86.64	–	92.61
Du *et al.* [10]	–	83.89	–	93.50
Wang *et al.* [45]	90.67	88.79	95.53	94.02
Wu *et al.* [48]	–	–	96.32	96.20
ACE (1D)	**91.68**	**91.25**	**96.70**	**96.22**

4.3. Counting Objects in Everyday Scenes

Counting the number of instances of object classes in natural everyday images generally encounters complex real life situations, e.g., large variance in counts, appearance, and scales of object. Therefore, we verified the ACE loss function on the problem of counting objects in everyday scenes to demonstrate its generality.

4.3.1 Implementation Details

As a benchmark for multi-label object classification and object detection tasks, the PASCAL VOC [11] datasets contain category labels per image, as well as bounding box annotations that can be converted to the object number labels. In our implementation, we accumulated the prediction

for category k to obtain $\hat{c}_{ik}$ by thresholding counts at zero and rounding predictions to the closest integers. Given these predictions and the ground truth counts c_{ik} for a category k and image i, *RMSE* and *relRMSE* is calculated by $\textit{RMSE}_k = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{c}_{ik} - c_{ik})^2}$ and $\textit{relRMSE}_k = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{(\hat{c}_{ik}-c_{ik})^2}{c_{ik}+1}}$.

4.3.2 Experimental Result

Table 4 presents a comparison between the proposed ACE loss function and previous methods for the PASCAL VOC 2007 test dataset for counting objects in everyday scenes. The proposed ACE loss function outperforms the previous glancing and subitizing method [6], correlation loss method [40], and Always-0 method (predicting most-frequent ground truth count). The results have shown the generality of ACE loss function, in that it can be readily applied to problem other than sequence recognition, e.g., counting problems, requiring minimal domain knowledge.

In Fig. 6, we provide real images processed by the counting model under ACE loss. As shown in the images, our counting model trained with ACE loss manage to pay "attention" to the position where crucial objects occur. Unlike the text recognition problem, where the recognition model trained with the ACE loss function tends to make a prediction for a character, the counting model trained with the ACE loss function provides a more uniform prediction distribution over the body of the object. Moreover, it assigns different levels of "attention" to different parts of an object. For example, when observing the red color in the pictures, we notice that the counting model pays more attention to the face of a person. This phenomenon corresponds to our intuition because the face is the most distinctive part of an individual.

Table 4. Comparison with previous methods on PASCAL VOC 2007 test dataset for object counting problem.

Method	m-RMSE	m-relRMSE
Always-0	0.665	0.284
Glance [6]	0.500	0.270
Sub-ens [6]	0.420	0.200
Two-stream [40]	0.389	0.189
ACE (2D)	**0.381**	**0.185**

4.4. Complexity Analysis

In Table 5, we compare the parameter, runtime memory, and run time of ACE with those of CTC and attention. The result is executed with minibatch 64 and model prediction length T=144 on a single NVIDIA TITAN X graphics card of 12GB memory. Similar to CTC, the proposed ACE does not require any parameter to fulfill its function. Owing to its simplicity, ACE requires marginal runtime memory, five times less than those for CTC and attention. Furthermore, its speed is as least 30 times higher than those of CTC and attention. It is note worthy that with all these advantages, the proposed ACE achieve performance that is comparable or higher than those with CTC and attention.

Figure 6. Real images processed by counting model using ACE loss function. The first four columns display examples that are correctly recognized by our model. The top-right image is correctly recognized, but with an incorrectly annotated label. (Incorrect predictions are provided with labels in brackets)

Table 5. Investigation over parameter (Para), runtime memory (Mem), and speed (Speed) (in units of MB, MB, and ms, respectively) of CTC, attention, and ACE.

Method	37 classes			7357 classes		
	Para	Mem	Time	Para	Mem	Time
CTC	none	0.1	3.1	none	47.8	16.2
Attention	2.8	6.6	78.9	17.2	143.6	85.5
ACE	**none**	**0.02**	**<0.1**	**none**	**4.2**	**<0.1**

5. Conclusion

In this paper, a novel and straightforward ACE loss function is proposed for sequence recognition problem with competitive performance to CTC and attention. Owing to its simplicity, the ACE loss function is easy to employ by simply replacing CTC with ACE, quick to implement with only four basic formulas, fast to infer and back-propagate at approximately $O(1)$ in parallel, and exhibits marginal memory requirement. Two following effective properties of ACE loss function are also investigated: (1) it can easily handle 2D prediction problem with marginal adaption and (2) it does not require character-order information for supervision, which allows it to advance beyond sequence recognition problem, e.g., counting problem.

Acknowledgments

This research is supported in part by GD-NSF (no. 2017A030312006), the National Key Research and Development Program of China (No. 2016YFB1001405), NSFC (Grant No.: 61673182, 61771199), and GDSTP (Grant No.:2017A010101027) , GZSTP(no. 201704020134).

References

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

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

[3] F. Bai, Z. Cheng, Y. Niu, S. Pu, and S. Zhou. Edit probability for scene text recognition. *CVPR*, 2018.

[4] T. Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. *NIPS*, 2016.

[5] T. Bluche and R. Messina. Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention. *ICDAR*, 2016.

[6] P. Chattopadhyay, R. Vedantam, R. R. Selvaraju, D. Batra, and D. Parikh. Counting everyday objects in everyday scenes. In *CVPR*, 2017.

[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*, 2017.

[8] Z. Cheng, X. Liu, F. Bai, Y. Niu, S. Pu, and S. Zhou. Arbitrarily-oriented text recognition. *ICDAR*, 2017.

[9] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In *NIPS*, 2015.

[10] J. Du, Z.-R. Wang, J.-F. Zhai, and J.-S. Hu. Deep neural network based hidden markov model for offline handwritten chinese text recognition. In *ICPR*.

[11] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2015.

[12] A. Graves. *Supervised sequence labelling*. Springer, 2012.

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

[14] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. *ICML*, 2014.

[15] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE TPAMI*, 2009.

[16] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

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

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[19] P. He, W. Huang, Y. Qiao, C. C. Loy, and X. Tang. Reading scene text in deep convolutional sequences. *AAAI*, 2016.

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

[21] K. Hwang and W. Sung. Sequence to sequence training of ctc-rnns with partial windowing. In *ICML*, 2016.

[22] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *CoRR*, abs/1406.2227, 2014.

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

[24] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. De Las Heras. Icdar 2013 robust reading competition. In *ICDAR*, 2013.

[25] S. Kim, T. Hori, and S. Watanabe. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *ICASSP*, 2017.

[26] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang. Casia online and offline chinese handwriting databases. *ICDAR*, 2011.

[27] C.-L. Liu, F. Yin, Q.-F. Wang, and D.-H. Wang. ICDAR 2011 chinese handwriting recognition competition (2011).

[28] W. Liu, C. Chen, K.-Y. K. Wong, Z. Su, and J. Han. Star-net: A spatial attention residue network for scene text recognition. In *BMVC*, 2016.

[29] Y. Liu, Z. Wang, H. Jin, and I. Wassell. Synthetically supervised feature learning for scene text recognition. In *ECCV*, 2018.

[30] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

[31] J. Lu, C. Xiong, D. Parikh, and R. Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*, 2017.

[32] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, 2016.

[33] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, R. Young, K. Ashida, H. Nagai, M. Okamoto, H. Yamamoto, et al. Icdar 2003 robust reading competitions: entries, results, and future directions. *IJDAR*, 2005.

[34] R. Messina and J. Louradour. Segmentation-free handwritten chinese text recognition with lstm-rnn. *ICDAR*, 2015.

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

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

[37] A. Risnumawan, P. Shivakumara, C. S. Chan, and C. L. Tan. A robust arbitrary text detection system for natural scene images. *Expert Systems with Applications*, 2014.

[38] 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*, 2016.

[39] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai. Aster: An attentional scene text recognizer with flexible rectification. *IEEE TPAMI*, 2018.

[40] Z. Song and Q. Qiu. Learn to classify and count: A unified framework for object classification and counting. In *ICIGP*, 2018.

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CVPR*, 2015.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

[43] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *ICCV*, 2011.

[44] Q.-F. Wang, F. Yin, and C.-L. Liu. Handwritten chinese text recognition by integrating multiple contexts. *IEEE TPAMI*, 2012.

[45] S. Wang, L. Chen, L. Xu, W. Fan, J. Sun, and S. Naoi. Deep knowledge training and heterogeneous cnn for handwritten chinese text recognition. In *ICFHR*, 2016.

[46] C. Wigington, C. Tensmeyer, B. Davis, W. Barrett, B. Price, and S. Cohen. Start, follow, read: End-to-end full-page handwriting recognition. In *ECCV*, 2018.

[47] Y.-C. Wu, F. Yin, Z. Chen, and C.-L. Liu. Handwritten chinese text recognition using separable multi-dimensional recurrent neural network. In *ICDAR*, 2017.

[48] Y.-C. Wu, F. Yin, and C.-L. Liu. Improving handwritten chinese text recognition using neural network language models and convolutional neural network shape models. *Pattern Recognition*, 2017.

[49] Z. Xie, Z. Sun, L. Jin, H. Ni, and T. Lyons. Learning spatial-semantic context with fully convolutional recurrent network for online handwritten chinese text recognition. *TPAMI*, 2018.

[50] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[51] X. Yang, D. He, Z. Zhou, D. Kifer, and C. L. Giles. Learning to read irregular text with attention mechanisms. In *IJCAI*, 2017.

[52] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola. Stacked attention networks for image question answering. In *CVPR*, 2016.

[53] F. Yin, Q.-F. Wang, X.-Y. Zhang, and C.-L. Liu. ICDAR 2013 chinese handwriting recognition competition. *ICDAR*, 2013.

[54] F. Yin, Y.-C. Wu, X.-Y. Zhang, and C.-L. Liu. Scene text recognition with sliding convolutional character models. *CoRR*, abs/1709.01727, 2017.

[55] B. Zhang, Y. Gan, Y. Song, and B. Tang. Application of pronunciation knowledge on phoneme recognition by lstm neural network. In *ICPR*, 2016.

[56] J. Zhang, J. Du, and L. Dai. Track, attend and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition. *TMM*, 2018.

[57] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, and L. Dai. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 2017.