SCENE TEXT RECOGNITION WITH DEEPER CONVOLUTIONAL NEURAL NETWORKS

Yuqi Zhang[1], *Wei Wang*[1], *Liang Wang*[1], *Liuan Wang*[2]

[1]Center for Research on Intelligent Perception and Computing,
National Laboratory of Pattern Recognition,Institute of Automation,
Chinese Academy of Sciences, Beijing 100190, China
[2]Fujitsu R&D Center Co., Ltd. China

zhangyuqi1991@gmail.com,{wangwei, wangliang}@nlpr.ia.ac.cn,liuan.wang@cn.fujitsu.com

ABSTRACT

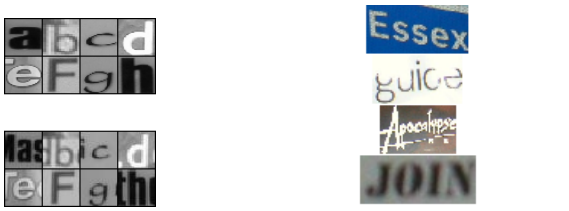
Scene text recognition plays an important role in many applications such as video indexing and house number localization in maps. Recently, some feature learning methods have been proposed to handle this problem, which often exploit deep architectures with no more than 5 layers and relatively large receptive fields. Meanwhile, to avoid model overfitting, they generally take advantage of large amount of additional data. Inspired by the great success of GoogleLeNet with a deeper network and VGG networks with smaller receptive fields in the ImageNet competition, in this paper, we adopt a much deeper network with up to 15 layers and smaller receptive fields (3×3) to learn better features for scene text recognition. Particularly, even without additional training data, our model can achieve better performance. Experiments on scene text datasets (ICDAR 2003, SVT, Chars74K) demonstrate that our method achieves the state-of-the-art performance on character classification and competitive performance on cropped word recognition.

Index Terms— scene text recognition, convolutional neural networks, receptive field

1. INTRODUCTION

With the great improvement in computer vision these years, the problems such as face detection and pedestrian detection have been well studied. Scene text recognition, however, seems to fall behind compared with other computer vision tasks. Text varies in fonts, texture, background and lighting, which makes the problem hard to solve. Commercial OCR products which are successful in scanned text recognition could not handle the problem well. Scene text recognition has many potential uses, such as helping guide robotic vehicles by reading text from the streets [1]. So great efforts should be taken on this task.

There are mainly two kinds of approaches in scene text recognition [2]: region-grouping-based and object-recognition-based approaches. Region-grouping-based methods group small components into larger components successively until all blocks are identified [3]. These methods are fast to compute and have been successful in scene text detection tasks [4, 5]. However such methods have difficulties in reading scene text images with noise or in low resolution. Object-recognition-based methods regard scene text recognition as common object detection and classification tasks. These methods [3, 6] extract features from the region and then use a classifier for recognition. Our proposed method in this paper belongs to object-recognition-based methods.



(a) Character examples (b) Cropped word examples

Fig. 1. Character and cropped word examples. (a) Character examples from ICDAR. Top: CH-tight. Bottom: CH-loose. (b) Cropped word examples from ICDAR.

Hand-crafted features designed by a lot of prior knowledge used to be prevalent. Recently feature learning methods have gained great success in various vision applications such as object detection, image classification and localization [7, 8, 9, 10]. Researches [6, 11, 12] have shown that feature learning, which just learns features from data, can be more effective in scene text tasks compared with hand-crafted features.

Wang et.al. [12] use SVM to classify features learned by unsupervised learning with a relatively large filter size (8×8 for the first layer). Their experiments also show that more learned features could lead to better performance. Jaderberg et.al. [13] make great improvements with convolutional neural networks (CNNs) by using much more training data and sharing features among case-insensitive, case-sensitive and bigram classifiers. Lee et.al. [2] learn features by choosing the most informative filters from many templates. The shape

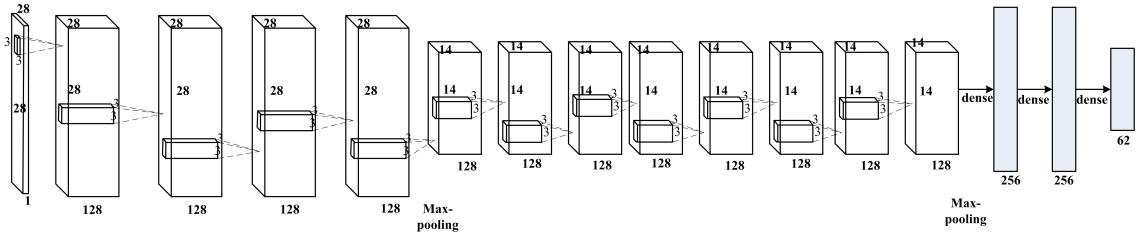


Fig. 2. An illustration of our deeper CNN architecture. There are 15 layers in total: 12 convolutional layers plus 3 fully-connected layers.

of the selected filters is not limited to square in their work, which is quite different from the methods above.

In this paper, we propose very deep convolutional neural networks to classify characters and further recognize cropped words as shown in Figure 1. The key to our method is the depth of the network and the smaller filter size. We demonstrate the state-of-the-art results on ICDAR03 [14], SVT-char [15] and Chars74K [16] in terms of scene character classification and competitive results on ICDAR03 and SVT [17] in terms of cropped word recognition. It should be noted that our models are trained only with the data from ICDAR03 for most of our experiments.

Section 2 describes the proposed CNNs architecture. Section 3 details the training and testing algorithms for character classification. Section 4 describes cropped word recognition briefly. Section 5 shows the results of our experiments prior to conclusion in Section 6

2. ARCHITECTURE

Figure 2 describes our deeper convolutional neural networks. There are 15 layers in total: 12 convolutional layers plus 3 fully-connected layers. We will discuss the characteristics of our architecture in the following.

2.1. Deeper Architecture

The winner in the ImageNet 2014 Competition indicates that depth of the network is of vital importance [9]. Different from ImageNet which has large-scale datasets in high resolution, we deploy deeper networks in relatively small-scale datasets in low resolution. Several deep architectures with 5,8,10 and 12 convolutional layers are experimented respectively while keeping other settings the same. Our experiment results illustrate that 12 convolutional layers achieve the best performance. Adding more layers seems to make no difference. See Section 5.1 for details.

2.2. Smaller Receptive Fields

Existing feature learning methods often extract features at the first layer with a larger filter size (for example 8×8). It is found in 2014 ImageNet Competition that using stacked convolutional layers with small receptive fields can make the decision function more discriminative and can be regarded as regularization on the larger filters [8]. We take stacked convolutional layers with receptive fields of 3×3 in our experiments.

2.3. Overall Architecture

As illustrated in Figure 2, there are 15 layers with weights to be learned: the first 12 convolutional layers plus the last 3 fully-connected layers. A 62-way (52 English case-sensitive characters + 10 number digits) softmax layer takes the output of the last fully-connected layer to give a distribution over the 62 class labels. We use cross-entropy loss function for this classification task.

All convolutional layers share the same receptive fields of 3×3. The convolution stride is fixed to 1 pixel. 1 pixel padding is used to keep the spatial resolution unchanged. Local Response Normalization (LRN) layers are adopted following the first and second convolutional layers. Max-pooling is used following the 4th and the 12th convolutional layers. The max-pooling stride is 2 and the size is 3×3. The neurons in the fully-connected layers are connected to all neurons in the previous layer. We apply dropout [18] to the first two fully-connected layers with 256 channels each to prevent overfitting. And we use Rectified Linear Unit (ReLU) [19] as the activation function.

3. TRAINING DEEPER NETWORKS

To make comparison with other methods, we adopt the 28×28 gray scale image as input. Data augmentation is performed to avoid overfitting. Each image is resized to 32×32, and then randomly scaled and rotated. Next we randomly crop 28×28 patches from the 32×32 gray image. Although the patches are highly correlated, data augmentation does benefit the performance.

As shown in Table 1, we propose several deep architectures with different depths: base model with 5 convolutional layers, model A with 8 convolutional layers, model B with 10

base model	model A	model B	model C	model D
4×conv	4×conv	4×conv	4×conv	4×conv
pooling	pooling	pooling	pooling	pooling
1×conv	*4×conv*	*6×conv*	*8×conv*	*8×conv*
pooling	pooling	pooling	pooling	pooling
FC-256	FC-256	FC-256	FC-256	*FC-4096*
FC-256	FC-256	FC-256	FC-256	*FC-4096*
FC-62	FC-62	FC-62	FC-62	FC-62
softmax	softmax	softmax	softmax	softmax

Table 1. Model configurations. Conv represents convolutional layers with a filter size 3×3, stride 1 and 128 channels. Pooling represents overlapping max-pooling layers with size 3×3 and stride 2. LRN(omitted in the table) is used after the first and second convolutional layers.

convolutional layers and model C with 12 convolutional layers. We experiment with these models for two reasons. First, we can explore how the depth effects the performance. Second, as mentioned in [8], very deep convolutional networks are hard to train at the beginning stage due to a large number of rectification non-linearities. So we first train a shallow network with five convolutional layers as a base model. Then we gradually add convolutional layers at the top. We initialize the weights and biases of the deeper network with the shallower one. The base model and the newly-added layers are initialized from a zero-mean Gaussian distribution with standard deviation 0.01. Different from [8] which only initializes some of the layers, we initialize all possible layers from the shallower network. In this way, we could train very fast since the initialized layers have been well learned. All layers share the same learning rate which means the initialized layers also need to learn as a whole.

To optimize the networks, we use stochastic gradient descent (SGD) with mini-batch 128, momentum 0.9 and weight decay 0.0005. The initial learning rate is set to 0.01 for weights and 0.02 for biases. We decrease the weights learning rate by a factor of 0.16 every 100 epochs and the biases learning rate by a factor of 0.1 every 200 epochs until the network converges at around 400 epochs.

For character classification, we crop 9 patches (top-left, top-middle, top-right, middle-left, middle, middle-right, bottom-left, bottom-middle, bottom-right) from the 32×32 images and average their predictions as the final results.

4. CROPPED WORD RECOGNITION

In this Section, our goal is to recognize the words which have been cropped out accurately with the given bounding boxes. We follow the postprocessing methods of [12] to make use of our learned character classifier. For a given cropped word image, we first resize it so that the height is 32 and then compute the 62-way probability for each position in a sliding window manner. It should be noted that we do not perform the multi-view test but simply take the center 28×28 patch of each 32×32 window. By padding properly, we get a 62×N score matrix M, where N is the width of the resized image. Non-maximum suppression is performed at matrix M to select the columns of candidate characters. Then we go through the whole lexicon to find the word with the highest matching score. See [12] for more details.

5. EXPERIMENTS

5.1. Character Classification

We evaluate our character classifier on ICDAR03 [14], SVT-char [15] and Chars74K-15 [16]. Note that there are several versions of ICDAR03 character datasets due to different cropping methods. The original character classification dataset from ICDAR03 [14] crops out characters tightly and has a testing set of 5379 images. We call this CH-tight. Other works [11, 12, 13] crop out characters loosely, which add some other characters together with the corresponding ground truth. They also ignore some obscure characters in the testing set, resulting in a testing set of 5198 images. We call this CH-loose. Our models are tested on both CH-tight and CH-loose. Chars74K-15 contains 15 examples per character class: 930 images for training and 930 images for testing in total. SVT-char has only 52 alphabet classes taking no consideration into numbers. It consists of 3796 characters from SVT-WORD [17]. We crop out characters loosely according to SVT-char bounding boxes. In the following experiments, we implement our methods with cuda-convnet2 [20].

Table 2 lists the character classification results on CH-tight and Chars74K, which share the characteristic of severe distortions after resizing. We train models with different depths on CH-tight to illustrate the importance of network depth. Models on Chars74K-15 are fine-tuned from the corresponding models on CH-tight. It can be seen that base model has already shown better performance compared with other methods [2, 17, 21]. Model A further improves the performance compared with base model, which means the 3 more convolutional layers in the middle does help learn better features. Considering model B and model C, we find that CH-tight benefits more from deeper networks while Chars74K-15 does not. This is possibly because model A has already learned necessary features for this relatively small dataset Chars74K-15. Our model C achieves the best performance with an accuracy of 84.9% on CH-tight and 78.8% on Chars74K-15.

Table 3 lists classification accuracy on CH-loose and SVT-char. We fine-tune the pre-trained model C with the CH-loose dataset and achieve an accuracy of 87.2%. This is higher than [13], which uses about 0.1 million images as training set and 2.6 million parameters for case-insensitive

Method	CH-tight	Chars74K-15
Lee et.al. [2]	79.0	64.0
Native-Ferns [17]	64.0	54.0
Synth+Ferns [17]	52.0	47.0
GHOG+SVM [21]	76.0	62.0
LHOG+SVM [21]	75.0	58.0
base model	82.2	75.5
model A	84.3	78.7
model B	84.6	78.7
model C	*84.9*	*78.8*

Table 2. Character classification accuracy on CH-tight and Chars74K(%).

Method	CH-loose	SVT-char
Wang et.al [12]	83.9	-
Jaderberg et.al [13]	86.8	80.3
model C	*87.2*	77.2
model D	*90.1*	*85.1*

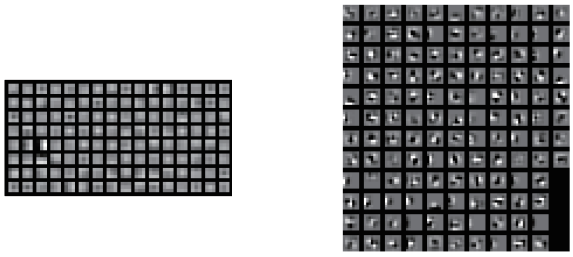
Table 3. Character classification accuracy on CH-loose and SVT-char(%).

classifier. However, we use fewer images (6113 images in CH-tight for pre-training plus 5980 images in CH-loose for fine-tuning) and 1.7 million parameters in model C, which proves the importance of better feature representation. It should be noted that for SVT-char, we directly test with model C from CH-loose because SVT-char has no training set for further fine-tuning. We achieve a lower accuracy of 77.2% (case-insensitive), which illustrates the importance of fine-tuning.

To make the best use of big data, we also train a larger model called model D with the 0.1 million training data from [13]. Model D, which initializes its convolutional layers from model C, expands the fully-connected layers with 4096 channels. We achieve 90.1% on CH-loose and 85.1% (case-insensitive) on SVT-char with model D. In the following, we use model D for cropped word recognition tasks.

Method	ICDAR (Full)	ICDAR (50)	SVT (50)
SYNTH+PLEX [17]	62.0	76.0	57.0
Lee et.al. [2]	76.0	88.0	80.0
Jaderberg et.al. [13]	91.5	96.2	86.1
Wang et.al. [12]	84.0	90.0	70.0
model D	*92.3*	*96.4*	*88.4*

Table 4. Cropped word recognition accuracy on ICDAR and SVT(%).



(a) First convolutional layer (b) Third convolutional layer

Fig. 3. Visualization of the filters from model C.

Figure 3 illustrates the learned convolutional filters from model C. Using Deconvnet described in [10], we could visualize in pixel space what causes the maximum activation in the third convolutional layer.

5.2. Cropped Word Recognition

We follow the same evaluation methods as [17], which ignore words shorter than 2 characters and provide a lexicon for each word. In Table 4, ICDAR-Full represents the lexicon with all the words in the testing set while ICDAR-50 means the lexicon with true tags plus additional 50 random distractor words. Similarly we define SVT-50 for the SVT cropped word dataset.

Table 4 lists the results of cropped word recognition. We achieve 92.3% for ICDAR-Full and 96.4% for ICDAR-50 with model D. The excellent character classifier makes great contribution to the performance. Although [22] has obtained better results, they take advantage of millions of synthetic word images as additional training data. We do not list it here to make fair comparisons with other methods which only use character images.

6. CONCLUSION

In this paper, we have used very deep convolutional neural networks with very small receptive fields for scene character classification and cropped word recognition. We demonstrate that the proposed network with fewer parameters can outperform the existing models without additional training data in the character classification task. We also used the powerful character classifier in the cropped word recognition and achieved competitive performance with very simple post-processings.

7. ACKNOWLEDGMENTS

This work is jointly supported by National Natural Science Foundation of China (61175003, 61135002, 61202328), National Basic Research Program of China (2012CB316300). We would also thank NVIDIA for donating GPU K40.

8. REFERENCES

[1] I. Posner, P. Corke, and P. Newman, "Using text-spotting to query the world," in *Proc. Conf. Intelligent Robots and Systems*, 2010, pp. 3181–3186.

[2] C. Lee, A. Bhardwaj, W. Di, V. Jagadeesh, and R. Piramuthu, "Region-based discriminative feature pooling for scene text recognition," in *Proc. IEEE Intl Conf. Computer Vision and Pattern Recognition*, 2014, pp. 4050–4057.

[3] K. Jung, "Neural network-based text location in color images," *Pattern Recognition Letters*, vol. 22, no. 14, pp. 1503–1515, 2001.

[4] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *Proc. IEEE Intl Conf. Computer Vision and Pattern Recognition*, 2010, pp. 2963–2970.

[5] X.C. Yin, X. Yin, K. Huang, and H. Hao., "Robust text detection in natural scene images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 970–983, 2014.

[6] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Intl Conf. Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[7] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *arXiv preprint arXiv:1409.4842*, 2014.

[10] M. Zeiler D and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV 2014*, 2014, pp. 818–833.

[11] A. Coates, B. Carpenter, C. Case, S. Satheesh., B. Suresh, T.Wang, D. J. Wu, and Andrew Y. Ng, "Text detection and character recognition in scene images with unsupervised feature learning," in *Proc. IEEE Intl Conf. Document Analysis and Recognition*, 2011, pp. 440–445.

[12] T. Wang, D.J. Wu, A. Coates, and Andrew Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Proc.IEEE Intl Conf. Pattern Recognition*, 2012, pp. 3304–3308.

[13] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," in *ECCV*, 2014, pp. 512–528.

[14] S. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R.Young, "Icdar 2003 robust reading competitions," in *Proc. IEEE Intl Conf. Document Analysis and Recognition*, 2003, pp. 682–682.

[15] A. Mishra, K. Alahari, and C.V. Jawahar, "Top-down and bottom-up cues for scene text recognition," in *Proc. IEEE Intl Conf. Computer Vision and Pattern Recognition*, 2012, pp. 2687–2694.

[16] T.E.de Campos, B.R. Babu, and M. Varma, "Character recognition in natural images," in *Proc.Conf. Computer Vision Theory and Applications*, 2009.

[17] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *Proc. IEEE Intl Conf.Computer Vision*, 2011, pp. 1457–1464.

[18] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th International Conference on Machine Learning*, 2010, pp. 807–814.

[20] A. Krizhevsky, "cuda-convnet2," https://code.google.com/p/cuda-convnet2/.

[21] C. Yi, X. Yang, and Y. Tian, "Feature representations for scene text character recognition: A comparative study," in *Proc. IEEE Intl Conf. Document Analysis and Recognition*, 2013, pp. 907–911.

[22] 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.