END-TO-END CHINESE TEXT RECOGNITION

Jie Hu[⋆] *Tszhang Guo*[⋆] *Ji Cao*[†] *Changshui Zhang*[⋆]

[⋆] Department of Automation, Tsinghua University
[⋆]State Key Lab of Intelligent Technologies and Systems
[⋆]Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing, P.R.China
[†] Beijing SinoVoice Technology Co.,Ltd.

ABSTRACT

In this paper, we propose a new method for Chinese text recognition, which comprises two main contributions: First, we create a large Chinese text dataset, including 260 thousand images collected from business card and 390 thousand synthetic images generated by rendering engine. Second, we use these images to train a deep network to perform text recognition, which can recognize more than six thousand kinds of Chinese character accurately. Although our system is composed of different types of neural networks (CNN and LSTM), it is end-to-end trainable. Experiments demonstrate that our system achieve a high recognition accuracy in which synthetic data plays an important role.

Index Terms— text recognition, OCR, convolutional neural network, LSTM, CTC

1. INTRODUCTION

Optical character recognition (OCR) has a wide range of applications and has been researched for a long time, but it is still an important challenge in computer vision. Nowadays deep learning provides great potential for improvement. In this paper, we focus on text images (e.g. business cards, bank cards) recognition rather than natural scene images. While many recent researches have achieved great performance on English text recognition, their methods cannot be applied to Chinese text due to the significant difference between Chinese and English.

Early systems[1, 2, 3] that recognize text need presegmentation of text line before classifying individual characters. Yao et al.[4] extract mid-level features to identify characters by a random forest classifier. Deep neural networks have also been widely used, such as [5] use Convolution Neural Network (CNN) as character classifier. The famous PhotoOCR[6] performs a neural network classifier on single character and find the best combination of characters through beam search.

However, character based methods bring an extra step before recognition, which is computation inefficient and increases the chance of error. Some researchers introduced an alternative way, which performs recognition on the whole word instead of a single character. In [7], the author treats the recognition task as a classification problem across a huge predefined dictionary of 90k words. This kind of approach lacks flexibility and cannot recognize a rarely occurred word that is not included in the lexicon. Besides, this method cannot be used on Chinese text recognition directly, because Chinese text line do not have word segmentation. There are more than 6k characters in Chinese, which means a text line containing 10 words has about 6000^{10} combinations.

Recently, recurrent neural network (RNN) made great progress in speech recognition which is very similar to text recognition. Graves et.al[8] applied bidirectional Long-Short Term Memory (LSTM) with Connectionist Temporal Classification (CTC) on phonetic labelling task. Deep speech2[9] also significantly improved the performance of recognizing English and Mandarin Chinese speech by using RNN and CTC.

[4, 5, 6] need extra character segmentation and [7] require a pre-defined lexicon, which is not suitable for Chinese recognition. In this paper, generalized from [8, 9] on speech recognition, we propose a new method on text recognition which can recognize Chinese text in a segmentation-free and lexicon-free style.

Most commonly used text recognition dataset works for English recognition in natural scene images, e.g. SVT[12], ICDAR03[10] and ICDAR13[11]. To our knowledge, there is now no publicly available datasets for Chinese document recognition. Besides, the number of text lines in these datasets is only in thousands, which is far from training a deep neural network. However, we wish to build a practical OCR system that can recognize more than 6k Chinese characters. This require large amounts of samples to fully train our network. Therefore, we create a new Chinese document text dataset which will be introduced in detail in section2.

2. DATASET

We create a large Chinese text dataset, containing 260k samples collected from business card images and 390k synthetic

Authorized licensed use limited to: Jiangnan University. Downloaded on December 29,2023 at 05:11:08 UTC from IEEE Xplore. Restrictions apply.

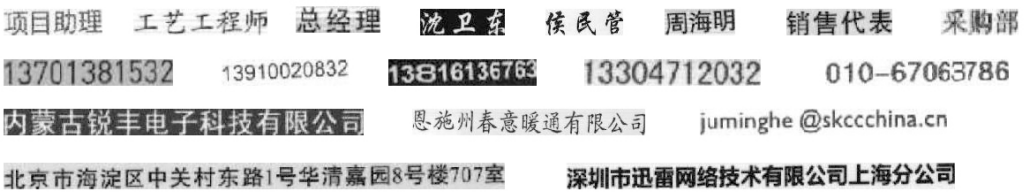

Fig. 1. Samples from business card images

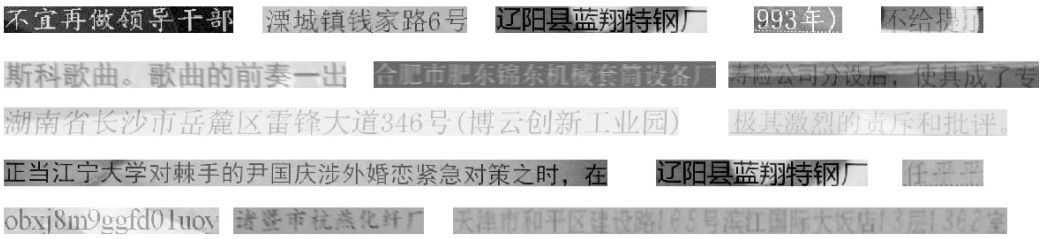

Fig. 2. Synthetic samples

samples. 90% of the samples are randomly chosen as training data, leaving the rest as test data. Chinese characters in business card images are not evenly distributed, which means some uncommon characters only appear a few times or even are not included in this dataset. Considering that there are more than 6 thousand commonly used characters, collecting sufficient images for every Chinese character is expensive and time costly. Following some previous success work on synthetic dataset[7], we build a synthetic image engine to generate about 390k images of text line without any manually annotation. The synthetic images make sure that there are sufficient samples for each character. In order to make our synthetic data similar to real-world images and be useful for training a practical network, the generation process includes several key points as mentioned in [7]:

1. Font rendering: We download 20 commonly used Chinese fonts which varies considerably. Every time a font is randomly selected from these 20 fonts.
2. Noise: Different types of noise, like Gaussian noise, random spot or line, motion blur and out-of-focus blur are added to the synthetic images.
3. Distortion: The engine performs scaling, rotation, projective distortion and skew transformation randomly, simulating the real-world images.
4. Background: These images are drawn with different background colors. Some texts are added with texture background or underlined. To simulate the light and shadow, some images are filled by gradient color.

All the images have a fix height of 32 pixels, while the width can be arbitrary length. Synthetic dataset contains 2.65 million characters in total, covering 6880 kinds of characters, including more than 6k Chinese characters, 52 English characters, 10 Arabic numerals and several punctuations. Figure 1 and Figure 2 illustrate some samples from real-world business card and some resulting generated data samples.

3. NETWORK ARCHITECTURE

As shown in Figure 3, our network consists of four main components: CNN for feature extraction, bidirectional LSTM for sequence processing, fully connected layer for predicting the probability of each character and CTC layer for loss computation.

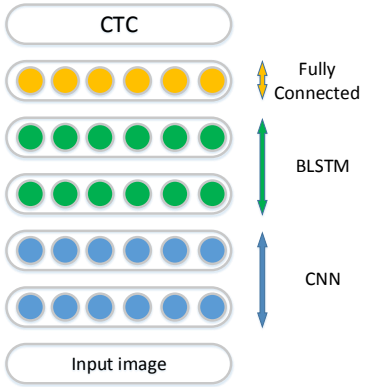

Fig. 3. Architecture of recognition system, consists of 4 main parts: CNN for feature extraction, bidirectional LSTM for sequence processing, fully connected layer for predicting the probability of each character and CTC layer for loss computation

Authorized licensed use limited to: Jiangnan University. Downloaded on December 29,2023 at 05:11:08 UTC from IEEE Xplore. Restrictions apply.

3.1. Feature Extraction

Our convolutional neural network follows standard CNN networks, composed of convolutional layers and pooling layers. ReLU[13] function $\sigma(x) = max(x, 0)$ is used as the activation function. The whole image is sent to CNN layer to extract features. Since all of our images have a fixed height of 32 pixels, all the feature maps have the same height but different width. As shown in Figure 4, the final feature map is sliced from left to right to form a feature sequence, which will be fed into next LSTM layer. In fact, each column of the feature map corresponds to a small rectangle region in the original image.

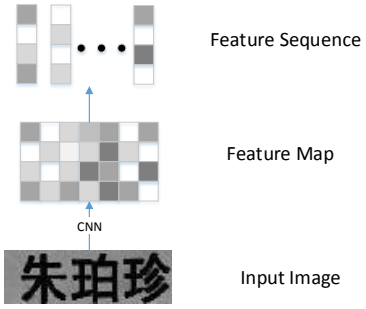

Fig. 4. CNN performs feature extraction on input images. The feature map is finally sliced into feature sequence.

3.2. Sequence Processing

RNN is placed after the CNN layer to process the input sequence. We denote the feature sequence of length T as $X = x_0, x_1, ..., x_{T-1}$. The goal of RNN layers is to convert the input sequence X into a output sequence $Y = y_0, y_1, ...y_{T-1}$. The input feature sequence is fed into the RNN network one by one. At each timestep t, the state of RNN units is not only determined by the input x_t, but also influenced by the state of last timestep: $h_t = f(h_{t-1}, x_t)$, where h_t stands for the unit state at timestep t, and the output y_t is computed from h_t. This architecture enables RNN to use contextual information when mapping between input and output sequences. While traditional RNN can process sequence in arbitrary length, it usually performs badly when the sequence is very long due to gradient vanishing problem. LSTM is another type of recurrent neural network which can address this problem. Since information from both forward and backward direction is useful, we use bidirectional LSTM to process input sequence, as shown in Figure 5.

3.3. CTC Loss

We apply a fully connected layer above LSTM given by:

$$y_t = f(Wh_t + b).$$

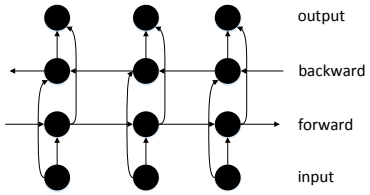

Fig. 5. Architecture of bidirectional LSTM

where h_t is the hidden unit state of last LSTM layer at time t, f denote activation function.

The output layer of the model is a softmax computing the probability distribution over 6881 characters (6880 normal characters as well as a 'blank' label denoted by '-'), i.e. the probability of outputting kth character at timestep t is computed as:

$$p(z_t^k) = \frac{exp(y_t^k)}{\sum_j exp(y_t^j)}.$$

Assuming the output probability distribution of each timestep to be independent, the model outputs a sequence π with probability calculated as:

$$p(\pi|x) = \prod_{t=1}^{T} z_t^{\pi_t}.$$

Following [8], we apply Connectionist Temporal Classification(CTC) to compute $p(\pi|x)$. A many-to-one function F is defined to remove repeated labels and blanks. For example, $F(a-ab-) = F(-aa--abb) = aab$. The probability of obtaining label l can be calculated by summing the probabilities of all the possible sequence mapped to it:

$$p(l|x) = \sum_{\pi:F(\pi)=l} p(\pi|x).$$

Although it is almost impossible to compute $p(l|x)$ directly, $p(l|x)$ can be calculated efficiently through dynamic-programming algorithm. We refer [8] for detail information.

Given $p(l|x)$, the loss function can be written as the negative log-likelihood of $p(l|x)$:

$$Loss = -log(p(l|x)).$$

This equation defined an objective function from input image to final cost, enabling our network to be end-to-end trained, which means we do not need to train the CNN and LSTM separately.

4. EXPERIMENTS

4.1. Implementation Details

Table 1 shows our network detail architecture. We implement the deep network (CNN and LSTM) and CTC layer using Theano[14] and open source code warp-ctc[1], respectively. We

[1]https://github.com/baidu-research/warp-ctc

Authorized licensed use limited to: Jiangnan University. Downloaded on December 29,2023 at 05:11:08 UTC from IEEE Xplore. Restrictions apply.

optimize using ADAM[15] with a learning rate 0.0002 and keep other parameters as default.

We use mini-batches of 64 images each. Because samples have different width, we apply zero-padding for short images. To accelerate training process and avoid redundant computation, each minibatch consists of images with similar width.

Experiments are conducted on GeForce GTX 1080. Training on both real-world data and synthetic data cost 40 hours to reach convergence.

layer	type	configuration
1	convolution	map: 16 k:2 $\times$ 2
2	convolution	map: 16 k:2 $\times$ 2
3	max pooling	ps:2 $\times$ 2
4	convolution	map: 64 k:2 $\times$ 2
5	convolution	map: 64 k:2 $\times$ 2
6	max pooling	ps: 2 $\times$ 2
7	convolution	map: 128 k:2 $\times$ 2
8	convolution	map: 128 k:2 $\times$ 2
9	max pooling	ps: 2 $\times$ 2
10	BLSTM	h: 800
11	BLSTM	h: 800
12	CTC	-

Table 1. Network configurations. map, k, ps and h stand for feature maps, kernel size, pooling size and hidden units

4.2. Results

We use line recognition accuracy (LRA) as evaluation protocol. A success recognition of a text line is defined as every character in the text is recognized correctly, and line recognition accuracy is defined as $LRA = |C|/|T|$ where C and T are the correctly recognized image number and total image number, respectively.

Table 2 summarizes our experiment result. Experiment 1 is carried out on real-world business card data and achieves a 96.27% LRA on real-world test data. Compared with experiment 4, the additional use of synthetic data results in a 2% accuracy raise, which prove our synthetic data to be useful. Interestingly, as shown in experiment 2, model trained only on synthetic data can also achieve 92.99% accuracy.

Experiment	training data	test data	LRA
1	real-world	real-world	96.27%
2	synthetic	real-world	92.99%
3	synthetic	synthetic	94.12%
4	real-world+synthetic	real-world	98.22%

Table 2. Comparison with networks trained on different datasets.

After trying various depths of LSTM layer, we find that the depth of LSTM greatly influences the recognition performance, as shown in Table 3. Note the drastic increase in LRA (+5.21%) as we increase the number of LSTM layers from 1 to 2. However, adding the third LSTM layers improves only marginally. We finally designate the LSTM layer depth as 2 to balance accuracy and computation speed.

depth	LRA
1	91.06%
2	96.27%
3	96.31%

Table 3. Comparison of networks with various depth of LSTM layer. All the models are trained and tested on real-world business card data.

Further, to establish that our model achieves high performance, we compared our result with two famous OCR software. Tesseract is a widely used open source OCR software, sponsored by Google since 2006. We test the performance of Tesseract v3.03 and ABBYY, a famous commercial software on real-world business card data. As shown in Table 4, our method significantly outperforms other two software. Take the difference between training data and test data into consideration, we list the result of the model trained only on synthetic data for a fair comparison.

method	LRA
Tesseract	28.51%
ABBYY	74.57%
our1	94.12%
our2	98.22%

Table 4. Comparison with other software. All the experiments are tested on real-world business card data. Our method 1 is trained only on synthetic data for a fair comparison. Our method 2 is trained on both synthetic data and real-world data.

5. CONCLUSION

In this paper, we proposed an end-to-end network to recognize Chinese text. Our system makes use of advantages of different kinds of neural networks (CNN, LSTM, fully connected layer) and can be jointly trained by applying CTC loss function. In order to fully train the network, we create a new Chinese text dataset which comprises 260k real-world images from business cards and 390k synthetic images. Experiments on our dataset have shown that our model achieves high recognition accuracy and the use of synthetic data brings additional improvement. Finally, we demonstrate that our system significantly outperforms other two widely used OCR software, including an open source software and a commercial software.

Authorized licensed use limited to: Jiangnan University. Downloaded on December 29,2023 at 05:11:08 UTC from IEEE Xplore. Restrictions apply.

6. REFERENCES

[1] Qiu-Feng Wang, Fei Yin, and Cheng-Lin Liu, "Handwritten chinese text recognition by integrating multiple contexts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 8, pp. 1469–1481, 2012.

[2] Sargur N Srihari, Xuanshen Yang, and Gregory R Ball, "Offline chinese handwriting recognition: an assessment of current technology," *Frontiers of Computer Science in China*, vol. 1, no. 2, pp. 137–155, 2007.

[3] Kai Wang, Boris Babenko, and Serge Belongie, "End-to-end scene text recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1457–1464.

[4] Cong Yao, Xiang Bai, Baoguang Shi, and Wenyu Liu, "Strokelets: A learned multi-scale representation for scene text recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 4042–4049.

[5] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman, "Deep features for text spotting," in *European conference on computer vision*. Springer, 2014, pp. 512–528.

[6] Alessandro Bissacco, Mark Cummins, Yuval Netzer, and Hartmut Neven, "Photoocr: Reading text in uncontrolled conditions," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 785–792.

[7] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016.

[8] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[9] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," *arXiv preprint arXiv:1512.02595*, 2015.

[10] Simon M Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, and Robert Young, "Icdar 2003 robust reading competitions.," in *ICDAR*. Citeseer, 2003, vol. 2003, p. 682.

[11] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere de las Heras, "Icdar 2013 robust reading competition," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, 2013, pp. 1484–1493.

[12] Kai Wang and Serge Belongie, "Word spotting in the wild," in *European Conference on Computer Vision*. Springer, 2010, pp. 591–604.

[13] Vinod Nair and Geoffrey E Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[14] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[15] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Authorized licensed use limited to: Jiangnan University. Downloaded on December 29,2023 at 05:11:08 UTC from IEEE Xplore. Restrictions apply.