Convolution Neural Network for Traditional Chinese Calligraphy Recognition

Boqi Li
Mechanical Engineering
Stanford University
boqili@stanford.edu

Abstract

In this project, we will use Convolution Neural Network to recognize different traditional Chinese calligraphy styles. We will train Convolutional Neural Network on a 5-class dataset which contains 15000 instances. We prepressed the images to have sizes of 96x96, and constructed 8 different models to test on different depth and filter numbers. We evaluated the performance of each model and analyzed the recognition ratio of each style. We also visualized both the reconstructed images maximizing activation and feature activation maps.

1. Introduction

Traditional Chinese calligraphy (TCC) is an essential part of Chinese traditional art and culture. The earliest TCC work can be dated back to 11 century BC. Along the history of TCC, many different styles of calligraphy have been developed. While some of the styles evolved to be modern style used by people in their daily life, others become pure formats for art performing.

In recent years, many databases of TCC have been setup, and a large amount of TCC have been digitized for both research purpose and common practice of art. Therefore, the need for TCC browsing and recognizing is increasing. Many approaches have been introduced for this need. Most of suggested solutions we searched are based on certain feature extraction and K-nearest neighbor technique. On the other hand, CNN has been widely used on hand-written character recognition. Based on such situation, we therefore want to explore the performance of CNN on TCC style recognition. However, even though TCC is one special type of hand-written characters, it doesn't make recognizing TCC styles and recognizing hand-written characters more similar tasks. Most TCC are written by traditional Chinese brushes, which make the strokes much thicker than normal hand-written characters and as a result store more shape information. Also in terms of the purpose of the classification, recognizing TCC will explore more on the similarities and differences of character styles.

There are five major TCC styles: *seal script, clerical script, standard script, semi-cursive script, and cursive script.* Fig. 1 shows examples of the same TCC characters in all five major styles.

Figure 1. *Standard script, clerical script, seal script, cursive script, and semi-cursive script* (From left to right)

The standard script is used in daily life. The clerical script is similar to standard script, but has thicker strokes and shorter and wider in shape. It was once used in the official documents in ancient China. The seal script is the oldest of the five style, and is usually carved on stones instead of paper. Semi-cursive and cursive are defined relatively. The semi-cursive style is more casually written than standard script but more carefully written than cursive style. If the character is written fast and casual enough, it becomes the cursive style. In this sense, we also expect to see correlations between some styles of TCC.

Therefore, in this project the input of our algorithm is an image of TCC character. We then use CNN to output a predicted style category of that character. We train CNN on individual TCC characters and test the influence of different depth and filter numbers on prediction performance. We also visualize the exacted features to evaluate whether the features make sense or not and how CNN recognizes TCC.

2. Related Work

In our literature search, we found that there are recent works which implement CNN technique on Chinese character recognition. Yang *et al.* [1] applied deep convolutional network on handwritten Chinese characters with a serial-parallel architecture to train several neural networks in parallel and make prediction based on a probability threshold on each CNN output. In a recent CS231N project, Zhang [2] trained CNN on handwritten Chinese characters dataset and evaluated the performance of different CNN filter numbers and depth. Although handwritten Chinese characters share similarity with TCC, there are major difference among the them. In terms of recognitions goal, handwritten Chinese character recognition aims to predict the label of every character, whereas in our project, our goal is to recognize TCC styles, which makes the task easier in terms of number of classes. Another difference is that, handwritten Chinese characters are usually written with pen, which have very thin line width, whereas TCC characters are written with brush and have significant features in the width of every line.

Many TCC recognition approaches focus on how to extract both the contour features and the width features. And different variations of HOG feature have been used. For example, Lu *et al.* [3] used a method of feature extraction called SC-HoG descriptor to represent the shape of TCC characters. However, most of the approaches use K nearest neighbor as classifier to solve the problem. And we didn't find examples of using CNN on TCC recognition.

3. Methods

We use architecture of CNN based on VGGnet model [4]. Since one of our goal in this project is to investigate the influence of number of layers and number of filters on the performance of the classification. We built eight different CNN models. The difference of the configuration of the models are the number of layers and number of filters. In every model, we use filters of 3x3 with stride of 1 and zero-padding of 1 to preserve input height and width, and use max-pooling with 2x2 filters and stride of 2 for down-sampling. Table 2 shows the detail layout of each model. Our models are built using Keras library [5] with theano backend [6].

Our baseline model is L6, which has 3 convolutional layers followed by 3 fully-connected layers. We then increase one more convolutional layer in L7 and L7+ models. The difference between L7 and L7+ is that we increase the number of filters in L7+ model. In L9 and L9+ model, we add two more convolutional layers between non-linearity and max-pooling. As is described by [4], we stack two convolutional layers with 3x3 filters. The stacking of two 3x3 convolutional layers is equivalent to one 5x5 convolutional layers in terms of the effective receptive field, but with fewer number of parameters and more non-linearity activation layers. Between L9 and L9+, we want to test the influence of putting additional layers in the front versus in the back of the convolutional layers. We added 2 more convolutional layers in L11 and L11+ models. And in L11+ we add batch-normalization layers after every convolutional layer. The batch normalization layer perform normalization for each training mini-batch. In training process, it computes the mean and variance over the mini-batch input and then normalize the input. And the scale and shift parameters can be learned during the training process. The equations are shown below.

Table 1. CNN configuration of 8 different models

CNN Configuration							
L6	L7	L7+	L9	L9+	L 11	L 11+	L 13
Input Image (1x96x96)							
Conv3-32	Conv3-32	Conv3-64	Conv3-32	Conv3-32	Conv3-32	Conv3-32 BN	Conv3-32
			Conv3-32		Conv3-32	Conv3-32 BN	Conv3-32
Maxpooling (2x2)							
Conv3-64	Conv3-64	Conv3-128	Conv3-64	Conv3-64	Conv3-64	Conv3-64 BN	Conv3-64
			Conv3-64		Conv3-64	Conv3-64 BN	Conv3-64
Maxpooling(2x2)							
Conv3-128	Conv3-128	Conv3-256	Conv3-128	Conv3-128	Conv3-128	Conv3-128 BN	Conv3-128
				Conv3-128	Conv3-128	Conv3-128 BN	Conv3-128
Maxpooling(2x2)							
	Conv3-256	Conv3-512	Conv3-256	Conv3-256	Conv3-256	Conv3-256 BN	Conv3-256
				Conv3-256	Conv3-256	Conv3-256 BN	Conv3-256
Maxpooling(2x2)							
							Conv3-256
							Conv3-256
							Maxpooling (2x2)
FC-512							
FC-256							
FC-5							
Softmax							

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \tag{2}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{3}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \tag{4}$$

According to *Ioffe and Szegedy*[7], batch normalization is able to stabilize the learning process, and allow using of higher learning rate. These advantages of batch-normalization help us a lot during the training of L11+, but

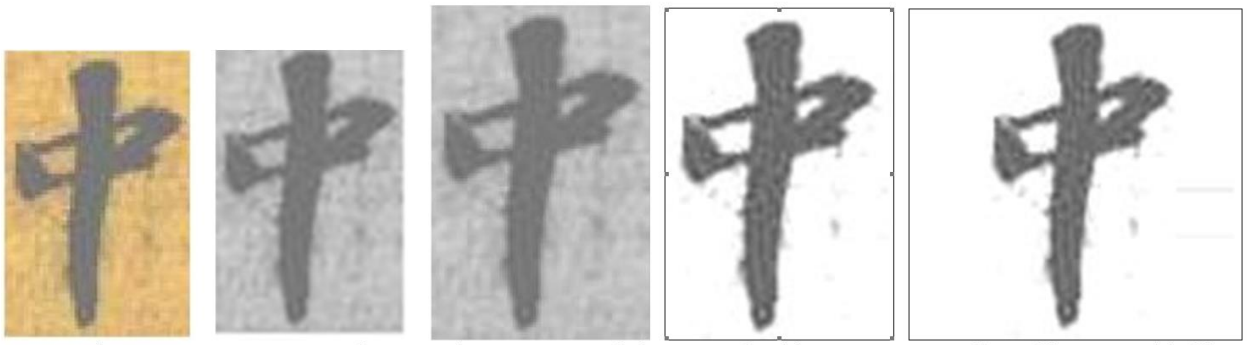

Figure 2. Data Pre-processing: Raw image, Grayscale image, resized image, contrast adjusted image, padded image

also introduce some problems, which will be discussed in the next section. In L13 model, we add one additional conv-conv-max-pooling structure to make the network deeper.

Although not shown in Table 1, we use non-linearity ReLU activation after convolutional layers in every model except L11+. In L11+, the ReLU layers are used after batch normalization layers. We use softmax function to compute the loss. The equation of softmax function is shown below.

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \tag{5}$$

For weights initialization, we use normal initialization based on fan-in with a variance equal to $2/n_l$, as is described by He et al. [8], the equation is shown below.

$$\frac{1}{2} n_l Var[w_l] = 1, \quad \forall l. \tag{6}$$

Such method of initialization is used to prevent complete saturation of neurons as we build deeper and deeper network models.

For optimizer, we use stochastic gradient descent with Nesterov Momentum update. The idea is that with Nesterov momentum update, we are able to build up velocity along flat directions and move one step first along the momentum update and use the gradient at the new point to make the gradient step, instead of making the momentum step and gradient step at the same point. The update equations are shown below.

4. Dataset and Pre-processing

During initial literature review, we studied several TCC database. We choose to use a Calligraphic Character Dictionary called CADAL, as is introduced by Zhang and Nagy [9] Although CADAL contains thousands of calligraphy images, we didn't choose to use whole images as data. Instead, we used CADAL's pre-segmented calligraphic characters to form our dataset. These characters are segmented from whole calligraphic works by CADAL. In this way, we also increase the size of our dataset. We collected 15000 TCC characters, with 3000 characters for each style. We use 12000 characters for training, and 1200 characters for validation, and 1800 for test.

We did preprocessing on our dataset. The dataset contains JPEG format RGB images with various dimensions. First, we imported the images into MATLAB and computed the mean height and width of the images. The result is shown in Table 2.

Table 2. Mean Height and Width of Images of Dataset

	Standard	Clerical	Seal	Semi-Cursive	Cursive	Total
Num.	3000	3000	3000	3000	3000	15000
H.	104	103	170	123	119	124
W.	112	131	123	112	95	136

Initially, we planned to use size of 128*128 as normalized dimensions in order to achieve more max-pooling layers with stride of 2. However, when we loaded our models on rye machine, the memory wasn't enough. Therefore, we rescaled the image to 96*96, which could still achieve enough max-pooling layers, as well as saving memory. We used the same input image sizes for every model for consistency even though we moved our model to AWS later.

Then we converted the RGB images to grayscale images. We want the images to be resized according to their largest dimension. Therefore, the largest dimension of the images will be 128, and the aspect ratio remains the same. Xia *et al.* [10] suggested that when doing preprocessing on TCC images, man-made seals and background (which sometimes contains ink stains) should be removed. However, since we will use CNN, we want to test the capability of CNN on handling these conditions. Therefore, we didn't remove any seals or ink stains. Next, we adjusted the contrast level of the images, which is suggested by Zhang [2]. We padded the images by repeating the border elements to center the character in the images. Finally, we computed the mean value of all the images and subtracted the mean from every image. We exported the dataset as a single array of dimension 15000*1*96*96, which will be used for training and testing. The image example after each step is shown in Fig.

2.

5. Experiments

As is mentioned in the previous section, we trained our models on 12000 samples and used 1200 samples for validation. Initially, we could only train with a batch size lower than 80 on rye machine. Larger batch size would cause memory allocation error since the rye machine doesn't have more memory. We tested several batch size and found that lower batch size would increase the training time without enhancing performance significantly. So we used a batch size of 64. We used the same batch size for every model for consistency even though we moved our model to AWS later.

For every model, we trained the model by one epoch with no learning rate decay, no weights regularization or dropout, and tuned only learning rate, starting from a learning rate value that was too high. And then we decreased the learning rate by a factor of two until we find a good range of learning rate. Then we added learning rate decay if the loss stopped to decrease in two or more epochs. We also added weight regularization and dropout if we saw overfitting. Also, we found that adding dropout of 0.25 after non-linearity to every model is good enough if overfitting is observed. Excessive overfitting could be dealt with higher weight regularization rate. With a narrower range of learning rate and weight regularization, we finally do random search to find the best pair of these two hyper parameters. In practice, this procedure reduced our time for tuning than doing random search on a wide range in the beginning, since we have 8 models to train. For most models, 15 epochs are enough to observe the validation accuracy stabilize during the last 1-2 epochs. Table 3 shows the hyper parameters we used for every model. To evaluate the performance of every model, we tested them on 1800 test samples to exam the overall accuracy of correct style recognition rate.

Table 3 Hyper parameters choices for each model

	L6	L7	L7+	L9	L9+	L11	L11+	L13
lr	1.62e-6	8.1e-6	3.4e-5	4.6e-5	5.1e-5	3.3e-5	8.2e-4	1.96e-5
decay	5.5e-3	1e-3	1e-3	1e-4	1e-3	1e-3	1e-3	1e-3
wr	2.1e-6	0	1.3e-6	1.9e-8	0	0	3.1e-4	1.1e-6

6. Results and Discussion

Table 4 shows the results of the performance of the 8 models. Comparing L6, L7, L9, L11, and L13, it suggests that increasing the number of layers will improve the performance. It appears that increasing the number of filters doesn't improve the performance, as both the validation and test accuracy of L7 model is a bit higher than those of L7+ model. Putting additional layers in the back of the convolutional layers have a better performance than putting them in the front, as L9+ has higher validation and test accuracy than L9. Adding batch normalization also improve the performance as L11+ model has a higher validation and test accuracy than L11. Also, the performance of L11+ is also better than L13 which has more convolutional layers but doesn't have batch normalization layers.

Table 4. Validation and Test Performance of All Models

Model	Validation Acc	Test Acc
L6	74.5%	73.3%
L7	77.4%	77.0%
L7+	76.0%	74.6%
L9	83.3%	81.2%
L9+	84.6%	83.4%
L11	88.4%	83.6%
L11+	90.0%	88.6%
L13	89.7%	86.7%

Then we analyzed the recognition ratio of all classes performed by the L11+ model, which has the best validation and test accuracy among all models. The recognition ratios are shown in Table 4. Each column represents the ground truth style of a sample, and each row represents the predicted style of a sample. The numbers in the table stand for the ratio of all samples of a certain style in that corresponding column that are labeled as a certain style in that corresponding row. Therefore, the numbers on the diagonal represents the correct recognition ratio of the five styles. According to the table, the top 3 accuracy belongs to the clerical, cursive and seal styles.

Table 5. Recognition Ratio of all classes in L11+ Model

		Ground Truth				
		ST	CL	SE	SC	CU
Prediction	ST	0.847	0.056	0.019	0.114	0.025
	CL	0.042	0.933	0.008	0.003	0.000
	SE	0.006	0.000	0.922	0.006	0.031
	SC	0.072	0.011	0.033	0.800	0.017
	CU	0.033	0.000	0.017	0.078	0.928

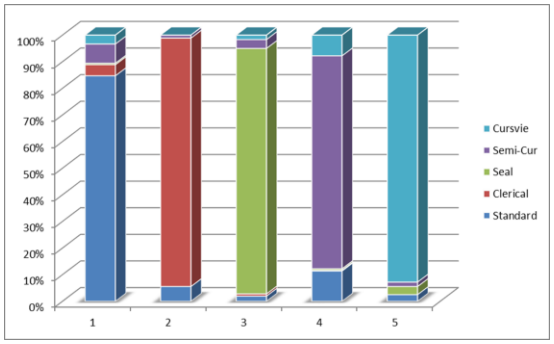

Figure 3. Recognition Ratio of all classes in L11+ Model

Fig. 3 shows the same results as table 4 for better presentation of the recognition ratio, it can be seen from the figure that a significant number of semi-cursive samples are mislabeled as the standard and cursive style. This is largely because that semi-cursive samples are sometimes similar to standard and cursive style samples. According to the definition of the semi-cursive style, it is more casual than the standard style, and less casual than the cursive style. The relativeness in the definition suggests that semi-cursive samples are more difficult to be recognized. Some examples of the mislabeled samples are shown in Fig. 4.

Figure 4. Examples of Mislabeled Images

One important characteristic of these mislabeled samples is that they are all simple Chinese characters with very few strokes. This makes their styles less explicit. In fact, some of these characters are really ambiguous in terms of styles. And they are defined into a certain style largely because of the style of the whole TCC image from which they are cropped. This also suggests that using images that contains several TCC character as sample input instead of individual character may output better performance.

For visualization of the L11+ model, we applied two methods to evaluate the feature activation map. Since we use filter size of 3x3, visualizing the filter weights directly is not very helpful because of the low resolution. Instead, we reconstructed images of feature activation maps at several layers. First, we loaded the model with the weights of the best L11+ model, and use images generated with random noise as input. We then extracted the output of the target convolutional layer after the ReLU activation, and defined a customized loss that corresponded to the activation. Next, we back propagated the model to find the gradient of the input image that maximize the loss, and use gradient ascent to generate images. The result images are shown in the Fig. 5. Since the reconstructed images are grayscale images, it is harder to see the feature textures in the images. Still, it can be observed that some images are filled with lines of certain directions, and some images have small circles which look like bubbles. And some images appear to be the same but with different rotation angles. These features suggest that the CNN model is looking for certain patterns within the images to increase activation.

The other method for visualization is to use real images as input and output the feature activation map after ReLU directly to see the activation. First, we chose 5 images from different styles and extracted the activation maps of different convolutional layers. The feature activation maps of several filters are shown in Figure 6. From the output of the first convolutional layer that uses training images as input, it shows that some of the filters activates certain edge lines of the character, and other filters activates the body of character strokes.

During the observation of each filter, we found that the feature map of some filters are completely black. And for

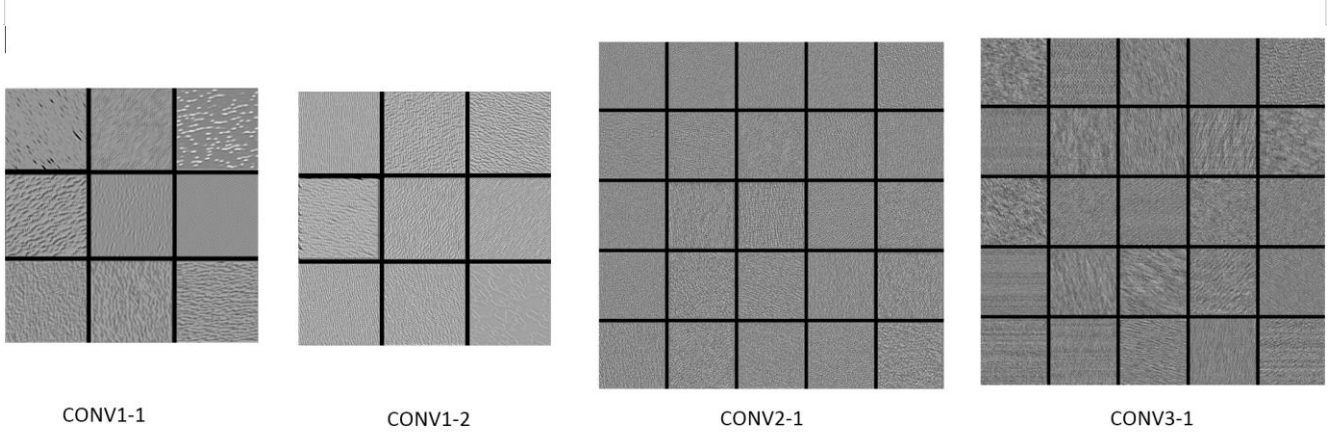

Figure 5. Reconstructed images maximizing feature activations of several layers

deeper layers, we observed increasing potion of filters that are completely black. Therefore, we focused on a fixed filter and found that some filters activate some images while are completely back on other images. This suggests that some higher layer filters are looking for certain features of the input images.

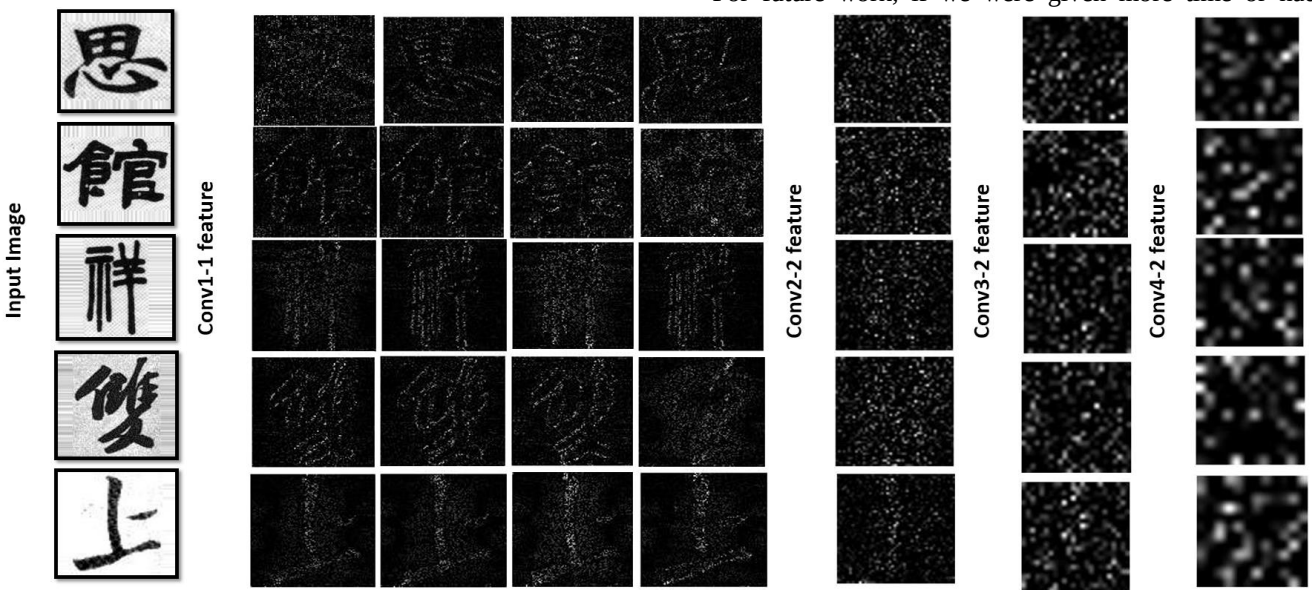

Figure. 6 Feature activation map of different style images

7. Conclusion and future work

In this project, we explored the performance of CNN models on TCC recognition. We constructed 8 different models with different number of convolutional layers and different number of filters. And we found that increasing number of convolutional layers, in other word, building deeper network models increases the overall recognition accuracy of the model. And increasing number of filters didn't increase the accuracy in our case. We also found that the best model in our configurations is the 11-layer network with batch normalization.

During visualization of, we used two methods to reconstruct the images which maximize activation of certain layers, as well as to view the feature activation map directly. And we observed that there are images containing series of lines in a certain directions and images containing small bubble like features. We also noticed that some images are similar in terms of the texture displayed but have different rotation angles.

When viewing the feature activation maps, we found that in the first convolutional layers, some filters activate a certain part of the edge of the characters, while other filters activate the body of each stroke of the characters. We also found that while some higher layers' filters are completely black in one images, they show activation when seeing other images.

For future work, if we were given more time or had more teammates, we would look into the work of recognizing images that have several characters instead of only one. We would also do a better data-preprocessing and do data augmentation to the training images.

8. References

[1] Yang, Weixin, et al. "Improved deep convolutional neural network for online handwritten Chinese character recognition using domain-specific knowledge." *Document Analysis and Recognition (ICDAR), 2015 13th International Conference* on. IEEE, 2015.H.

[2] Y. Zhang, "Deep Convolutional Network for Handwritten Chinese Character Recognition", CS231N course project

[3] Lu, Wei-ming, et al. "Efficient shape matching for Chinese calligraphic character retrieval." *Journal of Zhejiang University SCIENCE C* 12.11 (2011): 873-884.

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

[5] François Chollet, Keras, GitHub, 2015, https://github.com/fchollet/keras

[6] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley and Y. Bengio. "Theano: new features and speed improvements". NIPS 2012 deep learning workshop.

[7] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[8] He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[9] Zhang, Xiafen, and George Nagy. "The CADAL calligraphic database." *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*. ACM, 2011.Y.

[10] Xia, Z. Yang, K. Wang, “Chinese calligraphy word spotting using elastic HOG feature and derivative dynamic time warping”, *Journal of Harbin Institute of Technology (New Series)* 21(2):21-27 · March 2014