OCR Handwriting Project Report

Matthew Mulhall

matthew.l.mulhall@uconn.edu

August 13, 2019

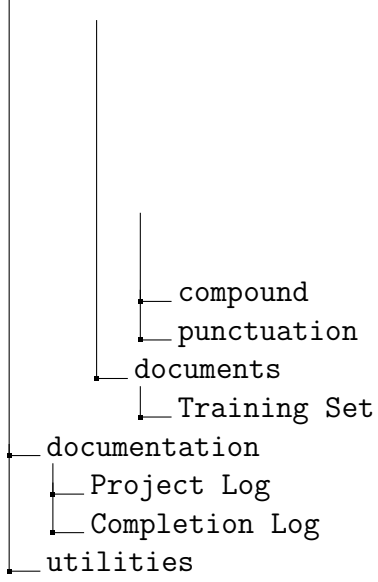
This is a report outlining what has been achieved so far for this OCR-Handwriting Transcription project for the summer of 2019 from May-August. This serves as a basis for understanding of what the project has achieved as well as a basis for writing an papers or grant proposals.

1 Technical Specifications

We currently have 7 documents that have been allocated for our project. The first 4 will be used to create the data set of images. On top of simple screenshots, we will also employ GPUs to transform the images to get the most mileage out of each photo. The last 3 will be later allocated into development and strict testing sets. These will be allocated as the training set is developed.

1.1 Description of file system

```
OCR-Handwriting
└── bin
    ├── src
    │   └── testing
    └── data
        └── char
            ├── ascii lower
            │   └── a to z
            ├── ascii upper
            │   └── A to Z
            └── ascii number
                └── 0 to 9
```



(i) Bin contains all of the 'raw' data such as images, and documents where the images come from. Each sub directory is ordered.

(ii) The section 'compound' has been added due to the nature of John Quincy Adams handwriting. There are several small phrases like 'Mr' and 'Dr' that appear more as one character than 2. This is why it is denoted as 'compound', meaning more than one letter interpreted as a single unit.

(iii) Documentation contains this document, as well as any other documents that are needed to explain the project.

(iv) Utilities contains all scripts, programs, or software that we use as a supplement in order to complete the project.

1.2 Software Information

(i) Python 3.7.3

(ii) Github

(iii) **Packages:**

(iv) Keras

(v) Tensorflow/Tensorflow-gpu

(vi) NumPY

(vii) cv2 (OpenCV)

(viii) imutils

(ix) GraphViz

(x) Matplotlib

(xi) PyDot

(xii) AutoAiLib - Library that was formed after noticing key gripes while developing the ML models. The library is a small collection of testing classes, data manipulation classes, as well as pure functions that make creating ML easier.

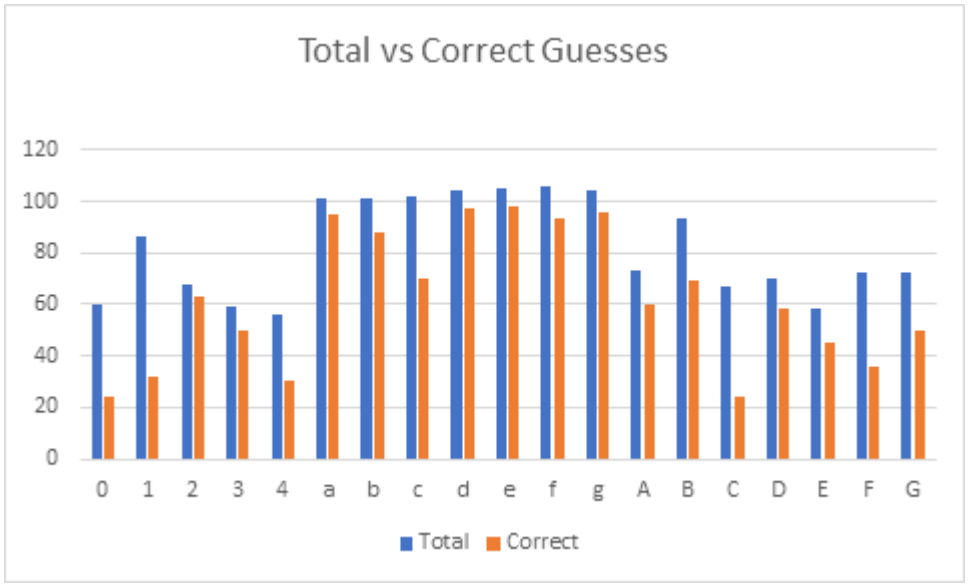
2 Models

Over the course of the summer the project has accumulated 41 models from the time of writing this report. Almost all models built upon the previous successful model and furthered that sucess. Even with our minimal amounts of data we have been able to achieve 96% + success rate for 4 way identification, and 86% + accuracy on 22 class identification.

3 Data

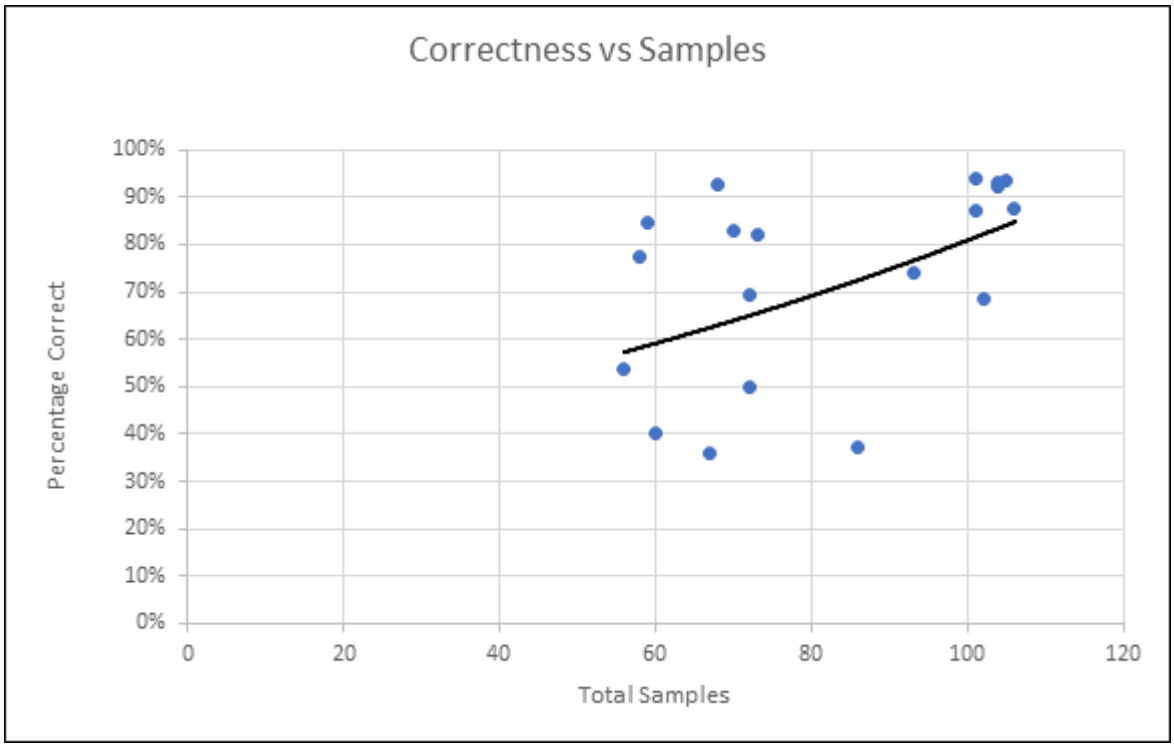
This project has amassed 16,000+ images of characters throughout the course of several months This averages out to 260 images per class. Historically speaking this is a very low amount of representative data for convolutional neural networks. Despite this, we have been able to get good results out of our models. Even so, I posit that to make enterprise level sofftware we will require data increases by a factor of 3-5x. We completed tests earlier in order to see what type of correlation we found between a class' sucess rate when plotted against its data size. The following is a graph showing the total number of samples vs correct guesses that was achieved during a test on the 7th iteration of the medium set model.

(i)



(ii) The following graphic used the previous graphs data to find the relationship between sample sizes and correctness.

(iii)

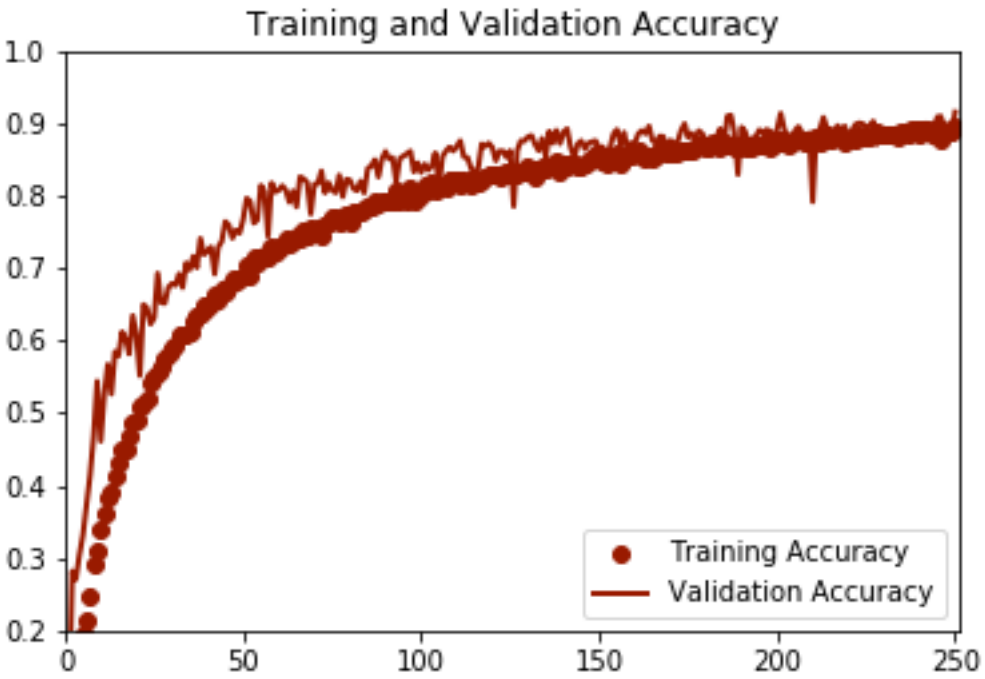


Given the conclusions of the tests as well as plotting the information into graphs, the model needs more data. The previous chart's trendline clearly suggests that

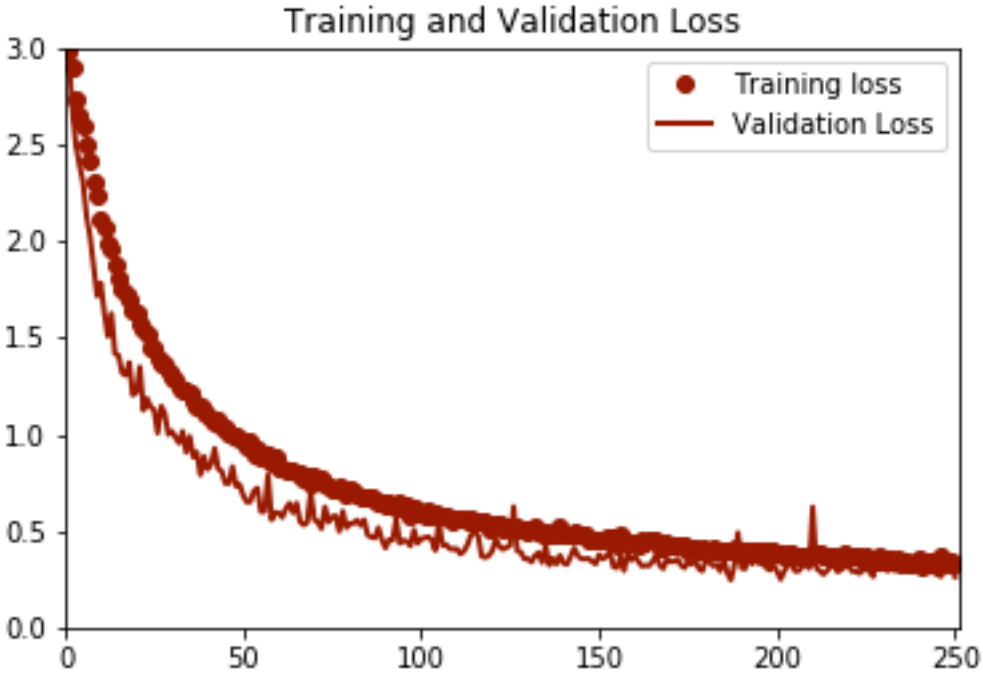
there is a positive relationship between samples taken and the model's success. Although this seems obvious, it was important to isolate each character, as well as rule out the interaction of certain characters. I.e. it was important to figure out if 'b' and 'd' interacted, or if they had high success despite their similarity. However, we have seen lower success than usual for 'b' in some tests, most likely due to focusing on smaller feature extraction in those models. From what I can deduce from our tests, it is useful to have strides be centrally distributed, meaning: 1-2 (1,1) , 1-2 (5,5), 1 (7,7), and mostly (3,3). This seems to be a winning combination in these CNNs. This is most likely due to the fact that the features between each image that are different are not large or small, but 'medium' as strange as that language sounds. This distribution however gives our model the opportunity to pick up on all possible differences and train for them.

4 Testing

(i) The truncated test from yesterday led to the following results:



(ii)



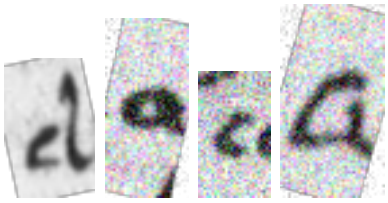
(iii)

(iv) The results are good, but still seem inconclusive. It does not appear clear that the lag in progress from previous iterations was due to a lack of data, although I think adding data would certainly improve scores.

(v) In order to gauge the effect of increasing size relative to a single sample rather than a change in sample classes (as was done in the previous test), we chose to expand a data set through manual augmentation and comparing it to the same, unaugmented set.

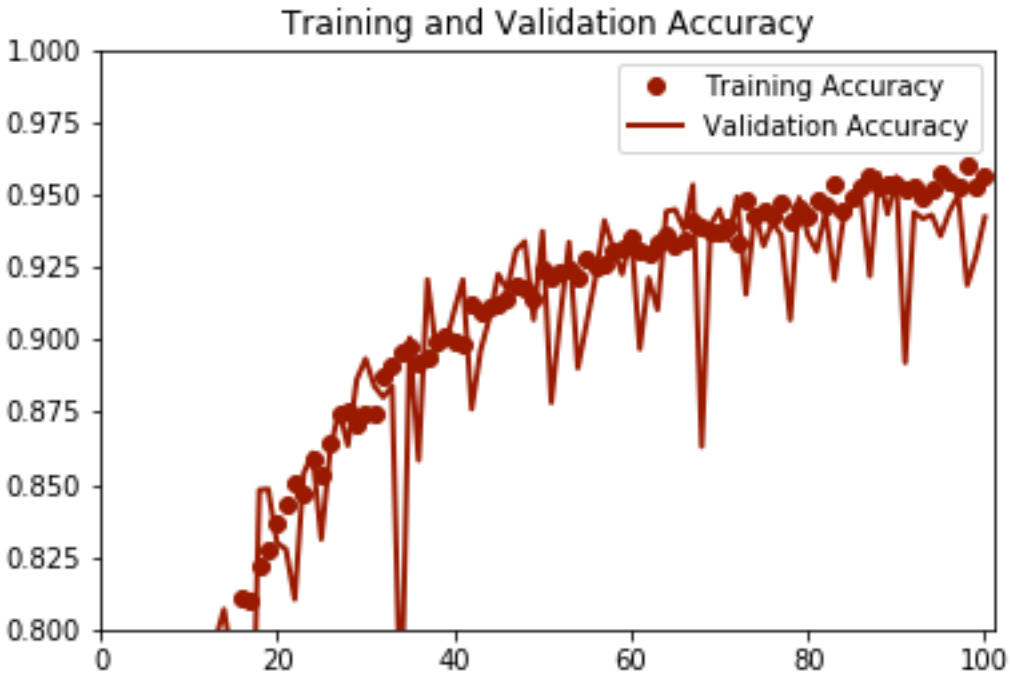
(vi) We went back to the original smallset test, which has 4 classes with 1,539 images. Through a program called image transform that I wrote, I augmented them and got 3,053 images total from the base set.

(vii) Examples of augmentation:

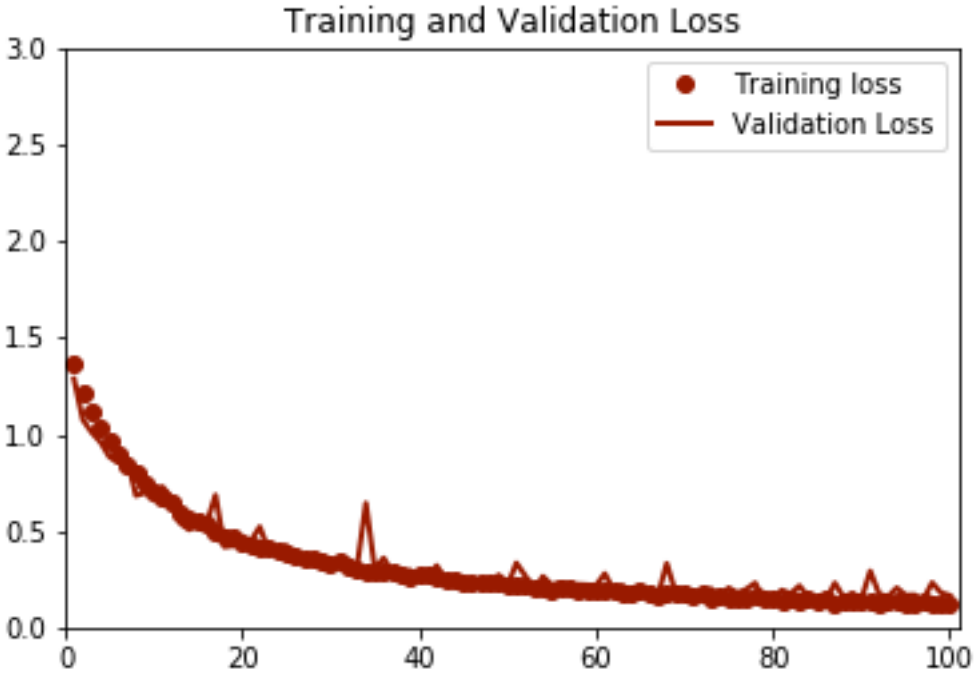
(viii) 

(ix) The results were as follows:

(x) Baseline:

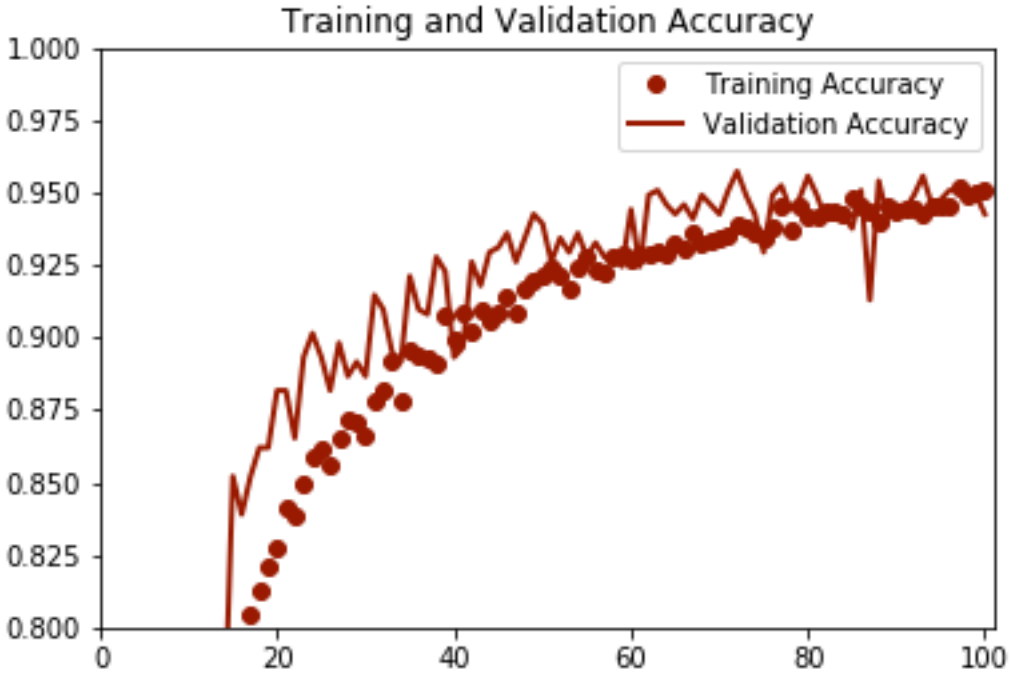


(xi)

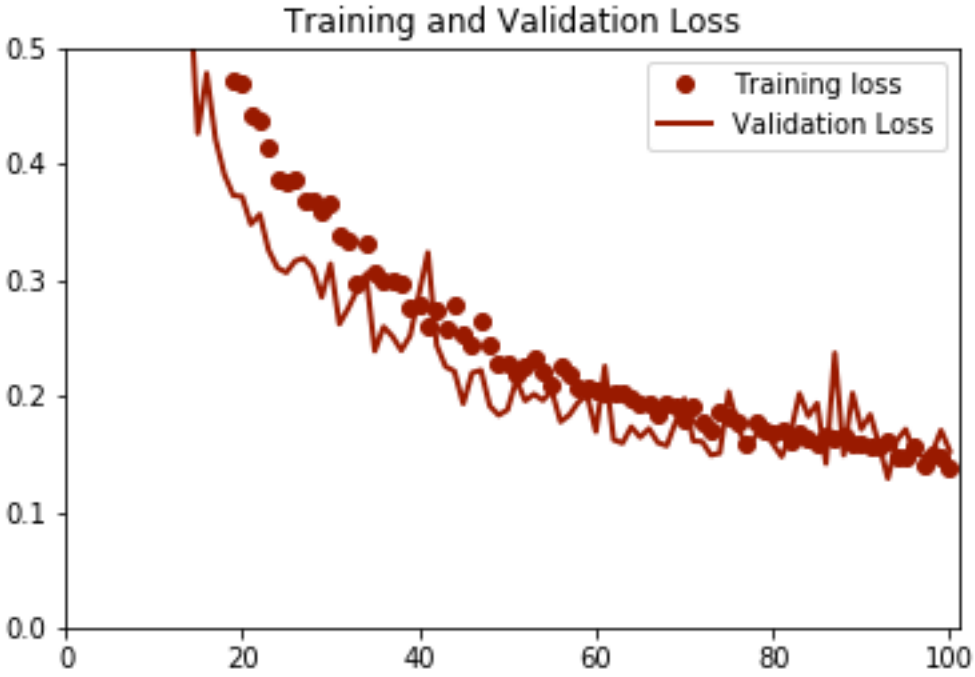


(xii)

(xiii) Test:


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0
20
40
60
80
100

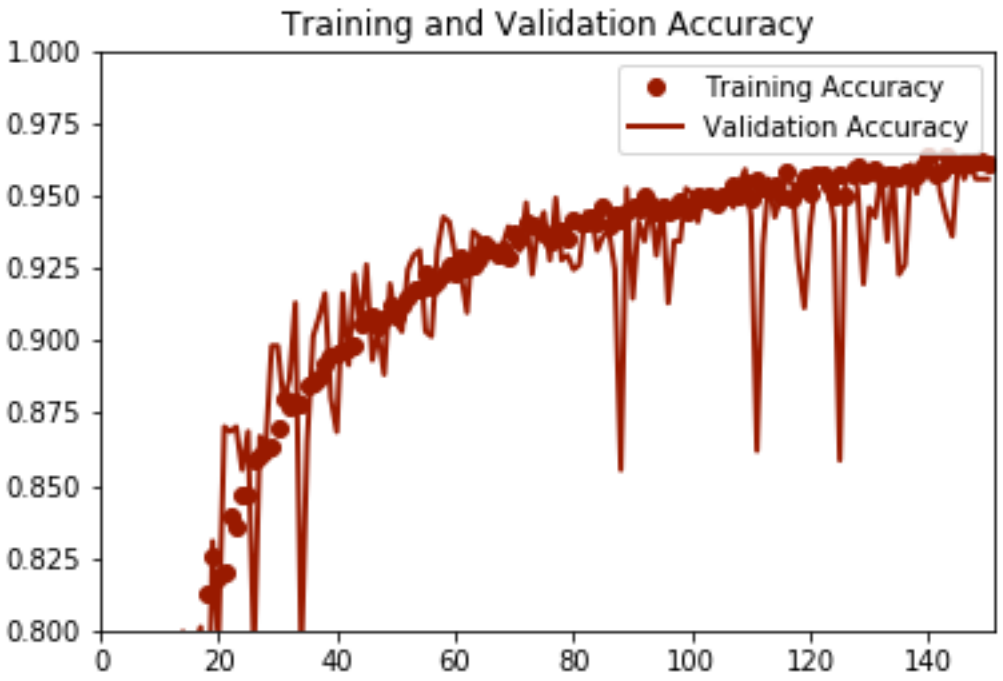
(xiv)


Training and Validation Loss
Training loss
Validation Loss
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100

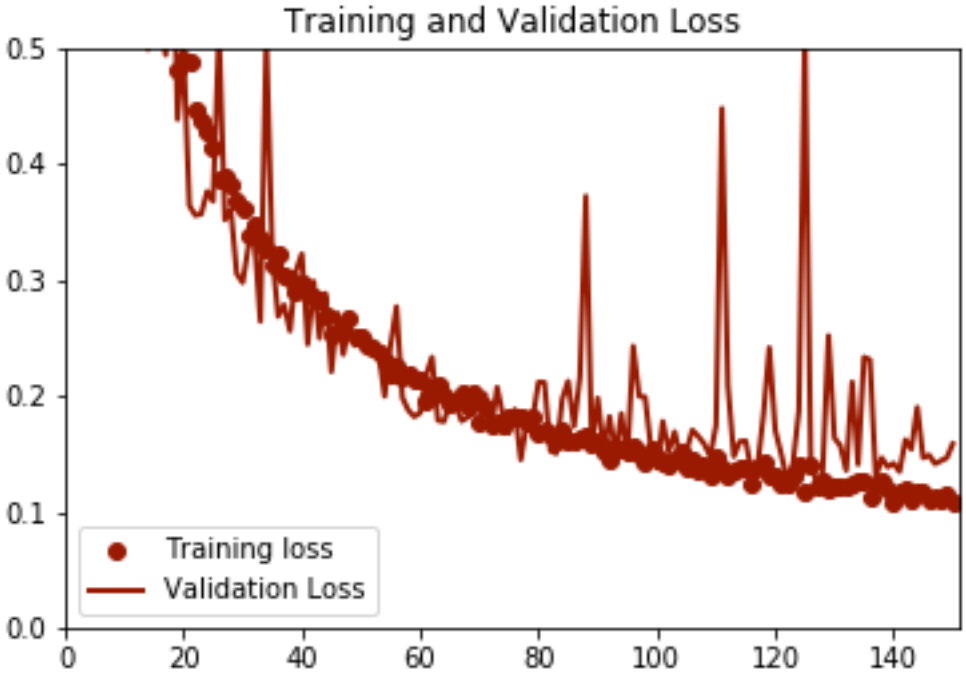
(xv)

(xvi) The results are not significant, although the decrease in loss is notable. As can be seen in Figure XIV there appears to be a small uptick in accuracy at the end of the training cycle. We intend to test what happens with slightly increased training to 150 epochs.

(xvii) Results:



(xviii)


Training and Validation Loss
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100
120
140
Training loss
Validation Loss

(xix)