Guqin Notation and Music Style Recognition

Chen Shi
Stanford University
pupushi@stanford.edu

***Abstract*— motivated by the Convolutional Neural Networks about digit recognition and ImageNet deep neural network by Krizhevsky et al. [1], I did this project on Guqin notation recognition, which classified reduced characters with positioned 1-10 (一 -十) in handwritten Chinese characters and translated to other music recording scores. I built a four-layer convolutional neural network using adjusted CaffeNet CNN [2] model to classify 8000 images from handwritten Guqin notations into 10 distinct classes. The network achieves a test set error rate of 15.7%. The model is trained using dropout on the fully-connected layers and performed PCA with assigned randomized variable for weight decay to reduce overfitting.**

Keywords—CNN; Guqin;Notation; CaffeNet; Text detection; Handwritten character recognition; Convolutional neural networks;

I. INTRODUCTION

Guqin is a plucked seven-string instrument with a very long history of more than 2500 years). Guqin music scores are written since the late Tang Dynasty in a system known as "Jianzi-pu (減字譜, Ch'ian Ts'u P'u in Wade-Giles, literally reduced ideograph notation"). The system consists of compiling a series of left and right hand movement syntaxes into one Chinese-like character

In each of the character, there are three main categories: actual notation (正字, showing the string), accompaniment notation (旁字, showing the left-hand position) and appending notation (旁註, showing the right-hand movement). Actual notation records the "proper tones" produced by obvious plucking, while accompaniment notation records the "resonance" or the sound(s) after the pronounced tone. Appending notation records rhythm modifiers.

Previous work for handwritten recognition for Chinese characters has used radical extraction as the main method. Shi et al., [3] used active shape models to extract radicals from Chinese characters with kernel PCA, and mapping the radicals to the reference models using a genetic algorithm to search for the optimal shape parameters. Chellapilla and Simard [4] proposed a convolutional neural network using the radical-at-location feature to process the whole character image and recognize radicals at a specific location in the character. This confirmed my choice of using CNN on this Guqin notation project.

Motivated by the success of Krizhevsky et al., [1], and several other groups in applying convolutional neural networks to image classification, and in particular, to the ImageNet and CaffeNet, [2] I applied deep, convolutional neural networks (CNNs) to classifying handwritten Chinese characters of the numbers 1 to 10 (一 二 三 四 五 六 七 八 九 十) at each of the above specific positions. This can translate invaluable ancient music sheet to more readable music sheets.

Figure 1. Decomposition of Guqin notation character

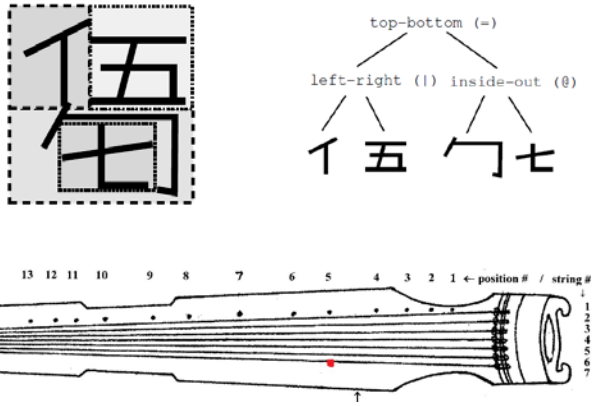


Figure 1 shows that the Guqin reduced character means string #7(七) and position #5(五). We can find the position on a Guqin as the red dot above

II. DATA

A. *Image Collection*

Guqin music sheets and handwritten music sheets were read in by text detection method [5], and cropped out to an adjusted region and saved as black and white 256*256 images. I randomly produced (一 -十) characters in different fonts to enlarge the training pool.

I modified the text detection method adjusting to the average size of the characters captured on the page. A second round of text detection was applied with Algorithm2: Radical detection algorithm (method 1) from Enzhi.et al. [6]

B. Image Detection Method

Given an image *I* and a radical template T:

Let R be the range of aspect ratio, S be the range of scale.

1. For r ∈ R

a. Change the aspect ratio of image I to r and get image *I*'

b. Compute the integral graph g of image *I* ' .

c. For s ∈ S

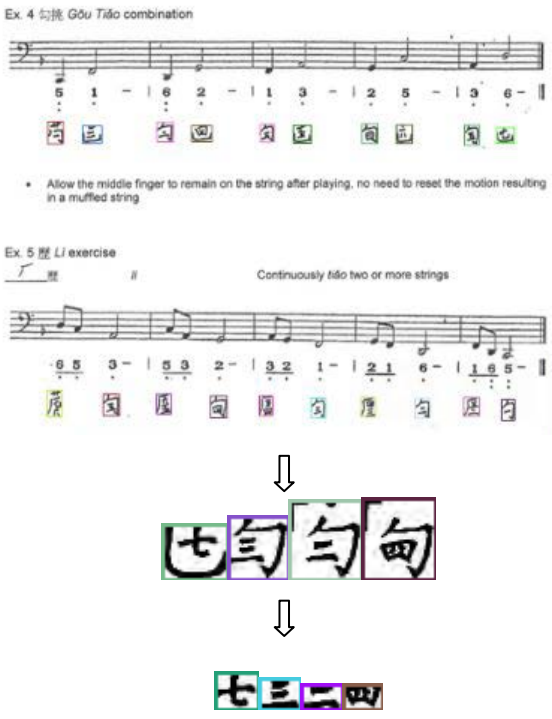
(1) Scale radical template T with s and get radical template T ' with size w× h.

(2) For each position (*x, y*) at image I ' Match the image within window (*x, y, w, h*) with template T', if they are matched, add this window to radical-object-like window set W.

2. Combine neighbor radical-object-like windows in W and output W. *x*, *y*, *w*, *h* later served as the decision boundary on which image is the string notation or the finger position notation.

Figure 2 below demonstrates how the above method worked in this project.

Figure 2. Text detection from music sheets



III. FEATURES

As mentioned above, for each black and white reduced character image, 1-3 smaller images were collected with position of the cropped region and saved in the file name by text detection. Image location (*x*, *y*, *w*, *h)* was detected and saved.

Now each single image should only contain a single unit of digit information. 96 convolutional kernels were learned by the first convolutional layer in the following model on the normalized 256*256*3 input images.

IV. MODEL

After text detection, my problem simplified to deep learning 一 -十 (1-10) in Chinese using CNN. This requires further tuning. Applying Krizhevsky's ImageNet network, I built a four-layer convolutional neural network with ReLUs, for 10 classifiers using CaffeNet CNN models.[2]

A. CaffeNet CNN Structure

CaffeNet provided a complete set of layer types including: convolution, pooling, inner products, and nonlinearities, local response normalization, element-wise operations, and losses like softmax.

The first convolutional layer learned 96 convolutional kernels from the normalized 256*256*3 input images. The kernels of the second, third and fourth convolutional layer are connected to those kernel maps in the previous layer. The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size 5 * 5 * 48..The third convolutional layer has 384 kernels of size 3 * 3 * 256 connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size 3 * 3 * 192. The fully-connected layers have 4096 neurons each. The ReLU non-linearity is applied to the output of every convolutional and fully-connected layer.

B. ReLU nonlinearity

A neuron's output f as a function of its input x is f(x) = max (0; x) as non-saturating nonlinearity model. Since the collection of cropped image and limited music score resource, I did not have the luxury of a big pool of data as ImageNet. I chose nonlinearity in neuron learning to achieve a faster learning speed.

C. Stochastic Gradient Descent

I incorporated CaffeNet's fast and standard stochastic gradient descent algorithm by Jia et al., [2]. The training model was using stochastic gradient descent with a batch size of 760 examples randomly selected from 10 pools of text detected classified images. With momentum of 0.9, I used a very small weight delay 0.0005 suggested by the above paper [1].The algorithm was showed as below:

$$v_{i+1} := 0.9 * v_i - 0.0005 * E * w_i - E * < \partial L/\partial w > w_i$$

$$w_{i+1} := w_i + v_{i+1}$$

Where i is the iteration index, v is the momentum variable, E is the learning rate, and $< \partial L/\partial w >$ ₩i is the average over the ith batch of the derivative of the objective with respect to wi,

I initialized the weights in the first layer from a zero-mean Gaussian distribution with standard deviation 0.01. Other layers were dealt with by the same method. This initialization accelerates the early stages of learning by providing the ReLUs with positive inputs. The initial neuron biases in the remaining

layers with the constant 0. By increasing the sample sizes to 7600 and rate stopped improving with the current learning rate 0.0075 from initialized rate 0.01. It took about18 hours to finish the training. The mini-batch size is 20 images as initial default; it is 2 times of the number of classifiers and not too big. I wanted the weight be updated at lease meeting one kind out of each classifier. The size of 20 gave me a safe bet on that also not too big to slow down each learning step.

D. Overfitting diagnosing and Dealing with Overfitting

As we know CNN could very likely lead to overfitting, I tried randomly select training sample and adding 一 -十(1-10) characters in different fonts to replace and dilute the existing training sample and to disturb the homogeneity which also served as a control and new samples here. The cross validation kept around 95% for overall training accuracy. It worked well as the training accuracy was stable between new mixed training from the original set. One possibility is that for a CNN, my data (8000 images) just were not large enough for overfitting. It took about 18 hours to train the model for 90 folders by 20 iterations.

To completely avoid overfitting, I performed PCA with assigned randomized variables (αi) to alter the intensities of the RGB channels in training images where pi and λi are ith eigenvector and eigenvalue of the 3 *3 covariance matrix of RGB pixel values.

V. RESULTS

I managed to collect about 760 training images for each classifier 一 -十 (1-10) and training errors range from (0% for 十/10 to 12.2% for 三/3). The sample size of test is pretty small as I manually mark the correct answer for each note which slowed down the whole process. Below shows the average results from 10 classifiers. Accuracy rate per classifier is listed below:

TABLE I. RESULTS FOR 10 CLASSIFIERS

Classifier	CNN Training and Test Results			
	Training Sample	Training Accuracy	Test Sample	Test Accuracy
一	760	98.9%	74	90.3%
二	762	98.0%	191	94.7%
三	757	87.8%	24	37.5%
四	756	99.7%	184	97.9%
五	771	93.1%	158	100.0%
六	757	92.3%	323	93.75%
七	759	94.8%	156	86.7%
八	763	99.7%	18	100.0%
九	763	91.3%	132	76.9%
十	759	100.0%	97	55. 6%

TABLE II. RESULTS FOR OVERALL

Classifier	CNN Training and Test Results			
	Training Sample	Training Accuracy	Test Sample	Test Accuracy
1-10	7607	95.3%	1357	89.3%

VI. DISCUSSION

The experiment results are very solid in 8 out of 10 classifiers. But 十/10 and 三/3 did not behave as good as the other 8 groups. I did not train each model individually but looking over the 10 categories I would suspect that 十/10 is overfitting and 三/3 was not sufficiently trained. As I am adopting the method of assigning randomized variable for weight decay to reduce overfitting only, the results were affected.

To simplify my experiments, I did adjust the strategy to have a two-step approach (text detection + CNN) as there was tons of awesome work done in both fields. Adopting a simpler CNN model also saved computational time. There was about 10% data loss between the two steps partly because they didn't contain 一 -十 (1-10) in the music score or these characters were not cropped out correctly in the first step. If I was starting from beginning again, I would allocate more time on learning about CNN pooling window and build a neuron for position detection in the image instead of spending much time just focusing on cropping out the images.

Originally, I expected to tell the style differences between the schools of Guqin playing from 2 schools of notations I collected. It ended up that the notations do not carry enough rhythm information and it is more up to each player to interpret the piece. This brought up an interesting direction that we can build a model to read in the recording and compose according to the style of it.

VII. CONCULSION AND FUTURE WORK

I am quite excited the model did successfully recognize 一 (1) to 十 (10) with an average 10.7% test error rate. The result differs between classifiers depending on the sample size and on how well they were trained in the CNN. As the model applies the same for all 10 classifiers, I might need to fine tuning.

In the future, I would implement Dropout by Hinton et al, [8] and also enlarge the notation database. As the notation gets more complicated, I would also add adjustment to pooling position into CNN learning model and introduce more notations on right hand playing techniques. There is existing neural network demo that tries to recognize real time handwritten character and I will try to develop the feature into an App to assist Guqin players.

VIII. ACKNOWLEDGMENT

Thanks all the CS229 teaching staff and especially Francois Germain for inspiring questions and comments.

REFERENCES

[1] Krizhevsky, A et al. ImageNet Classification with Deep Convolutional Neural Networks NIPS 2012

[2] Jia et al Caffe: Convolutional Architecture for Fast Feature Embedding Proceedings of the 22nd ACM international conference on Multimedia, 675-678

[3] D. Shi, G. S. Ng, R. I. Damper, S. R. Gunn, Radical recognition of handwritten Chinese characters using GAbased kernel active shape modelling," IEE Proceedings of Vision, Image & Signal Processing, vol. 152(5), pp. 634-638, 2005.

[4] K. Chellapilla, P. Simard, A new radical based approach to offline handwritten East-Asian character recognition," in:Proceedings of 10th International Workshop on Frontiers in Handwriting Recognition, La Baule, France, 2006.

[5] .Chen, Huizhong, et al. Robust Text Detection in Natural Images with Edge-Enhanced Maximally Stable Extremal Regions." Image Processing (ICIP), 2011 18th IEEE International Conference

[6] Enzhi,N et al. A Radical Cascade Classifier for HandwrittenChinese Character Recognition JOCCH Volume 3 Issue 3, March 2011

[7] Enzhi,N et al. Handwriting input system of chinese guqin notationJOCCH Volume 3 Issue 3, March 2011

[8] E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neu-ral networks by preventing co-adaptation of feature detectors. CoRR, abs/1207.0580, 2012.