Convolutional Neural Networks for Handwritten Chinese Character Recognition

Chao Li
Simon Fraser University
cla315@sfu.ca

Junsheng Pan
Simon Fraser University
johnsonp@sfu.ca

Jiameng Yang
Simon Fraser University
jiamengy@sfu.ca

Abstract

In this project the performance of deep convolutional neural networks, with different configurations and training parameters, are investigated through their application towards recognizing handwritten Chinese characters from a 300-class dataset. We offered a simplified method of preprocessing the dataset obtained from CASIA offline Chinese handwriting database. The experimental results are consistent with other researchers' work, showing that networks with more convolutional layers and more filters numbers give higher test accuracy. Visualization of the filter kernels and layer outputs are also provided in this report.

1 Introduction

Deep convolutional neural networks (CNNs) have been extensively used for vision recognition in recent years. One of the state-of-the-art network configurations is presented by VGGNet [1] which yields a top-5 test error of as low as 6.8% in 1000-class ImageNet Challenge 2014. There has been a lot of research [2,3,4] on recognizing handwritten digits and English alphabets using similar network architecture designs, which has demonstrated very good performance. However, less work has been done on recognizing handwritten Chinese characters due the significantly increased complexity of this classification problem.

Recognizing Chinese characters is more challenging in comparison to recognizing digits and English alphabets in that, firstly it has more categories, for example there are over 50,000 Chinese characters with 3,000 characters covering 99% for daily use, while there are conceivably less than 100 English characters, including lowercase, uppercase, diacritical marked, and other special characters. Secondly, compared to English, there are more strokes in Chinese characters, because of which personal handwriting style can vary quite differently. In addition, much like in English, the use of cursive writing (e.g. joining-up or dropping strokes as shown in Figure 1 for character 统) make the task of recognition even more challenging.

Figure 1: Different data examples in the same category

In this project, we explored the performance of CNNs with various configurations by feeding in images of scanned handwritten characters which were preprocessed using our own methodology. The rest of the report is organized as follows. We will first list related work

done on this topic, from which our work is built upon. Secondly, we will present our unique method of preprocessing the dataset. Afterwards, we will give details on implementations, network configurations and training parameters. Finally, experimental results will be presented and discussed.

2 Related work

Many researches have focused on recognition of handwritten characters. For example, the classic CNN architecture, LeNet has achieved above 99% test accuracy on MINST handwriting database [2]. C. I. Patel, R. Patel, and P. Patel [4] applied the backpropagation neural network on an English character dataset, which consists of A-Z typed characters of different size and type, and achieved a test accuracy of 98% in their best model.

However, for Chinese characters, the strokes and structures are more complicated compared to the handwriting of numbers or English characters, which requires more image preprocessing procedures in the beginning to achieve better performance. X. Chen [5] proposed the use of Sobel operators to extract gradient features. Then each gradient vector was decomposed into two nearest directions out of L direction defined with equal intervals. Y. Zhang [6] resized the image values to a size of 56×56 pixels and then added white margins to make it 64×64 pixels. He also rescaled the image values to maximize the contrast and applied normalization for the image examples. Both of them used several self-defined CNN models to train and test the images after preprocessing.

3 Data

3.1 Dataset

The dataset for this project is downloaded from the CASIA office Chinese handwriting database, named CASIA-HWDB1.0 [7]. It consists of 1,609,1369 scanned grey-scale images of 3,866 Chinese characters handwritten by 420 different people. Due to the limitation of computation resources and test time, we used a subset of 99,977 samples from 300 randomly chosen classes, with each containing approximately 330 samples. Each sample from each class is contributed by a distinct writer. This subset is split into two parts: one part is the training set with approximately 260 samples in each class, the other part is the test set with which the rest of samples (approximately 70) in each class will be used to test the performance of the CNNs. (Please check the database name and number and other info)

Figure 2: An example image after each data preprocessing steps. From left to right: input image, image with paddings, grayscale image with binary pixel values.

3.2 Preprocessing

Each raw data file contains labels and gray-scale matrices for a collection of samples. Therefore, the following four steps were taken to process the raw data into input data for our networks. First, the raw data was parsed and converted to isolated .bmp gray-scale images for each sample. Second, each image was centered and padded into a square matching its longest side. Third, the images were resized into a 64×64 pixel image. We chose this size as the default input size because previous studies [5,6] have found that the character in an image this size retains its structure and produces high accuracy results while minimizing file size. Fourth, the unique part of preprocessing images in our work takes is that, instead of

performing image standardization of the gray-scale image, we convert the resized image into a binary format image with background pixels labeled as 255 (white) and character strokes' pixels labeled as 0 (black). The change in the image of character 佛 due to preprocessing steps 2-4 is demonstrated by Figure 2.

4 Network configurations

During training, each binary input image is fed into the network. Naïvely, we first attempted to implement transfer learning into our character recognition by modifying only the top layers of a well-trained VGGNet [1]. However, as our data input differs significantly from the ImageNet dataset, this fine-tuned modified network with originally trained weights was not able to learn anything from our input data after 10,000 epochs. Then, we reconstructed our network using a 9-layer CNN with a stack of 3 convolutional layers; this shallower network has been proved to give good accuracy in two previous studies [5,6] conducted on the recognition of the same database.

Moreover, the same parameter settings have also been applied to our networks. In each convolutional layer, a 3x3 filter window is used with stride size of 1 and spatial padding of 1 pixel such that the spatial resolution is preserved after the convolution. Note that the rectification non-linearity (ReLU[8]) is applied in each hidden layer as the activation function.

Following each convolution is a max-pooling layer with a 2x2 pixel window and stride size of 2. The output from the last max-pooling layer is then flattened and passed through two fully connected layers, each having 1024 channels. The final layer is a fully connected layer with 300 channels combined with the soft-max activation.

By modifying the CNNs architectures based on the above default settings, we investigated the effects of the depth of the network, as well as the number of filters in the convolutional layers on the test accuracy.

The most straight forward and intuitive technique of visualizing a CNN is to visualize the training weights and the layer outputs after each convolution. For ReLU networks, the layer output usually looks filtered but relatively unchanged at the beginning of the network, then becomes sparse and localized as the image passes through a stack of layers. The visualization of weights is in general more interpretable on the first convolutional layer as it applies directly on the raw pixel of input image. Nice and smooth filters without any noisy patterns are desirable as this is an indicator of a well-trained network.

5 Experiment

5.1 Implementation

The CASIS-HWDB1.0 character database file is encoded in a custom binary format called "gnt". The file archived over 1.5 million images along with their actual Chinese character encoded in GB2312 character set (Table 1). While the "gnt" format is not supported by common image viewing tools, we implemented our custom tool in C++ for extracting arbitrary character samples into a separated bmp image file, along with its actual Chinese character.

All the images extracted from the character file have the image resolution of the writing's bounding box, which it usually not in a square shape (e.g. 10×50). Resizing non-squared images directly may create distortion on the character image and makes it difficult to be recognized. To solve this problem, we added extra while-pixel paddings to each character images and made them into a squared shape (e.g. 64×64). We also converted all bmp colorful image into bmp grayscale images since the writing of a character is color invariant.

We used Keras as the underlying deep learning framework to build and test our CNN models, with TensorFlow as the backend. We performed our training on a GPU-accelerated PC.

Table 1: Format of “gnt” character data file [7]

Item	Type	Length	Instance	Comment
Sample size	unsigned int	4B		Number of bytes for one sample (byte count to next sample)
Tag code (GB2312)	char	2B	"啊"=0xb0a1 Stored as 0xa1b0	
Width	unsigned short	2B		Number of pixels in a row
Height	unsigned short	2B		Number of rows
Bitmap	unsigned char	Width×Height bytes		Stored row by row

5.2 Training

Training is carried out on the training set of the 300-class subset by optimizing the soft-max objective using mini-batch gradient descent with momentum. The default batch size is set to 128 due to the limitation of our GPU power. The momentum is set to constant 0.9 for all networks. The dropout rate for all fully connected layer is set to 0.5. Note, no validation set is split and tested during training.

The initialization of networks is different depending on the depth of the network. For networks that are not very deep, i.e. with no more than three convolutional layers, their weights are randomly initialized. However, this could result in very slowly learning or not learning at all for very deep networks at the end of 10,000 epochs. The solution was to initialize the weights by loading trained weights from a “shallower” network which had some layers in common with the new deeper one.

5.3 Testing

Testing is conducted on the test set of the 300-class subset. For each network architecture, we tested 10,000 randomly selected samples from the test set for top-5 accuracy which represents the percentage of the correctly classified samples that are present in the top 5 predicted classes.

6 Results and discussion

Configuration parameters of CNNs with various architecture are described as in Table 2. The top-5 accuracy results produced by these CNNs are presented in Table 3. The network with the most impressive performance is the one that consists the most convolutional layers, N7. By comparing the results of N3 and N4, we found that for similar network architecture, a larger number of filters in each layer results in better overall accuracy. The reason could be that more filters would extract more features from the images. By comparing N2 and N4, we can conclude that replacing a fully-connected layer with a convolutional layer can offer us larger performance gain. By comparing N4, N5, N6 and N7, we confirmed that a deeper network with more convolution layers have better performance.

The visualization of filters and layer output of our default model is very intuitive as shown in Figure 3-5 where each box corresponds to one filter in a layer. The Chinese character for Buddha, 佛, was chosen for demonstration. The filters of three convolutional layers of our default network is show in Figure 3. Notice that the weights of all layers are very smooth and

Table 2: CNN configurations

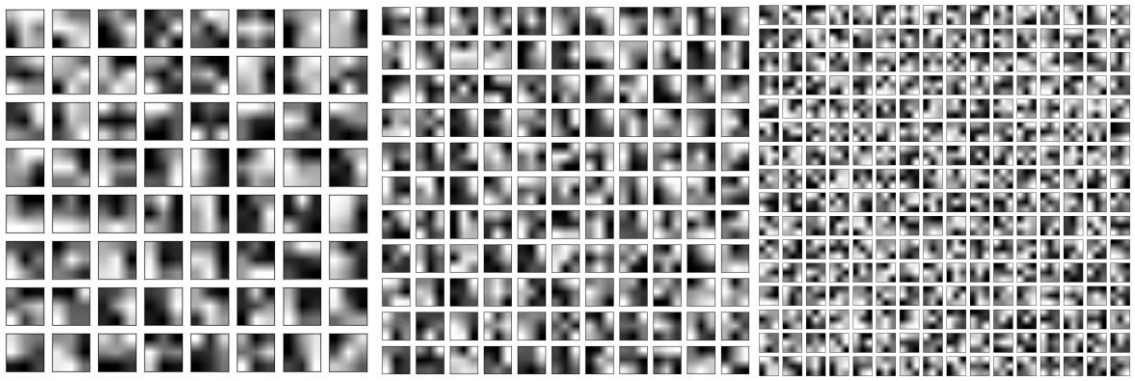
<table>
<tr><th>N2</th><th>N3</th><th>N4</th><th>N5</th><th>N6</th><th>N7</th></tr>
<tr><td colspan="6">Input data (64x64 binary image)</td></tr>
<tr><td>conv3-64</td><td>conv3-32</td><td>conv3-64</td><td>conv3-64</td><td>conv3-64</td><td>conv3-64</td></tr>
<tr><td colspan="6">maxpool</td></tr>
<tr><td>conv3-128</td><td>conv3-64</td><td>conv3-128</td><td>conv3-128
conv3-128</td><td>conv3-128</td><td>conv3-128</td></tr>
<tr><td colspan="6">maxpool</td></tr>
<tr><td>conv3-256
conv3-256</td><td>conv3-128</td><td>conv3-256</td><td>conv3-256
conv3-256</td><td>conv3-256</td><td>conv3-256
conv3-256</td></tr>
<tr><td colspan="6">maxpool</td></tr>
<tr><td colspan="4" rowspan="2"></td><td>conv3-512</td><td>conv3-512
conv3-512</td></tr>
<tr><td colspan="2">maxpool</td></tr>
<tr><td></td><td colspan="5">FC-1024</td></tr>
<tr><td colspan="6">FC-1024</td></tr>
<tr><td colspan="6">softmax</td></tr>
</table>

absent of noisy patterns, indicating a nicely converged network. Visualization of the layer outputs (Figure 4,5) show that the activations in this ReLU network are dense in the first 2 layers as the character 佛 can be easily recognized. However, after the 3rd convolution, it becomes so localized and sparse that we mostly see black, (activation values correspond to zero), in most boxes. In contract, the less localized visualization of the 3rd max-pooling layer demonstrates its ability of effectively extracting information from the previous convolutional layer.

Converting the character images into binary format in our data preprocessing means that we intentionally abandoned some information about a writer's writing style stored in a full gray-scaled image. Take an example of cursive writing, the joined-up strokes usually appear thinner than standard strokes, which in a scanned image would be represented as a lighter-color stroke. With a binary image input, our network treats cursive flourishes as an equal part of the standard stroke, the network is not able to compensate for the loss of information and thus yields lower accuracy in comparison with previous researchers' work [5,6].

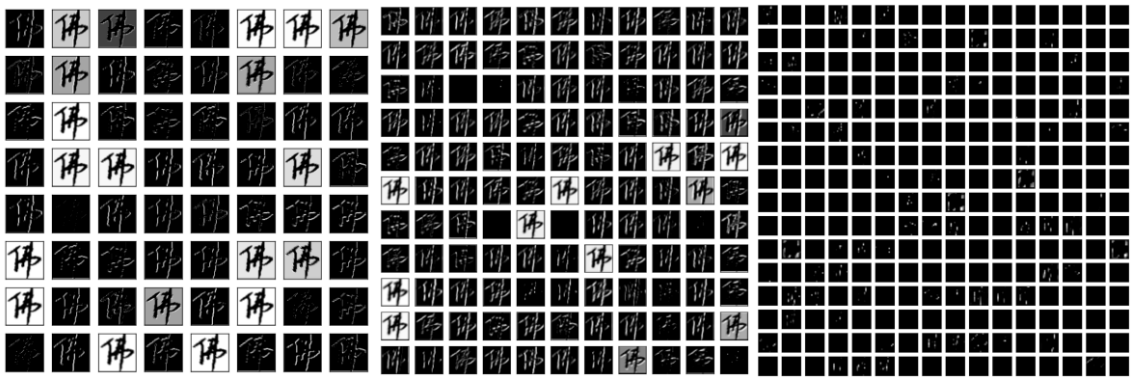
Table 2: Classification results on the 300-class dataset

Model	Top-5 Accuracy (%)
N2	95.29
N3	80.38
N4	93.57
N5	94.08
N6	95.49
N7	96.08



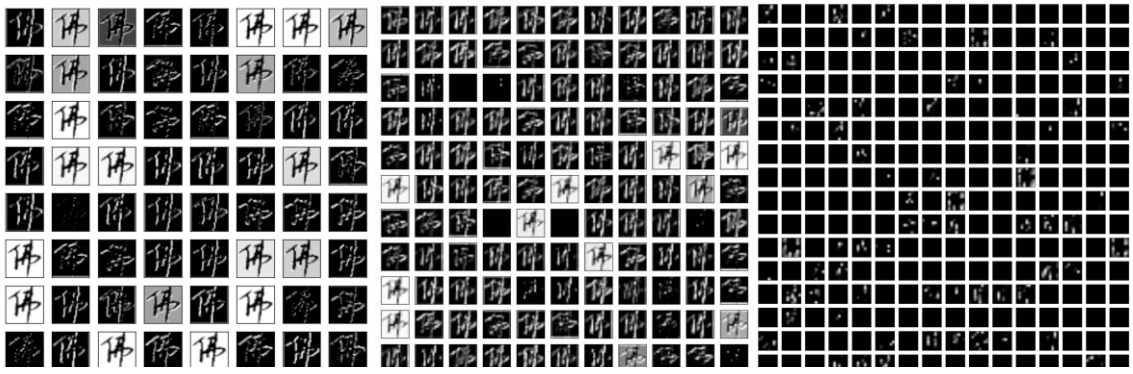
conv1 conv2 conv3

Figure 3: Visualization of weights in convolutional layers of N4



conv1 conv2 conv3

Figure 4: Visualization of outputs from convolutional layers of N4



pool1 pool2 pool3

Figure 5: Visualization of outputs from maxpooling layers of N4

7 Conclusion and future work

In this project, we provided a simplified method to preprocess handwritten character images. Though some information about cursive writing is missing in our preprocessed images, the deep convolutional neural network is still able to show relatively impressive performance on a 300-class subset in comparison with previous work done on subset from the same database. Our test results produced by various carefully designed networks aligned with the dependence of test accuracy on the depth and filter number of CNN which has been validated by many studies. In conclusion, 1) accuracy increases with the depth of the network; 2) accuracy increases with the number of filter; 3) adding a convolutional layer is more effective in increasing the accuracy than adding a fully-connected layer. Moreover, visualization of filters and layer outputs is a very intuitive way of understand the CNN.

No data augmentation was applied, such as rotating or shifting, because the handwritten characters were centered in each preprocessed input image. However, this could be an area of future research.

Acknowledgments

We would like to acknowledge the assistance from our TAs, Mengyao Zhai, Mehran Khodabandel, and Jiawei He, for helping us understand the course materials and aiding us throughout our project.

References

[1] Simonyan, K. & Zisserman, A. (2015) Very deep convolutional networks for large-scale image recognition. *arXiv Preprint arXiv:* 1409.1556.

[2] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE,* 86(11):2278-2324.

[3] Coates, A., Ng, A, et al. (2011) Text detection and character recognition in scene images with unsupervised feature learning. *Proceedings of the ICDAR*: pp. 440-445.

[4] Patel, C.I., Patel, L. & Patel, T. (2011) Handwritten character recognition using neural network. *Proceedings of the IJSER*: vol. 2, issue 5, May.

[5] Chen, X. (2016) Convolution neural networks for Chinese handwriting recognition. *Stanford University CS231n course project.* http://cs231n.stanford.edu/reports2016.html

[6] Zhang, Y. (2015) Deep convolutional Network for handwritten Chinese character recognition. *Stanford University CS231n course project.* http://cs231n.stanford.edu/reports.html

[7] CASIA Offline Chinese Handwriting Databases: http://www.nlpr.ia.ac.cn/databases/handwriting/Offline_database.html

[8] Krizhevsky, A., Sutskever, I, & Hinton, G.E. (2012) Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pp.1097-1105.

Contributions

Chao Li: Implemented transfer learning in network design, evaluation and visualization, did majority written work of poster & report.

Junsheng Pan: Implemented major data preprocessing works, setup & configure develop environment, prepared live demo tools.

Jiameng Yang: Implemented data preprocessing tools, performed major network implementation & benchmarking, literature review.