Chinese Character Stroke Counting

Github: <https://github.com/samrjack/CharacterStrokeCount> Samuel Jackson

# Background Information

Chinese[[1]](#footnote-1) is a truly unique language with many interesting facets such as its tonal pronunciation, lack of conjugation and a plethora of homophones. Arguably the most interesting aspect however is its writing system. The Chinese writing system is one of the oldest known systems on earth, and is the oldest still in use today. It is also the only prolific writing system that exclusively uses glyphs (characters) instead of an alphabet or syllabary. As such, there are thousands of different characters[[2]](#footnote-2). Their complex shape and variety has necessitated a rigorous system that dictates exactly how a character should be written. This system specifies the number and ordering of strokes which give a character a standard shape. An example can be seen below:



These attributes of a character are often used not only for writing, but also for classification in dictionaries. At first glance it may appear that every line constitutes its own stroke, but as seen above in the 3rd stroke, this is not the case.

# Project Goals, Specifications, and Milestones

This project aims to use machine learning to read in a picture of a Chinese character[[3]](#footnote-3) and predict its stroke count. This program will start by preprocessing the provided image to standardize it, then will use a neural network to predict the stroke count of the given character.

The error of this network will be quantified by comparing the predicted stroke count to an accepted standard.

In order to complete this project, the following steps will be taken:

1. Procure data
2. Data visualization
3. Run simple models on the data
4. Run complex models on the data
5. Evaluate results

# Dataset

## Definition

In order for this project to work, a large dataset of character images will be needed. There will need to be variety in the characters so that the results may be generalized more easily. The data set must contain a wide variety of characters (including both simplified and traditional variations). Every character must also have an associated trusted stroke count to compare any training to.

## Approach

In order to generate the sizeable training set, I wrote a program that produces images of characters using font packages. I then applied this program to [project Gutenberg’s version of Journey to the West](http://www.gutenberg.org/cache/epub/23962/pg23962.html).

To do this, I first found 6 fonts to use that gave varied styles to the characters (the number may be reduced later if performance is too poor). Then my program downloads the story from the web, filters out all duplicate characters, and runs them through the image producing function. This story however only contains traditional characters, therefore I wrote a small add on program that takes every character and adds its simplified variation (if any exists) to the list of characters.

Finally, I went through and removed (by hand) the 2000 or so pictures that were blank due to a font not supporting a character. In total producing all the data from scratch takes about 30 minutes, therefore I saved a copy of my images to a zip file (which can be found on [github](https://github.com/samrjack/CharacterStrokeCount)).

To figure out what the stroke count of a given character is, I found a file online containing the stroke counts of nearly every Chinese character in unicode. Although I cannot verify this entire file due to its length, I haven’t found a mistake yet.

## Challenges

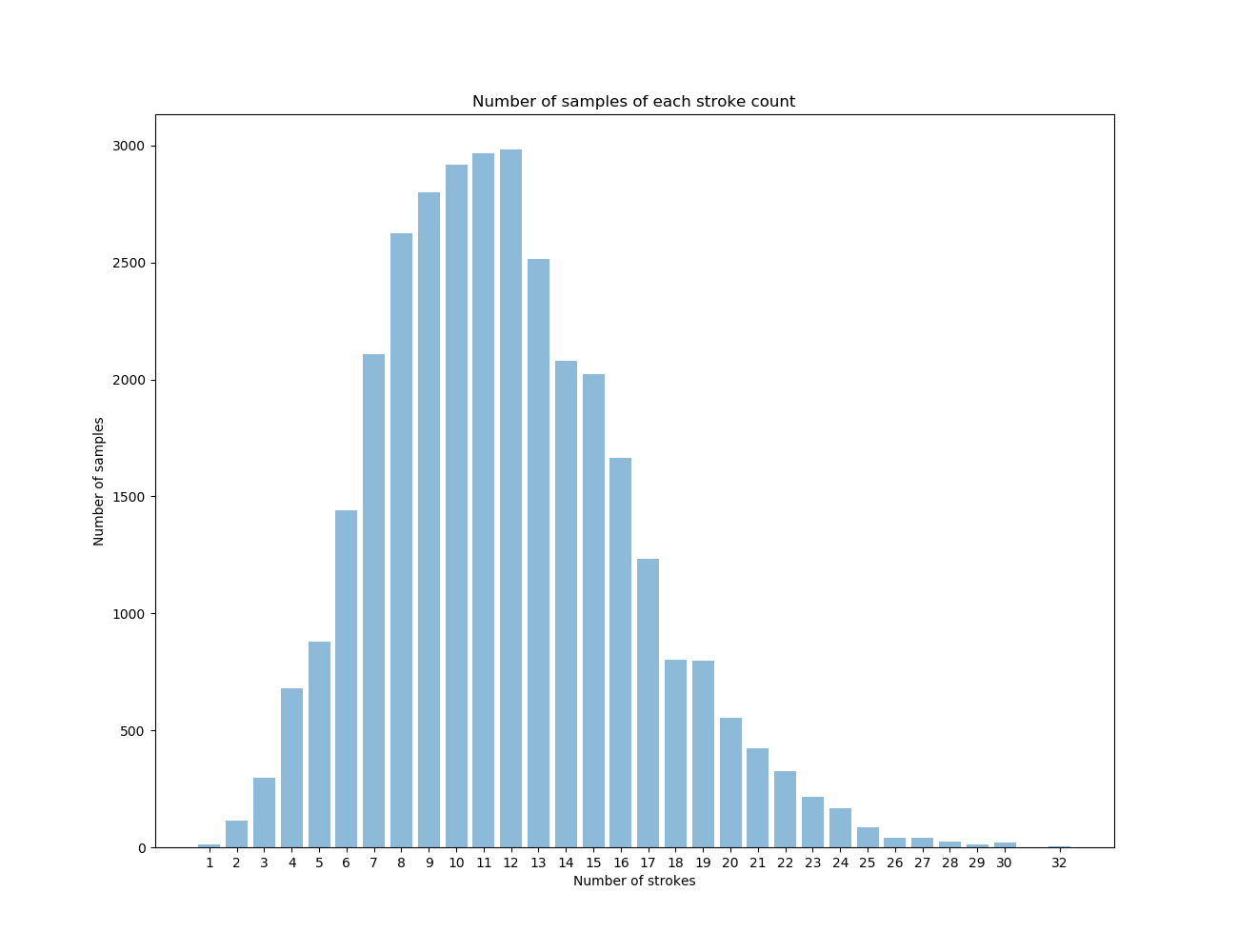
Producing this dataset had many unexpected challenges. The first was one that I am still struggling with: figuring out how to work with python. Python has many idiosyncratic methods of working that took a long time to adapt to.

The second challenge was a problem with python itself. My data production code was running up against what seemed to be a bug with python itself which would crash unexpectedly at random times with memory freeing errors. To get around this, I modified my code so I could quickly retry a character without starting over.

# Data Visualization

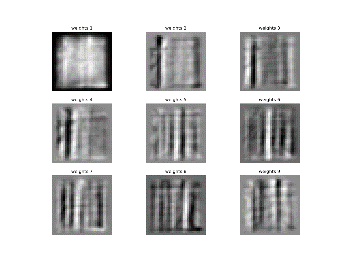
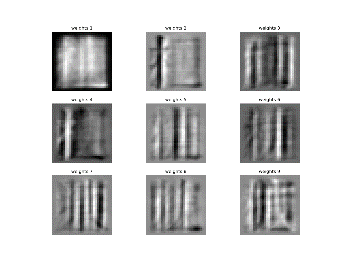
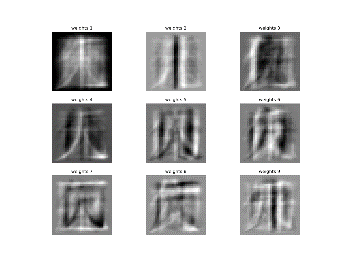
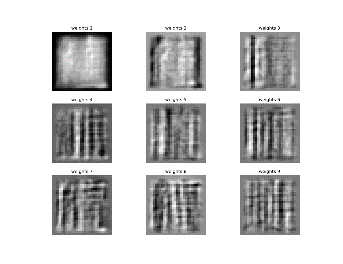
Visualizing data is very important to understand the data one is working with.

## Distribution



Above is the distribution of characters in my data set. The mode of the set is 11 with the max being 32 (籲).Because of this distribution, I expect characters with between 7 and 17 strokes to perform the best and the outliers to preform poorly.

## PCA Analysis

Above shows the top nine PCA components of the characters containing 4, 9, 11, and 18 strokes respectively[[4]](#footnote-4). We can see particular lines really popping out in the images which is what I expect a network will use to begin to classify the data.

## Challenges

There weren’t many challenges with producing the above data. If anything, the two challenges would be using python’s ploting library which is less intuitive than matlab’s, and working with the images in small enough amounts as my computer couldn’t load them all at once.

# Simple Models

## Linear

I am currently producing a linear model using tensorflow. I am close to finishing it but running into a few errors. Once this step is complete, I can generalize the step to larger models. The biggest challenge is learning how to use tensorflow in the context of this project. This step should be completed soon, progress can be seen on the [github page](https://github.com/samrjack/CharacterStrokeCount).

# Complex Models

To be completed

# Evaluation

To be completed

# Stretch Goals and Future of the Project

Depending on the initial results of the project, other goals I would like to attempt include:

* Extend the system to work with other writing systems such as Korean Hangul or Japanese Katakana.
* Classification of other attributes of a character such as radicals[[5]](#footnote-5) or its Cangjie[[6]](#footnote-6) representation.
* Extend the use of the program to develop a rudimentary OCR system.

# Inspiration

This project was inspired by the prevalent number of handwriting and digit recognition projects that are present on the internet. I have worked with Chinese and its writing system for years and feel this project is a good opportunity to try something new.

# Resources

The example I used to generate characters from a font file:   
<http://www.cnblogs.com/fengfenggirl/p/python_worddb.html>

The page I found the document for correct stroke count for any given character in Unicode. To find the file itself, I downloaded the project and extracted the file. The pearl script is unused. <http://www.mypolyuweb.hk/~sjpolit/cgi-bin/strokecounter.pl>

1. Here, Chinese refers not only to the standard dialect of Mandarin, but also to several related languages (Cantonese, Hakka, etc.) which share similar characteristics and, more importantly, a similar writing system. [↑](#footnote-ref-1)
2. In total about 50,000 characters, though only about 3000 are needed for most people. [↑](#footnote-ref-2)
3. Because this project is focusing on Chinese characters in general, both traditional and simplified characters should work as well as specialized characters from other regions such as Kanji from Japan, Hanja from Korea, or Han-Nom from Vietnam. [↑](#footnote-ref-3)
4. All the images may be found at <https://github.com/samrjack/CharacterStrokeCount/tree/76aa439f7bff6ea4b6b79e6cf193658b1c0bd7c8/data_statistics/figures> [↑](#footnote-ref-4)
5. Chinese characters share a set of fundamental components which are called radicals. There are about 250. [↑](#footnote-ref-5)
6. A way of classifying characters by component. Often used in Hong Kong as a keyboard input method. [↑](#footnote-ref-6)