CS 230 Project Final Report

Deep Neural Networks for Offline Handwritten Chinese Character

Gendong Zhang, Xiao Zhang, Lantao Mei

1 Introduction

Chinese character recognition is a challenging task compared to the task of recognizing handwritten digits and English alphabets. In this project, we fed handwritten Chinese character images into CNN models such as basic CNN, VGG-16, ResNet50 to output the accuracy of Chinese character recognition. We then did the fine-tuning of our models to achieve better performance. In general, ResNet50 outperforms with 94.37% in the test set. Finally we extend our projects to perform handwritten Chinese sentence recognition. We generated our own dataset and trained a CRNN model.

2 Literature Review

Our work on exploring the performance of deep convolutional neural network on handwritten Chinese character recognition is inspired by Yuhao Zhang and his paper *Deep Convolutional Network for Handwritten Chinese Character Recognition* [1]. In his paper, Yuhao ran experiences on 200-class dataset using convolutional networks with different depth and filter numbers. However, Yuhao does not explores more complex and deeper models such as ResNet and DenseNet. Also, Yuhao does not train and test his model on the full dataset with more than 1 million images.

3 Dataset and Preprocessing

3.1 Offline Chinese Character Preprocessing

For Chinese character recognition, we use offline handwritten data from Institute of Automation of Chinese Academy of Sciences (CASIA). More specifically, HWDB1.1trn_gnt (5.3GB) as training set; HWDB1.1tst_gnt (1.4GB) as validation set and competition_gnt (1.4GB) as test set [2]. A python script is written to first decode the binary data in .gnt file using gb2312 which is the official character set of the People's Republic of China and then convert the binary data into .png image. After the preprocessing, the training set has 900000 images, validation and test sets have 200000 images respectively.
Sample conversion result is shown in Figure 1 below:

Figure 1: Sample Chinese Character preprocessing result and sample synthetic dataset for OCR

3.2 Optical Chinese Character Preprocessing

In order to extend our project to perform handwritten Chinese sentence recognition, we decided to make our own dataset with labels so as to proceed the extension of our project. The dataset was created from a famous Chinese Novel named Swordsman, and we generated $100,000$ 200×64 images of sentence segments sequentially with ten Chinese characters each, and each sentence is coupled with the ground truth – each character corresponds to one character from the Chinese dictionary. In addition, the font can be changed, but in order to simplify our procedure, we only used the most used Chinese font, Songti. One example of our synthetic data is shown in Figure 1.

4 Model

4.1 Simple CNN

To start with, we built a simple CNN model. As the [1] states, CNN architecture with 6 weight layers can achieve a comparatively high accuracy without consuming too much training time, since as we go deeper into the network, the time spent on training will tremendously increase. Therefore, we have implemented a 6-weight-layer CNN model using Keras. We only count the weight layers –the convolutional layers and fully-connected layers. Batch-normalization layer before every activation function to fasten the training. In addition, we stacked two convolutional layers before feeding the output into the maxpooling layer. The detailed architecture is summarized as follows in Figure 2.

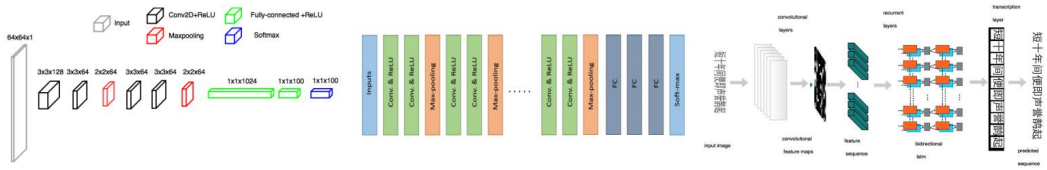

Figure 2: Architecture of CNN with 6 Weight Layers, VGG-16 and CRNN

4.2 VGG-16

Following the basic CNN model, we have further investigated the use of Visual Geometry Group Net, VGG-16, on our data set, since VGG-16 has shown robust performance in Computer Vision field and deeper convolutional neural network is critical in high resolution and classification [3]. We used the model described in [4], with the last three layers modified to meet the need of our project. For the last three layer, we used 1024-neuron fully-connected layer, 100-neuron fully-connected layer and one softmax layer(for 100-class training and testing). Figure 2 [5] shows a general architecture of a VGG-16 model.

4.3 ResNet-50

The residual learning framework eases the training of networks that are substantially deeper [6]. It solves the degradation problem by using shortcut connections between several layers. In this project, we trained ResNet50 which is a 50 layer Residual Network. Similar to VGG-16, in our implementation, we modified the last three layer of ResNet50 to meet our own need for this project. More specifically, for 100-class training and testing, we used 1024-neuron fully-connected layer, 100-neuron fully-connected layer and one softmax layer.

4.4 CRNN

Convolutional Recurrent Neural Network (CRNN) is specifically designed for recognizing sequence-like objects in images [7], and for the extension of project that to recognize handwritten Chinese sentence, this is a suitable model to try and use. Followed [7], we built our network model consisting of four main parts. The first part is a CNN to perform feature extraction. Following the CNN layers, a bidirectional LSTM component was built to do sequence processing. The third part is a fully connected layer to predict the likelihood of each character based on a dictionary. And the last Connectionist Temporal Classification (CTC) layer to compute the loss. The rightmost graph in Figure 2 is a simple illustration of the model and the recognition procedure. Since we did transfer learning, and we refer to and we refer to [7] to see more details about the model.

5 Experiment and Discussion

5.1 General Information

We trained our models on the HWDB dataset [2], which consists of around 900K training images and 200K validation images and have in total 3755 classes of characters. Besides, in order to verify

the performance, we also tested the trained baseline model on competition dataset[2], which consist of 200K images.
The network is implemented in Keras. We trained all the models with a NVIDIA TITAN X Pascal. For the models we have implemented, we used categorical cross entropy loss function to compute the loss between predictions and targets, as can be seen in Equation (1),

$$L_i = -\sum_j t_{i,j} log(p_{i,j}) \tag{1}$$

where p is predictions, i denotes the data sample and j denotes the class of the character.

5.2 CNN vs. VGG-16

At first, we chose a simple CNN model and VGG-16 model to train on a small portion of the whole dataset, which consists of only 100 classes. We trained each model in 15 epochs, with batch size being 64 per iteration. The training process for each model takes around 15 min.
When we are training the model, we have explored different methods to minimize the high variance problems encountering in the process of training, and also have drawn the accuracy plot for both training and validation, see Figure 3.

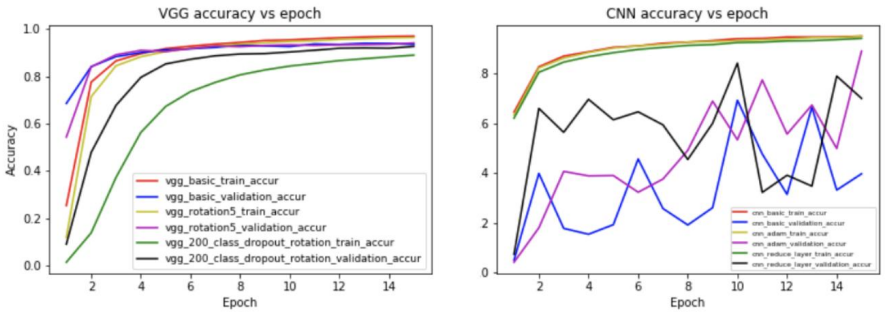

Figure 3: CNN vs. VGG-16: Accuracy vs. Epochs

From the right figure of Figure 3, the training and validation accuracy of basic 6-weight-layer CNN with RMSprop and dropout (Red and Blue) indicates there is a high variance issue, and oscillations exists in the validation accuracy curve (this might caused by dropout since basic CNN with dropout is not stable), so in order to minimize the variance problem, we tried Adam optimizer and model with fewer layers. Adam optimizer demonstrates the best performance (95.11% for training) among these three. As can be seen, the validation accuracy curves of these three methods have oscillations as the epoch increases, but the overall tendency is increasing. For VGG-16 model shown in right figure of Figure 3,the basic model without further optimizing methods (Red and Blue) shows a very promising accuracy with 97.01% and 93.47% for training and validation. While it demonstrates a small improvement if we add image rotation technique (Yellow and Magenta), with 96.45% and 94.11% for training and validation respectively. Finally, we combined the rotation technique, dropout and more classes (200 classes) into our model, and it shows a slow increase in accuracy for both training and validation at beginning, but it seems that the accuracy curves would continue increasing if we train for more epochs.

Model		Test Accuracy
Basic CNN	6 weight layers with RMSProp and dropout	41.67%
	6 weights layers with Adam and dropout	88.68%
	5 weights layers with Adam and dropout	69.53%
VGG-16	Basic VGG-16 without dropout and data augmentation	93.6%
	Image rotation = ±5 degrees	94.48%
	200 class and Dropout=0.5 and Image rotation ±5 degrees	93.75%

Figure 4: CNN vs. VGG-16: Methods and Test Accuracy

We also test our models on the test dataset. The result is shown in Table above. The testing results are overall consistent with the test/validation accuracy graph.

5.3 VGG-16 vs. ResNet-50

Then after finding the VGG-16 model performs much better than the simple CNN, we trained ResNet-50 and VGG-16 on a larger training set (still a portion of the whole dataset, this time it consists of 3755 classes, but each class only contains 30 images). With the limitation of time, we trained each model in 15 epochs, with batch size being 64 per iteration. The training process for VGG-16 takes around 10h, for ResNet-50 it takes about 16.5h.
We trained ResNet-50 and VGG-16 in same dataset, the VGG-16 is already tuned to have the best performance parameters, and we observe the ResNet-50 outperforms VGG-16, the results are shown in Figure 5.

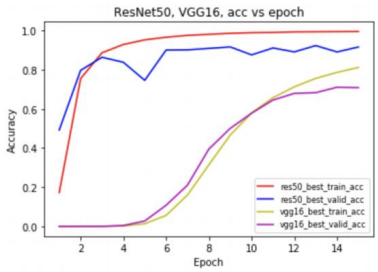

Figure 5: ResNet-50 vs. VGG-16: Accuracy vs. epochs

As shown in figure 5, the ResNet-50 obviously outperforms VGG-16 by about 20% for both training set and validation set.

We also tested the trained model on our testing set, the accuracy of VGG model is 72.66%, and for ResNet model, is reaches 87.31%, which also agrees with our expectation that ResNet performs much better than VGG.

5.4 ResNet-50 Parameter Tuning

Since ResNet-50 outperforms the VGG-16 in best performance parameters. Now we want to further tune the parameters of ResNet-50, to obtain our best model.
We trained ResNet-50 on the same training set (still a portion of the whole dataset, this time it consists of 3755 classes, each class contains 30 images).
Similarly, with the limitation of time, we trained each model in 15 epochs, with batch size being 64 per iteration. The training process for VGG-16 takes around 10h, for ResNet-50 it takes about 16.5h.
During the training process, we tuned different learning rates and different optimizers. The results are shown in Figure 6.

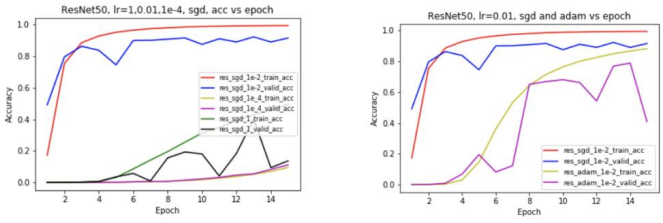

Figure 6: ResNet-50 Parameter Tuning

As you can see, the ResNet-50 with learning rate 0.01 outruns the other learning rates, and sgd optimizer works better than Adam in this case.
We also tested our model in testing set, the testing accuracy of the ResNet-50 model with best parameters is 87.31%.

5.5 Training on Whole Dataset

After obtaining the best model with best parameters, we apply the ResNet-50 with 0.01 learning rate and sgd optimizer on the whole dataset (which consists of 3755 classes with each class containing 239 images for training set and 60 images for testing set)
With the limitation of time, we only trained this model in 10 epochs, with batch size being 64 per iteration. The training process takes about 85h.
The result of training is shown in Figure 7 .

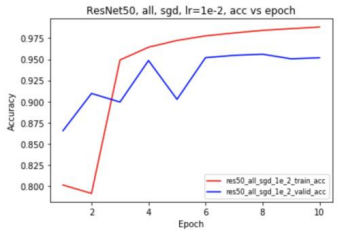

Figure 7: ResNet-50 on whole dataset

As you can see, the results performs very well even on the whole dataset, with training accuracy being 98.82% and validation accuracy being 95.19%.
We then tried this model on the testing set and the testing accuracy is 94.37%, which is a really high value.

5.6 Model Ensembling

We ensembled ResNet50 with learning rate 1e-2 with SGD optimizer and the one with Adam optimizer. However since the test accuracy of the latter model is poor, the ensembled model is not better than the first model itself. Our initial plan is to ensemble ResNet50 with DenseNet121. However in DenseNet121 each epoch takes 40 hours to train so we do not have enough time to finish it. This will be our future work.

5.7 Train CRNN model on synthetic dataset

We conducted transfer learning from an OCR for English sentence to Chinese sentence, and also refer to other sources [], and we also used a pre-trained weights due to our insufficient dataset. Due to the limited time and lack of resources such as decent dataset with ground truth, we were only able to train the CRNN model on our synthetic dataset with a small portion, with only $10,000$ images for training and validation, and $1,000$ images for testing. The result was not promising with only 30.71% accuracy on training set 28.1% on validation set and 25.78% on test set. We performed fine-tuning, but there was no improvements. We could only speculate the cause of this low accuracy was due to the fact that our synthetic dataset was not suitable for the CRNN model and also the pre-trained weights did not match our dataset.

6 Conclusion

In this project, we uses basic CNN, VGG-16 and ResNet-50 models with custom modifications to recognize Chinese character. We then fine tuned and compared models. In the end, ResNet50 outperforms with 94.37% in the test set. We further extended our project to offline handwritten sentence recognition and applied CRNN model. Even though the result is not perfect, we did error analysis and we would improve the accuracy in our future work.

7 Future Work

The handwritten Chinese sentence can be performed without the use of CRNN model, it could be done using image segmentation with OpenCv tools or other equivalent. Each character can be boxed in the sentence line and predicted separately, and finally combining each single character to recognize the whole sentence.

8 Contribution

Lantao Mei: Data preprocessing for the whole training dataset, ResNet50, DenseNet121, Model Ensembling implementation using Keras. Basic CNN model and ResNet50 fine tuning, model evaluation using competition test set and wrote introduction, conclusion, part of data preprocessing, model and result analysis and of the report.
Gendong Zhang: did preprocessing the data, built basic cnn and VGG-16 basic model, did extension of this project on different ocr models, wrote program to generate dataset with label, and write Data Preprocessing, Model, part of Experiment and Discussion, Future work of the report.
Xiao Zhang: CNN, VGG, ResNet model training testing and fine tuning. Testing different models using GPU and gathered model weights, training, validation loss and accuracy. Data analysis, poster preparation, writing final write-up.

References

[1]Zhang Y Deep convolutional network for handwritten chinese character recognition. Computer Science Department, Stanford University

[2]"Download", Nlpr.ia.ac.cn, 2018. [Online]. Available: http://www.nlpr.ia.ac.cn/databases/handwriting/Download.html. [Accessed: 22- Mar- 2018].

[3]U. Pal, A. Belad, Ch. Choisy, "Touching numeral segmentation using water reservoir concept," Pattern Recognition Lett. vol. 24, pp. 261-272, 2003.

[4]Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556(2014).

[5]S. Roy, N. Das, M. Kundu and M. Nasipuri, "Handwritten isolated Bangla compound character recognition: A new benchmark using a novel deep learning approach", Pattern Recognition Letters, vol. 90, pp. 15-21, 2017.

[6]He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[7]B. Shi, X. Bai, C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition", CoRR, vol. abs/1507.05717, 2015.

Link to github:https://github.com/tedMei/CS230