**Weekly report**

Produced a short report on Tesseract-OCR. Showed Tesseract is most likely unsuitable for our needs as it seems to only work on strings of characters, rather than single isolated characters. It would probably be more work than useful trying to manipulate data to work for Tesseract.

As such, we have moved back from the idea of standardizing Chinese characters, to simply classifying them.

Produced a report on four ML methods we could employ: convolutional neural networks, wavelet transforms, eigenfaces, random forest.

All methods seem currently possible and plausible to recognise Chinese characters.

Transfer learning seems to be especially promising. One paper notes (*Ciresan, Meier, Schmidhuber – Transfer learning [2012]*) that by pre-training a neural net on only 1% of a dataset of Chinese characters and only training the last few layers on the whole dataset drastically speeds up training time as well as reducing the net’s error rate.

A brief literature search for work done on character recognition produces many varied results for which methods are used.

Converted sample .gnt files from the CASIA database into images of Chinese characters alongside each image’s true character. This provides us with ~200,000 samples to work with for now, enough to start testing ML methods.

Fig. 1 shows output from the script detailing each sample in the .gnt file and its general information. Fig. 2 shows three character images from the CASIA database corresponding to the characters '角', '饺', '缴'.

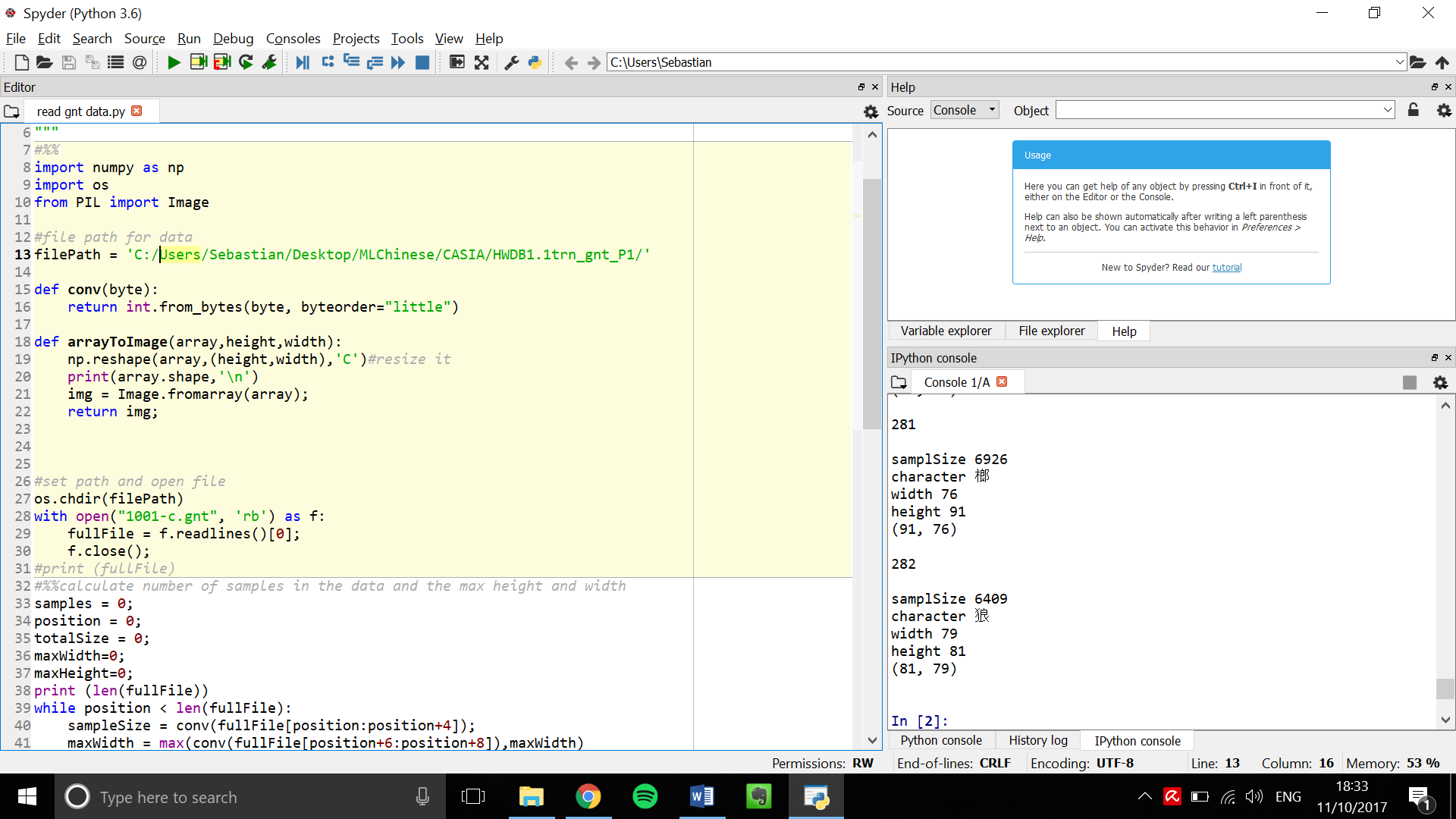


Figure : Output from one .gnt file, this .gnt file contains 282 samples.

The maximum height/width of the Chinese character images was found, and a border was added to images so that all images have the same pixel size and aspect ratio



Figure : Examples of images extracted from the .gnt files

**Action points for the next week**

1. Download the rest of the CASIA database.

2. Write a script to manipulate CASIA data – translating, scaling, rotating images to increase the data set.

3. Also to downsize images to something on the scale of 48x48 pixels (to reduce memory usage)

4. Research different thresholding techniques and their effect on optical character recognition.

5. Play around with Tensorflow and try to implement a ML method on some of the data set available.

6. Convert dataset into a text file rather than saved images (which take up much more space).

7. Loops over each .gnt file available.