Honor Pledge :"I affirm that I will not give or receive any unauthorized help on this academic activity, and that all the work I submitted is my own."

Signature: Hang Zhang

An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition-BLOG POST

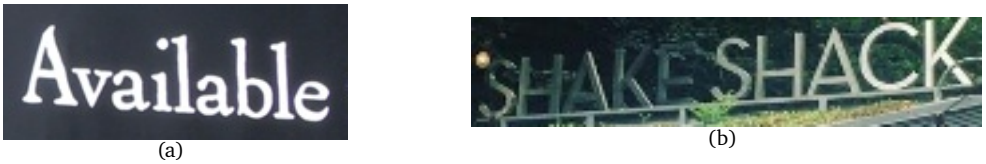
Hang Zhang
hz2447@nyu.edu

New York University— March 27, 2021

1 Introduction

Image-based text sequence recognition has been a long-standing OCR(Optical character recognition) research topic in computer vision. The work is to recognize the text sequence in a picture. Of course, there are many kinds of recognition tasks such document text recognition, scene text recognition. The paper focus on scene text recognition.

What is scene text? Briefly, scene text is text that appears in an image captured by a camera in an outdoor environment. Nowadays, smart phones with good cameras become ubiquitous, which makes recognition of scene text from camera captured images become much important than before. However, the text in scene images varies in shape, font, color and position sometimes even is vague due to illumination and camera out of focus, and therefore, the recognition of scene text is further complicated and harder compared with other text recognition tasks such as document text recognition.



(a) (b)

Figure 1.1: Examples of scene text

2 Challenge

2.1 Sequence recognition

In our real world, most of visual objects, such as scene text, handwriting and musical score, tend to occur in the form of sequence, not in isolation. Unlike general object recognition, recognizing such sequence-like objects often requires the system to predict a series of object labels, instead of a single label. Therefore, recognition of such objects is actually a sequence recognition problem.

2.2 Variable-length

Another unique property of sequence-like objects is that their lengths may vary drastically. For example, English words can either consist of 2 characters such as "OK" or 15 characters such as "congratulations". Consequently, the most popular deep models like DCNN(Deep Convolutional Neural Networks) cannot be directly applied to sequence prediction, since DCNN models often operate on inputs and outputs with fixed dimensions, and thus are incapable of producing a variable-length label sequence.

3 Previous Work

To solve the problem in **Challenge** section, some attempts have been made, however, all have bad trade-off.

3.1 Method 1

We can detect individual characters and then use DCNN model to recognize these detected characters respectively. It seems like a good work since this method makes DCNN applicable.
However, this method is really demanding since it needs us to design a strong character detector for detecting and cropping each character accurately out from image.

3.2 Method 2

Some other approaches treat scene text recognition as a image classification tasks. Assign a class label to each English word(90k totally) in dictionary. The model makes prediction for the input image and classify it to a label constructed from dictionary. This method's defect is very obvious. First, they are too many words in dictionary, which means they are too many classes predicted by neural network, in this case, to achieve good performance, the neural network will be trained with a huge number of parameters and a big-scale training dataset to learn the feature of every word. Second, it is difficult to be generalized to other language such as Japanese, Chinese, etc., because the number of combinations of such kinds of sequences can be even greater than 1 million.

4 Insightful Ideas

Since we have analyzed the problem, then we should propose an model with some components which can address it. Maybe we should refer to some methods from other fields.

4.1 Recurrent neural networks

Recurrent neural networks (RNN) models, widely used in Natural Language Processing, are mainly designed for handling sequences. One of its advantages is that the input and output can be variable-length. However, we always want our model to be trained in an **end-to-end** way.
End-to-end (E2E) learning refers to training a possibly complex learning system represented by a single model (a Deep Neural Network) that represents the complete target system, bypassing the intermediate layers usually present in traditional pipeline designs. The benefits are that it doesn't require programmers to have deep knowledge about every step of the pipeline, despite its complexity, and also, since all of the steps are merged into a whole model, we don't need to annotate the data manually for every step, which saves a great amount of labor resource. For recurrent neural networks, a preprocessing step that converts an input image into a sequence of image feature is needed, and most of those preprocessing steps are independent from RNN, which makes the model can not be trained in an end-to-end way.
So if we want to use RNN and train the model in an end-to-end way, we should do some further improvement.

4.2 Combination of RNN and DCNN

Don't forget DCNN is an architecture which is really good at extracting features from images! The only thing we need to do is to combine those two kinds of neural networks together. The DCNN serves as base, extracting features from text images and then transiting those features to RNN. Therefore, we achieve an end-to-end way to train our model.

5 The proposed Network Architecture

We can see the Network Architecture in Figure 5.1.
At the bottom of CRNN, DCNN extracts feature from the input image and output feature sequence. Above

is the RNN module, making prediction for each frame of the feature sequence outputted by the DCNN. The top is the transcription layer which translates the prediction made by RNN to final text sequence.

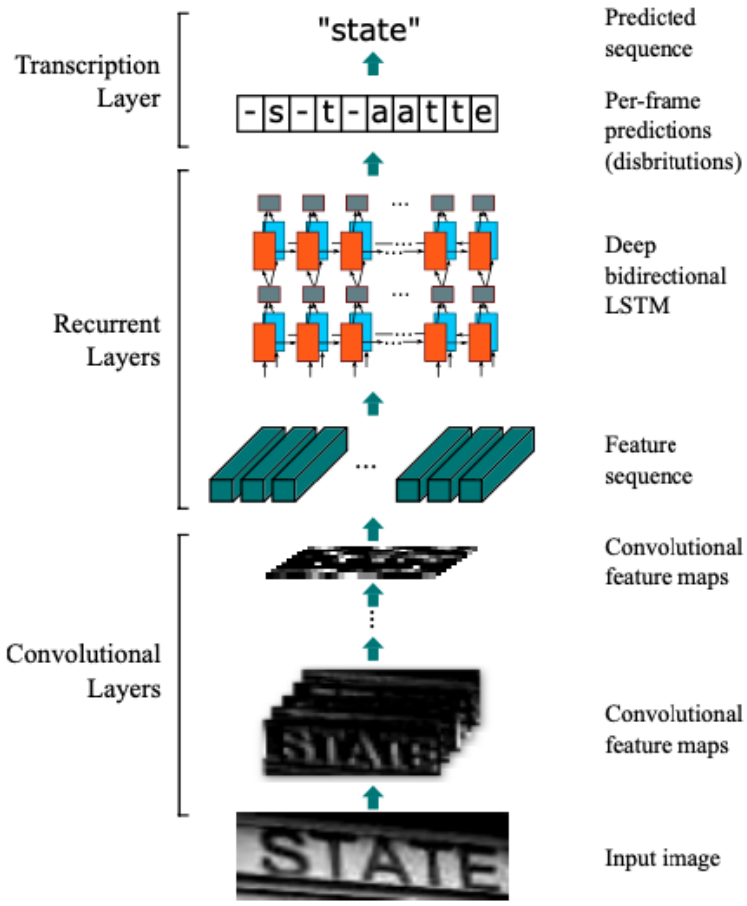


Figure 5.1: The network architecture. The architecture consists of three parts: 1) convolutional layers, which extract a feature sequence from the input image; 2) recurrent layers, which predicts a label distribution for each frame; 3) transcription layer, which translates the per-frame predictions into the final label sequence.

5.1 Convolutional Layer

5.1.1 The structure of our convolutional layer

Traditionally, DCNN has convolution layer, pooling layer, activation function, fully connected layer. Specifically, fully connected layers in a neural networks are those layers where all the inputs from one layer are connected to every activation unit of the next layer. In most popular machine learning models, the last few layers are full connected layers which compiles the data extracted by previous layers to form the final output.
However, we don't want to output the final outcome now, since we have Recurrent Layers and Transcription Layers on the top, so we just remove the fully connected layer.

5.1.2 How feature transit in DCNN layer

Before input to the DCNN layer, all the images need to be scaled to the same height. Then, a sequence of feature vectors is generated from left to the right on the feature map, which means the the i-th feature vector is just the concatenation of i-th column of all the maps. We know that convolution, pooling, and element-wise activation function operate on local regions, so they are translation invariant. Therefore,

every feature vector corresponds to a rectangle region of the original image, we call this region as Receptive Field, show in Figure 5.2.

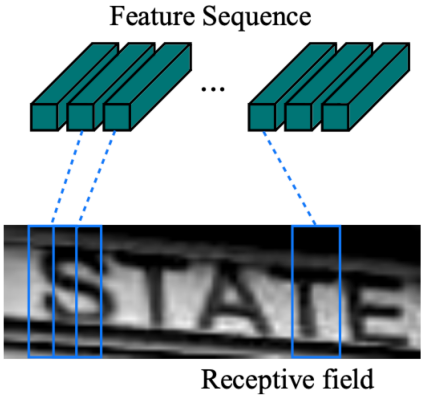


Figure 5.2: Receptive Field

5.2 Recurrent Layer

On the top of convolution layer is the recurrent layer, the recurrent neural network. Recurrent layer is to predict each feature vector x_i of the feature sequence $X = x_1, x_2, ..., x_T$ outputted by convolution layer.

5.2.1 Benefits

The benefits of recurrent neural network are tow-fold. **1)** It can handle sequence of arbitrary length, which offsets the deficiency of convolution layer. **2)** It is really good at capturing contextual information of a sequence. Since the words showed together in an image always have some contextual relation between each other, recurrent neural network possibly could improve accuracy of the final output.

5.2.2 Long Short Term Memory

A traditional RNN unit has a hidden layer between its input and output. At every time step t, given an input feature vector x_t, h_t is computed by both x_t and h_{t-1}, then make prediction based on h_t. Since the t-th hidden layer contains the information from previous hidden layer, the prediction is made based on context, not individually. However, the traditional RNN suffers from vanishing gradient problem, which limits the range of context it can remember.
To solve vanishing gradient problem, Long Short Term Memory unit is proposed. As illustrated in Figure 5.3(a). The **Cell** in the middle can be seen as Memory, which stores the contextual information. The Forget Gate is to clear the useless and redundant contextual information.
LSTM is directional, that is, it just records the past contextual information. It would be better for us to predict according to information from both directions. So we can combine two LSTMs, one forward and one backward, into a bidirectional LSTM. And also, to improve our model's performance, we can stack many bidirectional LSTMs.
RNN can back-propagates error differentials to its input, i.e. the convolutional layer, allowing us to jointly train the recurrent layers and the convolutional layers in an end-to-end way.

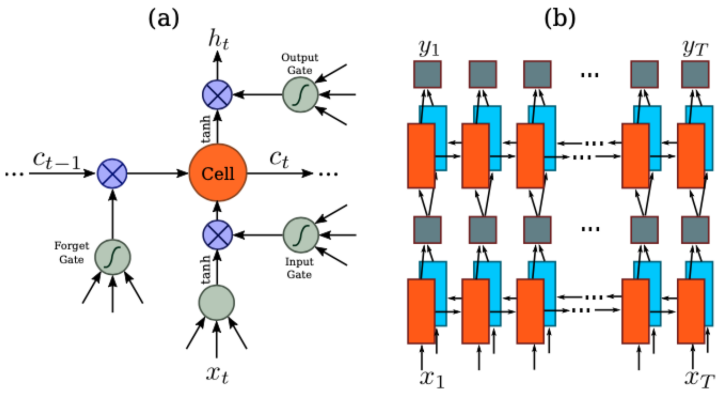


Figure 5.3: (a) LSTM unit. (b) The structure of deep bidirectional LSTM.

5.3 Transcription

The RNN layers make prediction for every frame and generate probability distribution for every frame. So what the transcription layer do is to transfer the probability distribution to text sequence. In practice, there are two modes of transcription, namely the lexicon-free and lexicon-based transcription. The former is that we make prediction for every frame just by choosing the character with highest probability. The latter is that we make prediction constraint to a lexicon, that is the text sequence must exist in a lexicon.

5.3.1 Probability of text sequence

How to translate the probability distribution to a text sequence. We have several problems to consider into.
1) First, there possibly exists several blank spaces between two words.
2) Second, one character may occupy several frames, such as "S" in Figure 5.2.
Let's think about **Speech Recognition** and find out what's the similarity between these two fields. When we speak out sentences, there's a short pause between two words or two sentences. The short pause is like our blank spaces between two words. And also, some words have prolonged sound, which occupies several time steps but others have short sound. It's similar that some characters are wide, which occupy several frames but others are thin.
So we use **Connectionist Temporal Classification(CTC)** method from **Speech Recognition**
The formulation of the conditional probability is briefly described as follows: The input is a sequence $Y = y_1, ..., y_T$, where T is the sequence length. Each y_i is a probability distribution over set L, L contains all characters of the task (e.g. all English characters) as well as a blank space label denoted by. A sequence to sequence mapping function $\mathcal{B}$ is defined on sequence $\pi \in L^T$, where T is the length. $\mathcal{B}$ maps π onto label sequence l by first removing the repeated labels and then remove blanks. For example, $\mathcal{B}$ map "–hh-e-l-ll-oo–" onto "hello", where "-" represents blank space. Then the conditional probability is defined as sum of probabilities of all π that are mapped $\mathcal{B}$ by onto l.

$$p(\mathbf{l} \mid \mathbf{y}) = \sum_{\pi:\mathcal{B}(\pi)=1} p(\boldsymbol{\pi} \mid \mathbf{y}), where\ p(\boldsymbol{\pi} \mid \mathbf{y}) = \prod_{t=1}^{T} y_{\pi_t}^t \tag{1}$$

$y_{\pi_t}^t$ is is the probability of having label π_t at time stamp t. This equation can be computed in a high-efficiency way-Dynamic Programming.

5.3.2 Lexicon-free transcription

Since we have no constraint by lexicon. We can just take the most probable π_t at each time stamp t to from π, and then map π to the final text sequence.

$$\mathbf{l}^* \approx \mathcal{B}\left(\arg\max_{\pi} p(\pi|\mathbf{y})\right) \tag{2}$$

5.3.3 Lexicon-based transcription

In lexicon-based model, each output is associated with a lexicon $\mathcal{D}$, that is we choose the label in lexicon with highest probability as the final output. The easiest method is to calculate the probability of every text sequence. However, the scale of a lexicon is very big, it would be very time-consuming to perform exhaustive search over the lexicon. The insightful idea is that we find the label sequences predicted via lexicon-free transcription is very close to ground-truth under the edit distance metric, which means we can limit our search in δ edit distance from the label sequences predicted via lexicon-free transcription. Define the nearest-neighbor candidates as $\mathcal{N}_\delta(\mathbf{l}')$ in δ and the sequence transcribed from y in lexicon-free mode as $\mathbf{l}'$.

$$\mathbf{l}^* = \arg \max_{1 \in \mathcal{N}_\delta(\mathbf{1}')} p(\mathbf{l} \mid \mathbf{y}) \tag{3}$$

The $\mathcal{N}_\delta(\mathbf{l}')$ can be found efficiently with BK-tree data structure, which could constructed offline and enable us to search in $O(\log |\mathcal{D}|)$ where $|\mathcal{D}|$ is the size of lexicon.

5.4 Network Training

Denote the training dataset by $\mathcal{X} = \{I_i, \mathbf{l}_i\}$, where I_i is the training image and l_i is the ground truth label sequence. The objective is to minimize the negative log-likelihood of conditional probability of ground truth:

$$\mathcal{O} = -\sum_{I_i, \mathbf{l}_i \in \mathcal{X}} \log p\left(\mathbf{l}_i \mid \mathbf{y}_i\right) \tag{4}$$

6 Experiments

6.1 Datasets

All the experiments are trained with the synthetic dataset (Synth) released by Jaderberg. Do test on four popular benchmarks for scene text recognition are used for performance evaluation, namely ICDAR 2003 (IC03), ICDAR 2013 (IC13), IIIT 5k-word (IIIT5k), and Street View Text (SVT).

6.2 Configurations

The first row is the top layer. 'k', 's' and 'p' stand for kernel size, stride and padding size respectively.

Type	Configurations
Transcription	-
Bidirectional-LSTM	#hidden units:256
Bidirectional-LSTM	#hidden units:256
Map-to-Sequence	-
Convolution	#maps:512, k:2 × 2, s:1, p:0
MaxPooling	Window:1 × 2, s:2
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
MaxPooling	Window:1 × 2, s:2
Convolution	#maps:256, k:3 × 3, s:1, p:1
Convolution	#maps:256, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:128, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:64, k:3 × 3, s:1, p:1
Input	W × 32 gray-scale image

Figure 6.1: Configurations

6.3 Comparative Evaluation

All the recognition accuracy on the above four public datasets, obtained by the proposed CRNN model and the recent state-of-the-arts techniques are shown in this table.

	IIIT5k			SVT		IC03				IC13
	50	**1k**	**None**	**50**	**None**	**50**	**Full**	**50k**	**None**	**None**
ABBYY [34]	24.3	-	-	35.0	-	56.0	55.0	-	-	-
Wang *et al.* [34]	-	-	-	57.0	-	76.0	62.0	-	-	-
Mishra *et al.* [28]	64.1	57.5	-	73.2	-	81.8	67.8	-	-	-
Wang *et al.* [35]	-	-	-	70.0	-	90.0	84.0	-	-	-
Goel *et al.* [13]	-	-	-	77.3	-	89.7	-	-	-	-
Bissacco *et al.* [8]	-	-	-	90.4	78.0	-	-	-	-	87.6
Alsharif and Pineau [6]	-	-	-	74.3	-	93.1	88.6	85.1	-	-
Almazán *et al.* [5]	91.2	82.1	-	89.2	-	-	-	-	-	-
Yao *et al.* [36]	80.2	69.3	-	75.9	-	88.5	80.3	-	-	-
Rodrguez-Serrano *et al.* [30]	76.1	57.4	-	70.0	-	-	-	-	-	-
Jaderberg *et al.* [23]	-	-	-	86.1	-	96.2	91.5	-	-	-
Su and Lu [33]	-	-	-	83.0	-	92.0	82.0	-	-	-
Gordo [14]	93.3	86.6	-	91.8	-	-	-	-	-	-
Jaderberg *et al.* [22]	97.1	92.7	-	95.4	80.7*	**98.7**	**98.6**	93.3	**93.1***	**90.8***
Jaderberg *et al.* [21]	95.5	89.6	-	93.2	71.7	97.8	97.0	93.4	89.6	81.8
CRNN	**97.6**	**94.4**	**78.2**	**96.4**	**80.8**	**98.7**	97.6	**95.5**	89.4	86.7

Figure 6.2: Comparative Evaluation

7 Weekness and Promising further research

7.1 Vertical text sequence

First, the CRNN model recognize the text from left to right, this is perfect for English text, since we never write English words vertically. However, for some other languages, sometimes they could be expressed vertically. Figure7.1 is an example of Spring couplets. In this case, CRNN model would fail since it does not consider vertical text sequence.

Figure 7.1: Spring Couplets. Vertical arrangement.

The CRNN model can not handle image like Spring couplets, which disable us to generalize it to Chinese or other language text sequence recognition.

7.2 Complex character structure

This is also a generalization problem. Even though CRNN performs well in English text recognition, it would fail in other language due to the complexity of character structure such as Chinese. We know that English character's structure is really simple, such as "a", "b" etc. However, the structure of Chinese character is really complex, lots of characters have left parts and right parts, such as "称" . The left part is "禾" , the right part is "尔" . If we just recognize it from left to right, the model may output "禾尔" but not expected "称" possibly. To solve this problem, I think we should refer to some stronger natural language processing module.

8 Conclusion

In this paper, we propose CRNN neural network to recognize scene text, the structure is Convolutional layer, Recurrent layer and Transcription layers, from bottom to top. Convolutional layer is to extract features from image, Recurrent layer receive the feature sequence from Convolutional layer, help to solve the problem that text sequence could be length-variable and bidirectional long short term memory recognize text according to context in the meantime. Using Connectionist Temporal Classification method, Transcription layer transfer the probability distribution outputted by Recurrent layer to final label sequence based on lexicon or not. In terms of the Transcription layer based on lexicon, we use BK-tree to improve efficiency of searching. The lexicon-free Transcription layer enable CRNN model to handle random strings, sentences which is not constrained to dictionary. Since these three layers are connected in a whole network architecture and are not independent, CRNN model can be trained in an end-to-end way.
In terms of the performance, we see it outperforms other recent state-of-art models in eight of ten test sets, which confirms the advantages of the proposed algorithm.
However, I think there's lots of room for improvement in generalization since it doesn't the arrangement and complex structure of other languages' characters, for instance, Chinese sentences could be showed in vertical arrangement and also Chinese characters' structure are more complex than English character. Referring to some NLP module may help us to solve this problem.

References

[1] Shi B, Bai X, Yao C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 39(11): 2298-2304.