CS229 Project Milestone

Li Deng Liyi Wang Zhaolin Ren[a]

[a]*SUID: dengl11 liyiw rzl*

Abstract

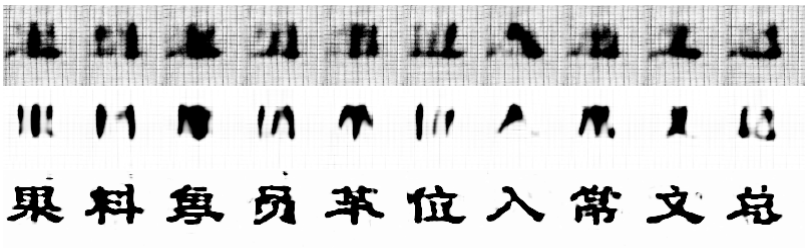
As an intermediate step toward our final goal of caligraphy font reconstruction, font classification and font learning have been implemented for the milestone report. Classification are implemented in *Keras* with one-layer model, and almost perfect result is obtained. Font learning with 3-layer Convolutional Neural Network architecture is implemented in *Tensorflow*. After training of 2000 characters, font style can be generated for new characters with pretty good result.

Keywords: Font Style Conversion, CNN, Tensorflow

1. Introduction

As a part of Chinese ancient culture, the calligraphy of Chinese characters contains various font families. These fonts include Regular(Kai Shu), Clerical(Li Shu), Semi-cursive(Xing Shu), Cursive(Cao Shu), Seal(Zhuan Shu), etc. However, most ancient calligraphy works are carved onto stones, thus after years of weathering, many of them are incomplete. The goal of our project is to, first, recognize which font certain a character belongs to, and second, reconstruct the incomplete characters based on their font.

For the reader's overview, below is a sample of character regeneration evolution.



2. Method

At this stage, our machine learning approach to recognizing and reconstructing Chinese characters is Convolutional Neural Network(CNN), which is a efficient tool in computer vision problems. Typically, CNN contains several hidden layers such as convolutional layers, maxpool layers.

The convolutional layers contains two operations: convolution($f(x) = w^{\mathrm{T}}x + b$) to the whole image based on a certain strides, and Rectified Linear Unit (ReLU, $f(x) = max(0, x)$). Maxpool layer is usually a 2x2 filter with stride of 2. It simply choose the maximum value of the 4 numbers.

A lot of python packages are helpful to implement CNN. Font recognition is a classification problem, which is well-done by Keras package. While in terms of reconstruction of characters, tensorflow, which is also the backend of Keras, is more powerful to achieve this goal. Our milestone outcome of reconstruction is transforming a character from a certain font to another.

3. Implementation and Results

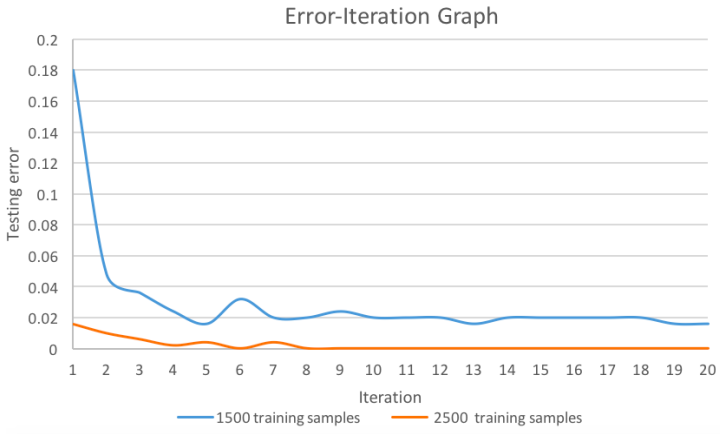
3.1. Font Pre-processing

For data reading purpose, bitmaps (80x80 pixels) are loaded from the (*.ttf*) files, each one representing a character. As a result, a font style file(*.ttf*) is converted into a (*.npy*) file, with dimention $[nCharacter \times characterWidth \times characterHeight]$. Then 2994 characters are randomly picked from it, and each batch of 20 characters are used for each training step.

3.2. Font Classification

To simplify the work, we choose 5 Chinese font libraries(*.npy* files) as our dataset, including Clerical, Song, Yahei, Regular, Semi-cursive. As input(X), bitmaps(80x80 pixels) are loaded from the (*.npy*) files, each one representing a character. As output(y), the fonts are labelled as 0 to 4 and then transformed into one hot vector to apply softmax approach. Since CNN is very powerful in classification, our Keras model only use one convolutional layer with 32 filters(size 5x5).

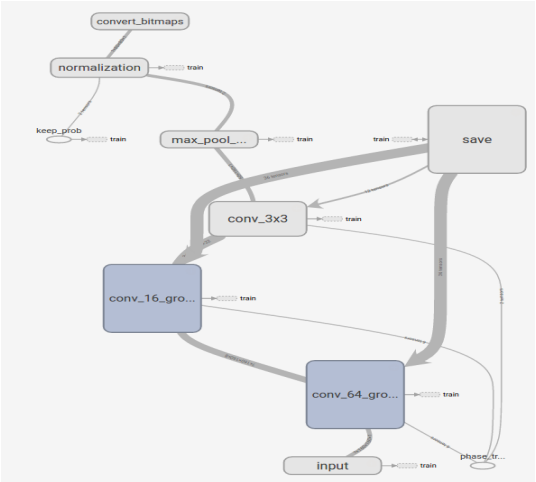
Our result are shown in the chart below. As can been seen, with 2500 training examples, the model converges very fast and it ends up with 0 testing error. That means we get very accurate prediction of font classification.



Error Diagram

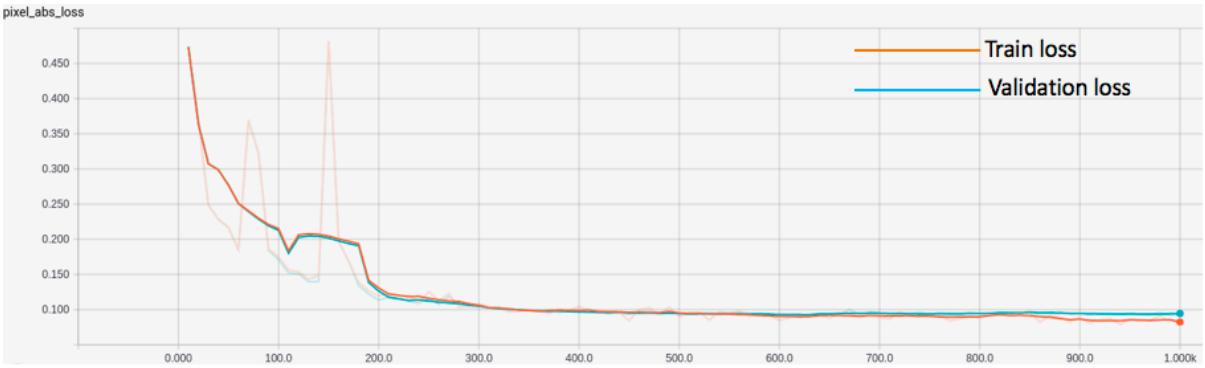
3.3. Font Generalization

In character construction, we also choose font style files(*.npy*) as our training and testing dataset. This time, we choose bitmaps of characters(160x160 size) from Song font as input(X) and bitmaps of characters(80 x80 size) from Clerical font as output(y). Each pair of bitmap from X and y represent the same Chinese Character.



CNN Architecture Generated on TensorBoard

Also shown above, our CNN model has three convolutional layers, the filter sizes are 64x64, 16x16, 3x3. While three layers cannot give perfect regeneration, we can still tell which character it is and which font it belongs to(e.g. the characters in Introduction part clearly represents Clerical font).



Loss Diagram

4. Future Work

So far, font classification and font learning have been implemented with satisfactory results. But they are limited to training and test examples from standard font style file, which makes training and testing comparatively much easier. The next step is to learn from real calligraphy, and ultimately to reconstruct from weathered calligraphy characters.