OCR Handwriting Project Outline

Matthew Mulhall

matthew.l.mulhall@uconn.edu

July 25, 2019

1 May 14, 2019

1.1 Summary of Design Decisions

The project will follow an abstraction based design: letters, words, lines, and entire documents. Every document can be broken down into these respective groups of abstraction.

(i) An entire document.

(ii) A collection of lines in a document.

(iii) A collection of words that are consecutively placed on each line.

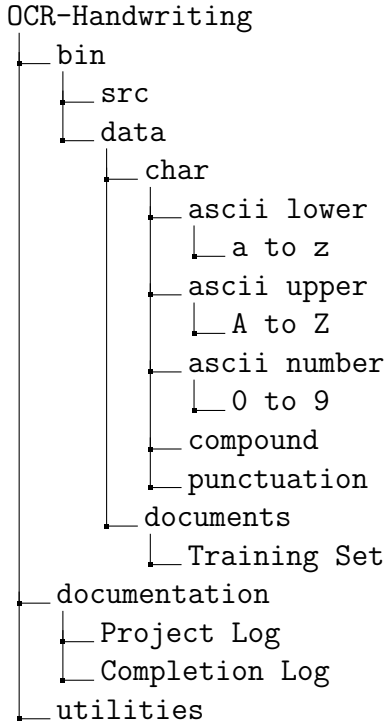
(iv) Single characters that make up the words.

It can be seen that each level abstraction relies on the previous, going all the way down to the individual letters that are on the document. Given the nature of that abstraction Dr. Johnson suggested we start from the ground up, meaning first we will be building the data set for letters, and training a model to recognize other letters of similar (1800's English) style. Our current priority is to build this large data set of characters for our neural network to pull from. After this set is built up we will work on figuring out the optimal design of our model and start to train it. After this section is completed we will have a network that can identify individual characters. From this base level we will then work on the next level of abstraction, that will be able to identify the words in a line. The project will follow a similar style of abstraction based progress until we can use every level to read an entire document.

1.2 Some specifics

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.

1.3 Description of file system



(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.

1.4 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

1.5 Significant Developments

1.5.1 May 14, 2019

Total images taken: 296

Matt created a python script that renames the pictures in the subdirectories according to a naming scheme, this allows for saving files without having to worry about typing the name into the save box. Doing this means the whole process takes 10x less time. When taking photos one can either: focus on a letter saving several in a certain directory (fastest), or save all photos to a "dump" folder and place them afterwards in their correct directory.

1.5.2 May 15, 2019

Letters completed: 'a' , 'e' .

Total images taken: 2,075

Process for quickest imaging (Modified 5/21):

(i) Pick a letter that has a lower than needed sample size (¡ 500).

(ii) Using Lightshot, take a screenshot of a letter, and click the save button. Navigate to the respective directory and save.

(iii) For all subsequent letters, take the photo and use shortcut CTRL-S and it will auto-save to the same directory.

(iv) After you find as many as you can on the page, or several pages, move on to the next page.

(v) After around 125-150 letters from a given set of pages, go onto another set of pages.

(vi) Before pushing to git, run the renameUtilityScript.py file which will rename all of the files to the appropriate schema.

1.6 May 16, 2019

Letters completed: 'b'.
Total images taken: 864

(i) Made changes to the script so that it can be run on any machine without needing to edit the path in the file. If anybody wants to run it, python must be installed and they can either manually run it or write a bat file. There is a provided bat file skeleton, all that needs to be added is the path to the .py utility, and it can be run from anywhere.

1.7 May 21, 2019

Letters completed: 'c', 'd', 'i', 'f', 'g', 'h'
Total images taken: 3,054

(i) Purchased the font "Old Man Eloquent" that we will use to diversify our samples. The current plan is to photograph the

font in various contexts and use CUDA to transform the images to extract a large amount of diverse images from one example.

(ii) Mike and Matt had a conversation outlining the plan for hardware to be able to transform images. Once the types of image transformations are chosen, Matt will create software that will be able to be used without programming experience.

(iii) Matt suggested using an image normalization algorithm to give each image the same scale. It would involve locating the global min and max for width and height, and setting each photo to those dimensions. This could be important for making sure the network does not pick up on unintended scale related differences between letters.

1.8 May 22, 2019

Letters completed: 'j', 'l', 'm', 'n'
Total images taken: 1,928

(i) Matt created a completion log complete with all characters so that we can more easily keep track of completed characters.

1.9 May 23, 2019

Letters completed: 'k', 'o', 'p'
Total images taken:1,004

(i) No important developments today. Good progress on imaging.

1.10 May 28, 2019

Letters completed: 'q', 'r', 's', 't', 'u'
Total images taken: 2,244

(i) Today we completed 5 characters towards the end of the alphabet. I have high confidence that by tomorrow we will complete all lowercase imaging. This means we are slightly under halfway to completing the data set

(ii) Another important development is that we passed 10,000 images in just 7 working days, a great achievement. We are currently working with: **11,465 images.**

1.11 May 29, 2019

Letters completed: 'v', 'w', 'x', 'y', 'z'
Total images taken: 1,924

(i) Today I completed 5 characters. Additionally, with the completion of 'z' the entire lowercase alphabet has been concluded. I would venture to say we are likely half way currently. Although there are more than just the upper-case section left, those are far less common and hence will include fewer screenshots.

1.12 May 30, 2019

Total images taken: 1,001

(i) Today I worked from home, the log is being updated retroactively.

(ii) The most important development from the day was a change in the way we collect screenshots for the upper case characters. Rather than picking a character and moving through each page for that character, for upper case (or generally less frequent sets of multiple characters) find all examples on the page and screenshot them. After this, move onto the next page.

(iii) Because the amount of upper case characters is far lower than lower case letters, we are just taking as many samples as we can get. Therefore there will be no more "letters completed." Upper case letters will be completed when all examples of them are recorded

(iv) Finished up to page 17 of DJQA 1829-02.

1.13 June 4th, 2019

Total images taken: 1,495

(i) Today was a standard day, continued to image the upper case letters.

(ii) There was quite a lot of progress in terms of images taken as well as total traversal of the data set. We are almost half way through the data for upper case letters. This puts us in a great position, well over half way done with overall imaging.

(iii) Finished up to page 16 of DJQA 1829-03.

1.14 June 5th, 2019

Total images taken: 1,247

(i) Not many important developments today, the only noticable change would be that as I have progressed, the density of upper case characters has seemed to dwindle, as can bee seen from the total for today.

(ii) Finished up to DJQA 1829-04 section 17. Meaning we got just over 30 pages of material today, exactly the same as the previous 2 days of upper case imaging.

1.15 June 6th, 2019

Total images taken: 936

(i) Completed up to DJQA 1829-05 section 19. On the road to complete upper case before lunch of the next work day.

1.16 June 11th, 2019

Letters Completed: All capital letters besides 'O', 'L', 'X', 'Z'.
Total images taken: 1,077

(i) Capital letters are virtually completed, all that is left is specfic imaging to expand the sets of several select capital letters.

(ii) Imaging has moved onto compound letters, completed through DJQA 1829-03 Section 20.

1.17 June 12th, 2019

Letters Completed: 'Mrs', 'Mr', 'Dr', 'ss'
Total images taken: 750

(i) Completed all compound letters.

(ii) After completing compound letters, we moved onto completion of imaging the numbers.

(iii) Imaging of numbers reached DJQA 1829-03 section 24.

(iv) Also did a lot of reading that is necessary for the continuation of the project.

1.18 June 13th, 2019

Total images taken: 750

(i) Continued imaging of numbers for numbers, currently on DJQA 1829-04 section 19.

1.19 June 18th, 2019

Letters Completed: ':', comma, period, question mark, exclamation mark.
Total images taken: 800

(i) Significant imaging of the punctuation set.

(ii) The last 2-3 hours were spent on the first iteration of proof of concept.

(iii) This proof of concept will focus on making a convnet that properly identifies 4 of our letters with using minimal data altercations. After this we will move to iteration 2 where we simply alter the data.

(iv) After 2 we will most likely open up to more letters and continually scale and test the software.

(v) Phase 1 of the testing was completed. With a data set totaling 2,000 images in 4 categories (namely 'a', 'b', 'c', 'd') Matt constructed a convolutional neural network that has a peak of 94.69% validation accuracy. Note, the network did not use image preproccesing to change the size of the dataset, which will be used in test 2.

(vi) The network form that was used in this exercise was as follows:

(vii) Model: Sequential

(viii) Layer 1: Convolutional 2D

(ix) Layer 2: Max Pooling 2D

(x) Layer 3: Convolutional 2D

(xi) Layer 4: Max Pooling 2D

(xii) Layer 5: Convolutional 2D

(xiii) Layer 6: Max Pooling 2D

(xiv) Layer 7: Convolutional 2D

(xv) Layer 8: Max Pooling 2D

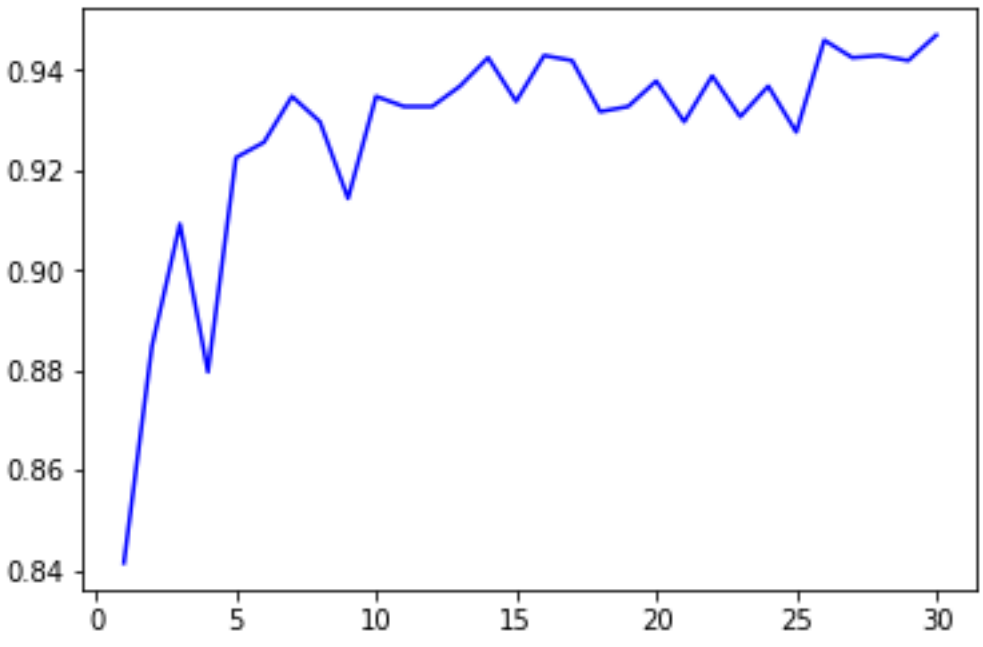
(xvi) Layer 9: Flatten

(xvii) Layer 10: Dense (512)

(xviii) Layer 11: Dense(4 - categories)

(xix) Below is an image that depicts the 'learning' of the model from test 1.

(xx)



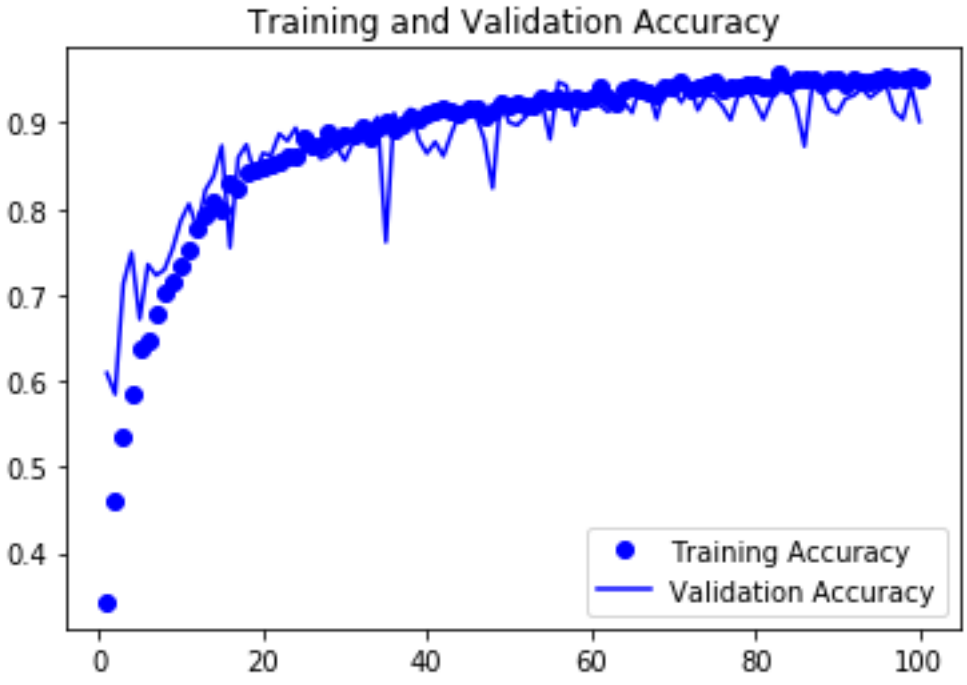
1.20 June 19th, 2019

(i) Today we began with phase two of the small set convnet. The changes will be as follows:
Steps per epoch up to 100, data augmentation (specifics provided in the code).

(ii) We completed all images today. The total count being 21,515 images. There are still some select images that I believe we will need to address, but this will happen as we continue our push forward in testing.

(iii) After completing the training of the new neural network that used augmentation we got the following results:

(iv)



(v) As we can see, the tests did well to limit overfitting. However, further tuning is required to get our percentage even higher. Currently it lags at around 96%

(vi) Today I created several important scripts for the project. The first is a utility to automate the entire prediction process of any model. In this file you give a path to an image and the program will predict, and create an image with text that displays the guess and its percentage certainty.

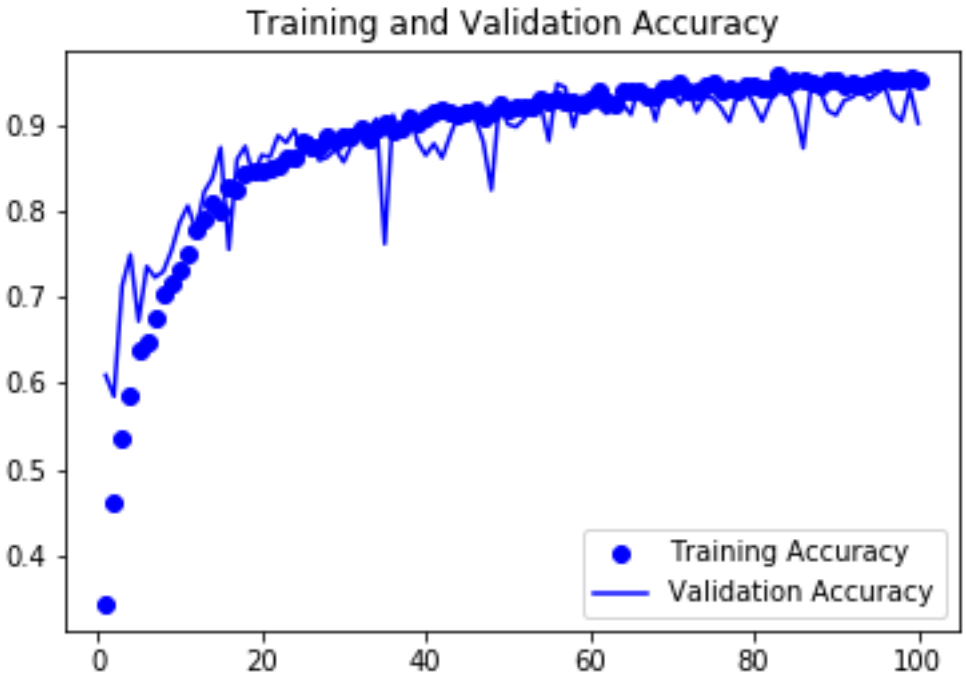
(vii) 

(viii) This image shows an example of our second network trained with augmentation predicting a letter 'd'. The catch is, this was written by a different author than our training set!

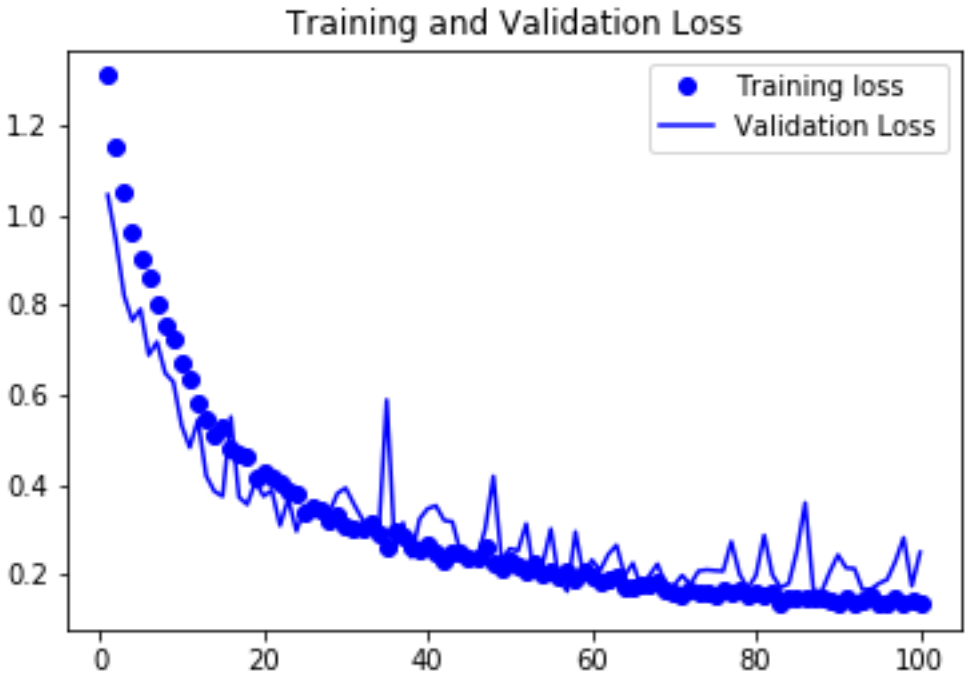
(ix) The second script was a utility for moving through directories and placing photos in respective test/train/validation directories. This is much easier for med-large sets, rather than placing each image manually.

1.21 June 20th, 2019

(i) Today I was able to do several things, first and foremost get better plots of the smallset test #2.

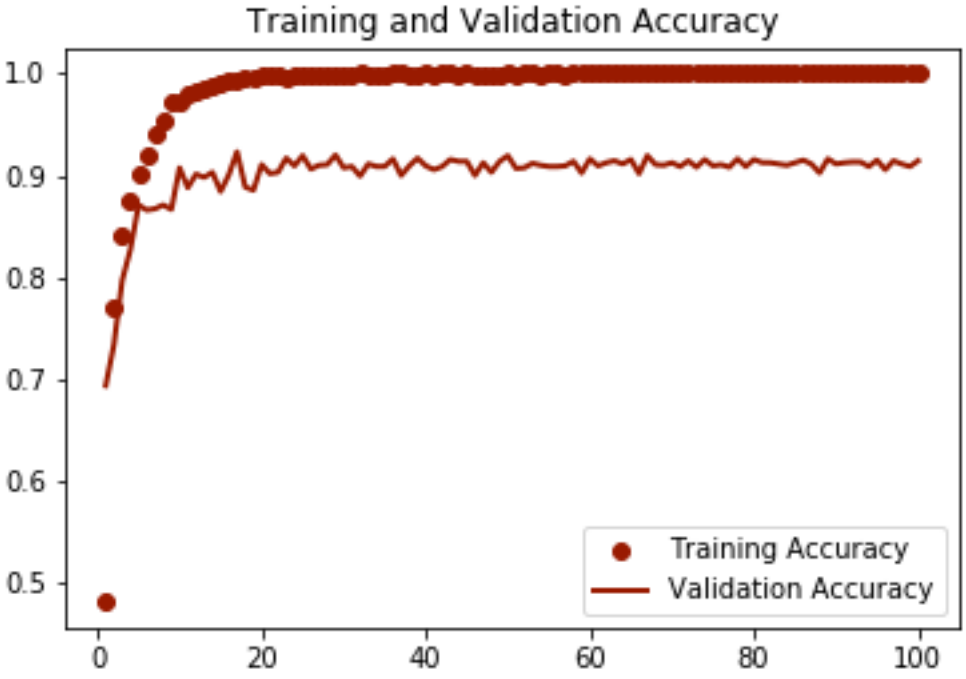


(ii)

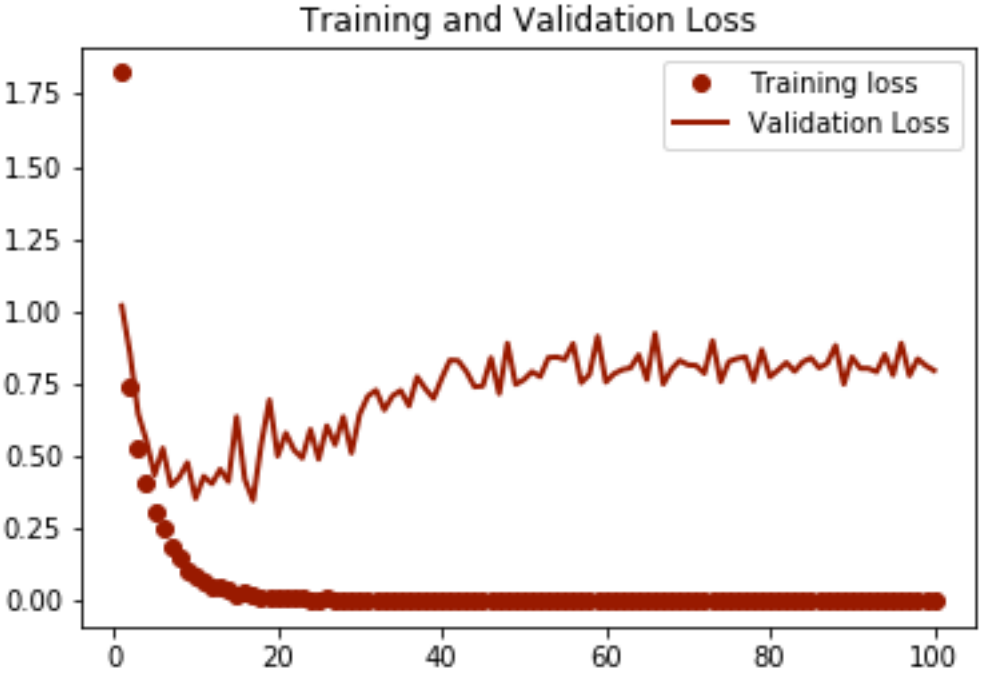


(iii) As we can see from testing, this second network design, which includes augmentation, has far less over-fitting. This was to be expected, but the results are quite impressive. I suggest that for this second small set model we try varying epochs, and new layers.

(iv) Secondly I was able to finish the training of the med-set network, which did not turn out as we would have hoped. But not to be cynical, we will be able to fine tune this.


Training and Validation Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
Training Accuracy
Validation Accuracy

(v)


Training and Validation Loss
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
Training loss
Validation Loss

(vi) Although, on first glance, it appears that the validation accuracy was pretty good (although heavily over-fitted), yet the loss remained above 75% for the duration of the training. I think that this means a few things, firstly we need to make the network deeper and use more advanced **layer techniques**. Secondly, adding image augmentation will certainly take a lot of that loss, and over-fitting away. These augmentations to the model will happen next Tuesday.

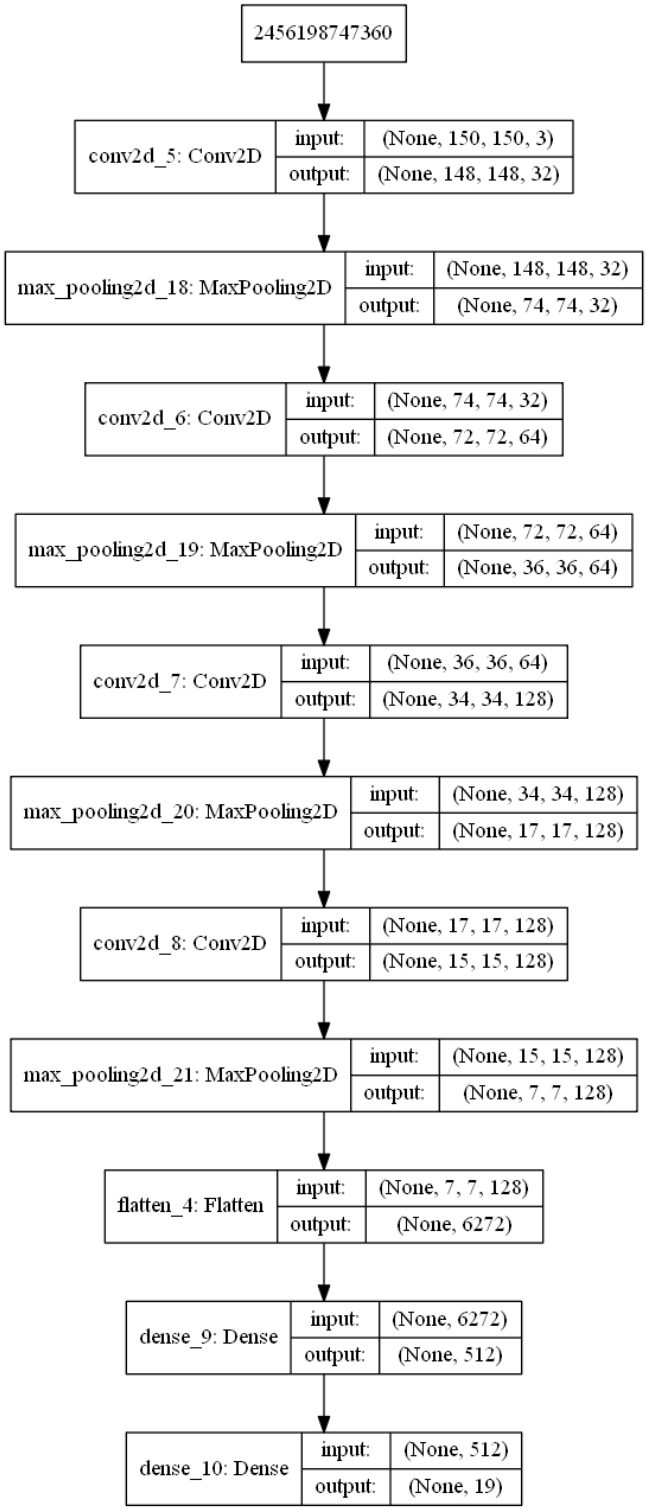
1.22 June 25th, 2019

(i) Based on the poor showing of the previous model, to start the day I will try and modify it in a second medium test. This test will aim to bring the loss as low as possible.

(ii) Rather than continually attempting to adjust layers, I believe it would be intelligent to do the changes in batches and closely analyze the difference between each method. First we are adding data augmentation to the second test. After, we will move onto layer changes.

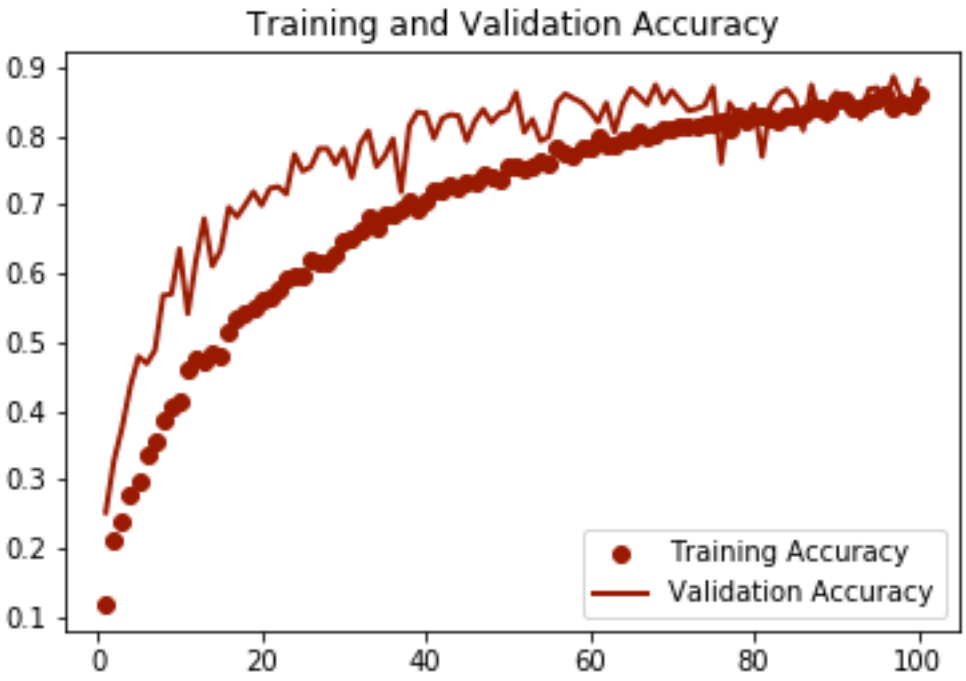
(iii) **Today I switched work stations to a computer with a GTX 1080, because of this, training time has decreased by 90%.**

(iv) The following is the model design used in test 'medset-2', this model included augmented training images, defined by the parameters available on github.

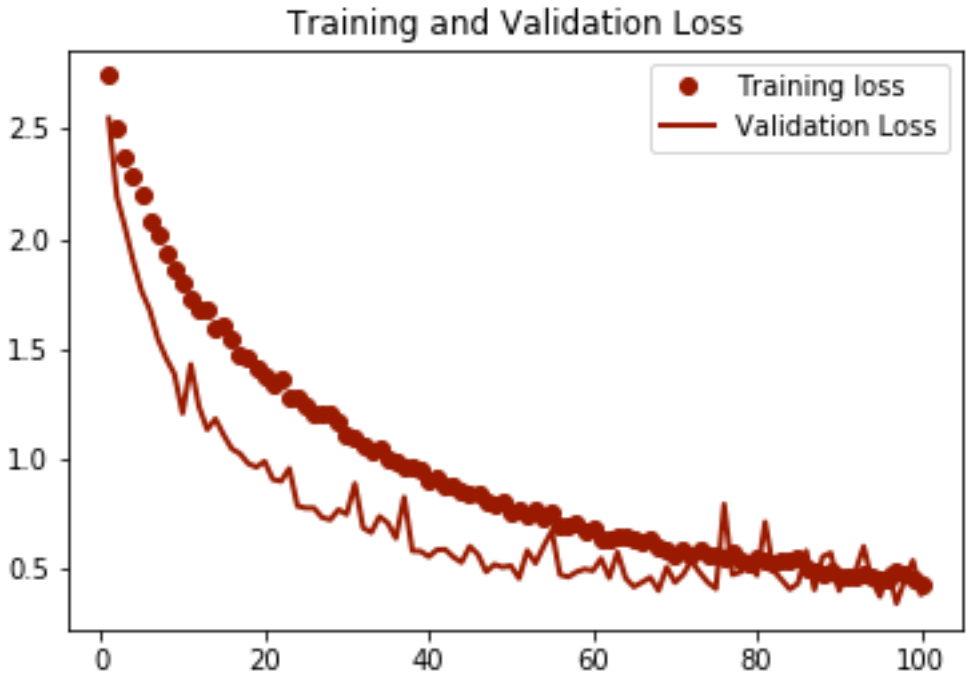

2456198747360
conv2d_5: Conv2D
input: (None, 150, 150, 3)
output: (None, 148, 148, 32)
max_pooling2d_18: MaxPooling2D
input: (None, 148, 148, 32)
output: (None, 74, 74, 32)
conv2d_6: Conv2D
input: (None, 74, 74, 32)
output: (None, 72, 72, 64)
max_pooling2d_19: MaxPooling2D
input: (None, 72, 72, 64)
output: (None, 36, 36, 64)
conv2d_7: Conv2D
input: (None, 36, 36, 64)
output: (None, 34, 34, 128)
max_pooling2d_20: MaxPooling2D
input: (None, 34, 34, 128)
output: (None, 17, 17, 128)
conv2d_8: Conv2D
input: (None, 17, 17, 128)
output: (None, 15, 15, 128)
max_pooling2d_21: MaxPooling2D
input: (None, 15, 15, 128)
output: (None, 7, 7, 128)
flatten_4: Flatten
input: (None, 7, 7, 128)
output: (None, 6272)
dense_9: Dense
input: (None, 6272)
output: (None, 512)
dense_10: Dense
input: (None, 512)
output: (None, 19)

(v)

(vi) This model performed much better than the previous iteration due to augmentation. Albeit, still below our goals. The following graphics show that this model did much better dealing with overfitting:



(vii)



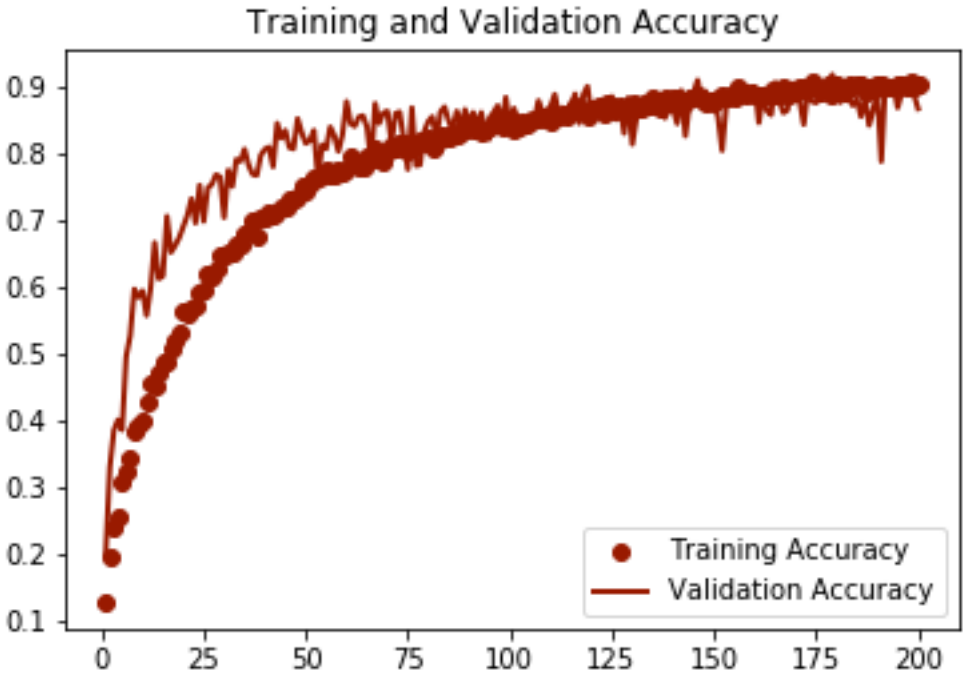
(viii)

(ix) Some other important takeaways: Loss was kept far lower, decreased by $30\%^{+}$ relative to the previous test. Accuracy remained lower than previous attempts at 85%, but in this context the lower accuracy is not a negative as it coincides with a falling loss value.

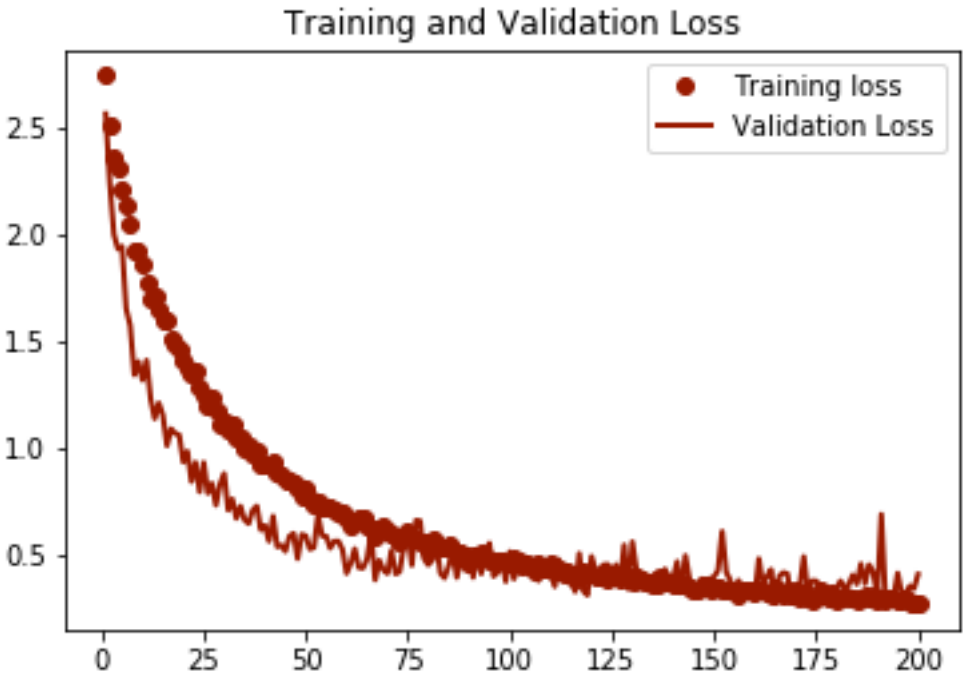
(x) After feeling confident about the network structure in test 'medset-2' I continued onto test 'medset-3' to try and further the progress. The only difference for this test (medset-3) was doubling the epochs. I decided to make this choice because the network seemed to be continually improving, rather than reaching a plateu before 100 epochs. Continuing the training further has seemed to do a great deal of relative improvement on both loss and accuracy. The next test 'medset-4', will have to focus on continuing with this and perhaps extending even further if a plateu is not reached in the previous test.

(xi) Although the results are not perfect at the moment, in a mere day we have been able to take loss from .7 in medset1 to .28 in medset3.

(xii) As can be seen from the two following graphs, increasing epochs did translate to an increase in accuracy and decrease in loss (90% and 28% respectively). This bodes well for the future, we can use this to further explore techniques of training as a road to better results as well as hyper parameters.



(xiii)

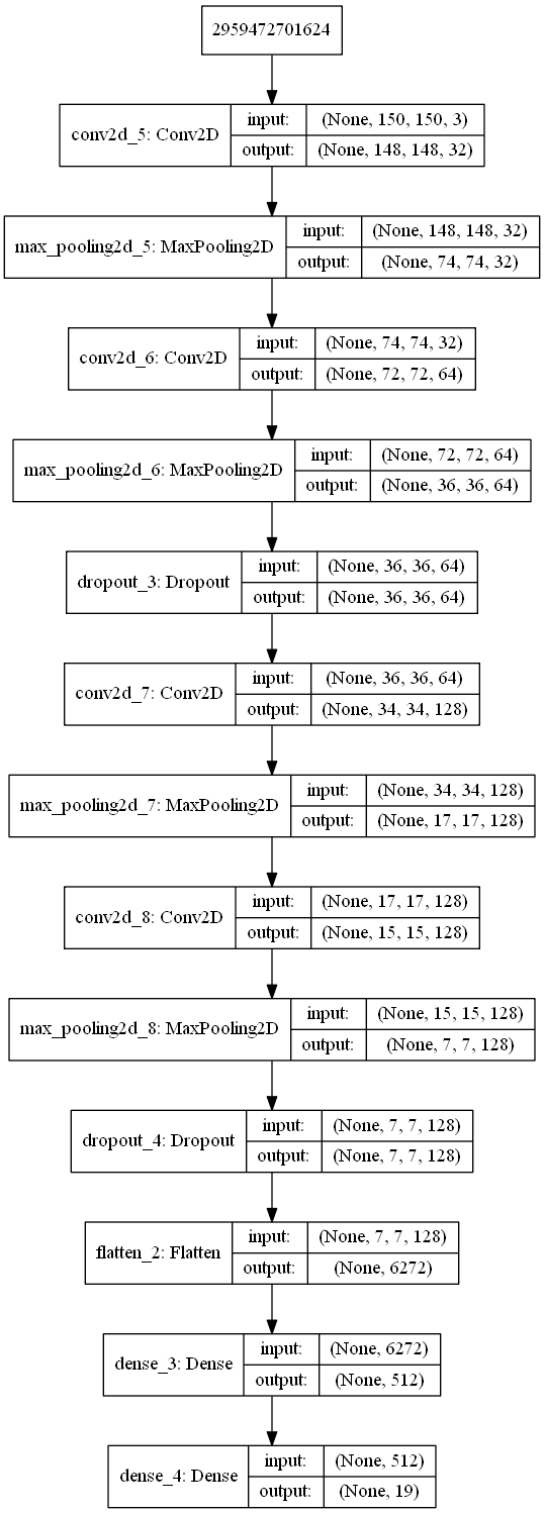


(xiv)

1.23 June 26th, 2019

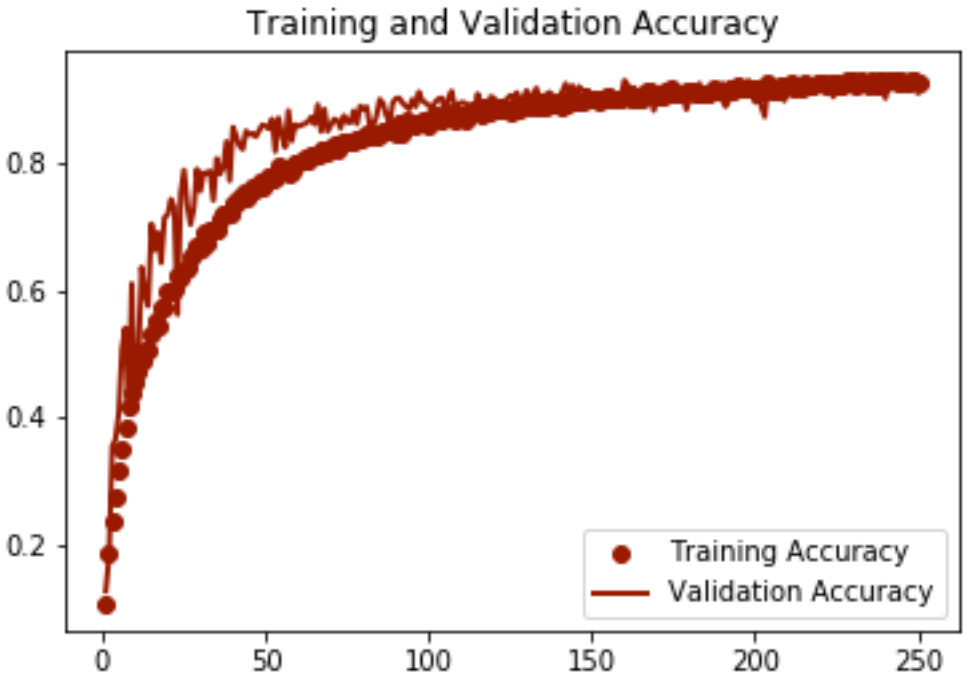
(i) Today I decided to train another model on the medium test set. This model will have several differences and similarities. It will be trained longer (250 epochs), have 64 for batch size, and will also have two dropout layers added. This model, denoted as medset4, will have the following form:

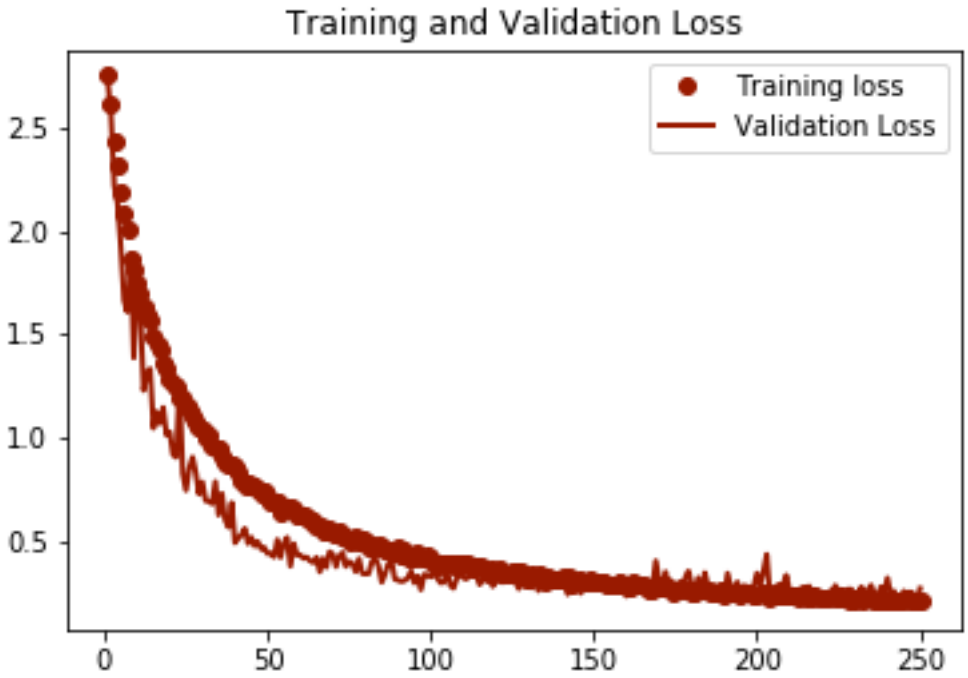
(ii)


2959472701624
conv2d_5: Conv2D
input: (None, 150, 150, 3)
output: (None, 148, 148, 32)
max_pooling2d_5: MaxPooling2D
input: (None, 148, 148, 32)
output: (None, 74, 74, 32)
conv2d_6: Conv2D
input: (None, 74, 74, 32)
output: (None, 72, 72, 64)
max_pooling2d_6: MaxPooling2D
input: (None, 72, 72, 64)
output: (None, 36, 36, 64)
dropout_3: Dropout
input: (None, 36, 36, 64)
output: (None, 36, 36, 64)
conv2d_7: Conv2D
input: (None, 36, 36, 64)
output: (None, 34, 34, 128)
max_pooling2d_7: MaxPooling2D
input: (None, 34, 34, 128)
output: (None, 17, 17, 128)
conv2d_8: Conv2D
input: (None, 17, 17, 128)
output: (None, 15, 15, 128)
max_pooling2d_8: MaxPooling2D
input: (None, 15, 15, 128)
output: (None, 7, 7, 128)
dropout_4: Dropout
input: (None, 7, 7, 128)
output: (None, 7, 7, 128)
flatten_2: Flatten
input: (None, 7, 7, 128)
output: (None, 6272)
dense_3: Dense
input: (None, 6272)
output: (None, 512)
dense_4: Dense
input: (None, 512)
output: (None, 19)

(iii) This model performed very well. Pushing our loss further to a cool 21% and accuracy up to 93%.

(iv) The models performance can be seen here:





(v) The following test, medset5, will continue the progress of 4. Primarily we will introduce batch normalization to the model.

1.24 July 5th, 2019

(i) The gap in timing for updates has been due to the stall in model improvements over the last week.

(ii) As we continue to push forward in our efforts, we will most likely see that once we reach a certain point, the gains will take longer to achieve.

(iii) We are currently still on the medium test set, 19 images, and will continue to develop the model until we feel that our performance is as high as we would like. The work that is going on now is heavily tuning the hyper parameters of our most successful

model so far. The hope is that by varying these parameters, we can find an optimal model for our next test which will involve a larger set of characters.

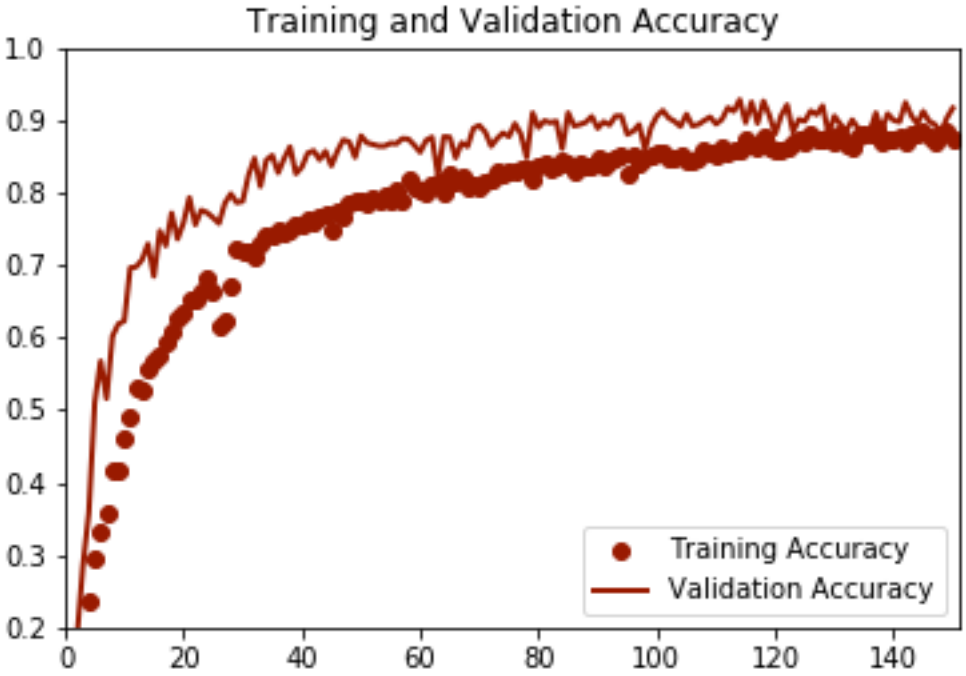
(iv) Most of these tests have yielded sporadic results. None of them were awful, but none of them yielded better results.

(v) The following has been tried so far: Increasing dropout, decreasing batch size, adding batch normalization, adding convolution layers.

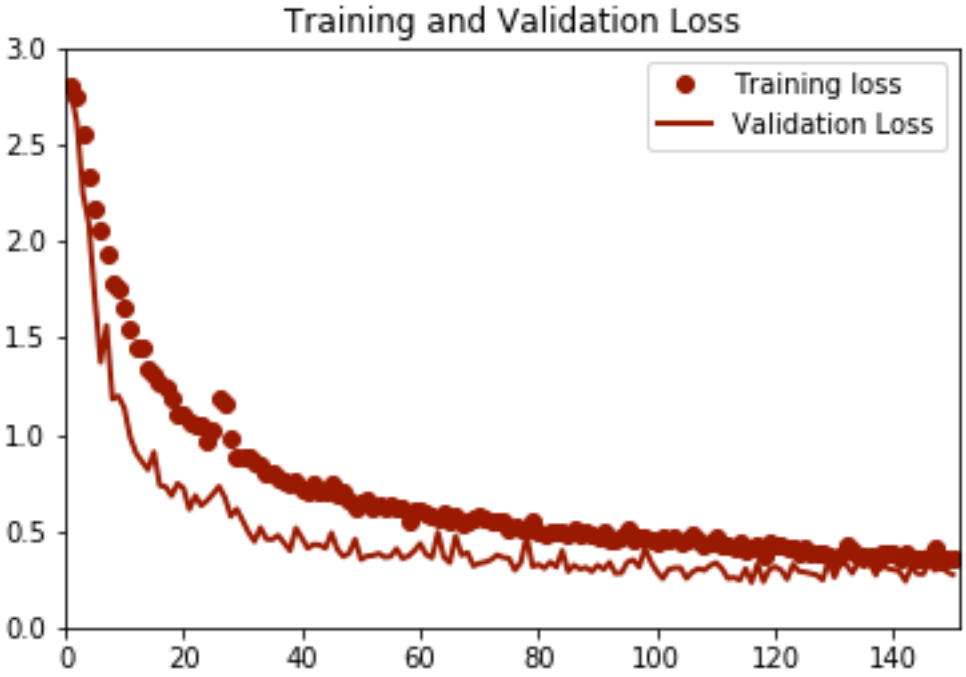
(vi) So far it seems like the road ahead will consist of doing this tuning for several weeks and seeing what we can get from it. I would estimate that if we can't get further than 93% after a month or two then we will need to go back and gather more data, which very well may be necessary.

(vii) I am currently running the last training test of the day, and am hopeful that this will provide a useful direction to take the model.

(viii) The results are as follows:


Training and Validation Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
120
140
Training Accuracy
Validation Accuracy

(ix)


Training and Validation Loss
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
20
40
60
80
100
120
140
Training loss
Validation Loss

(x)

(xi) The results are preliminarily good, and I believe that we will be able to work to improve this version.

1.25 July 15th, 2019

(i) After two weeks of rather stagnant improvements in test results, I decided to test a hypothesis.

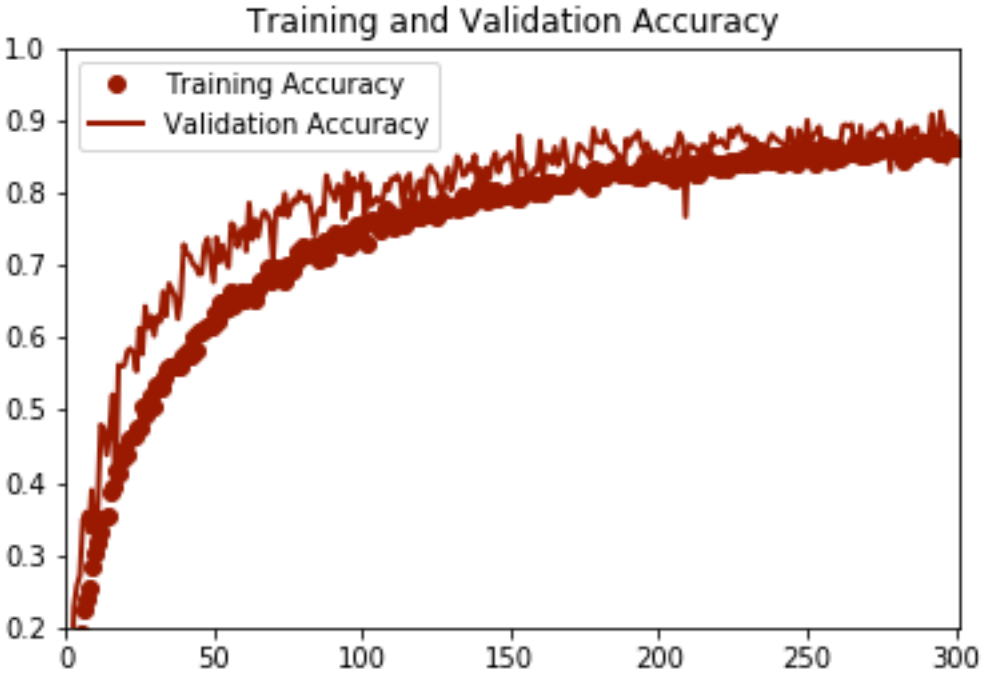
(ii) It is my opinion that the training set for images in the first medium set may have been insufficient when given such large amounts of classes to choose from.

(iii) I posit that if I were to change the classes to those with more example data, the results of the training would improve dramatically. I have now started a medium set 2.0 test.

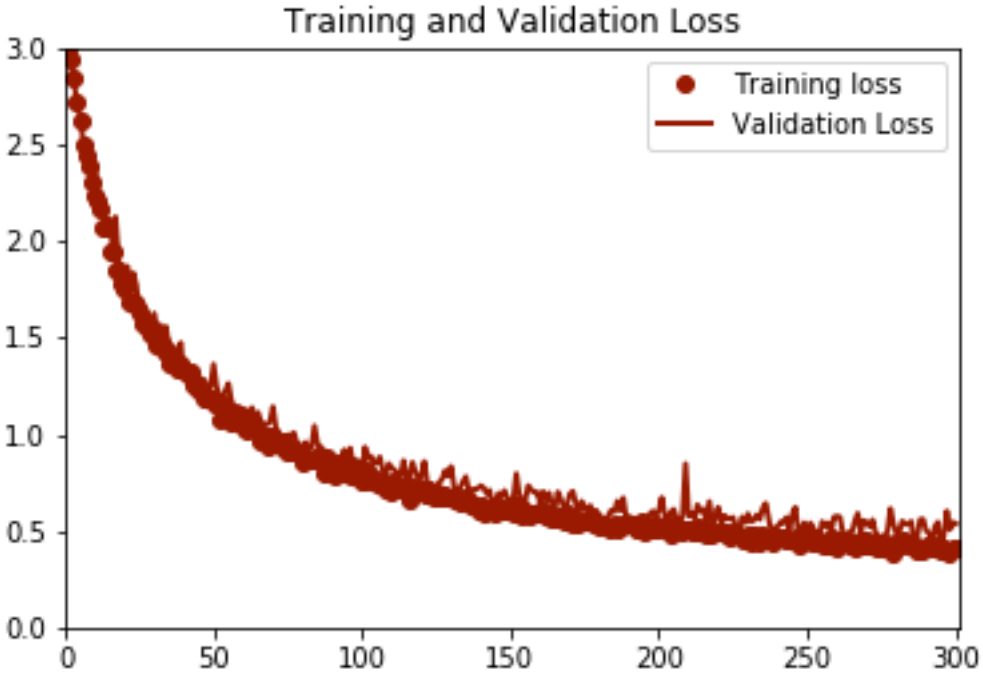
(iv) This test set will be 22 letters that have 8000 examples, compared to 5000 like the previous set of letters. This will give us a concrete decision of where to take our data set from here, more imaging seems necessary.

(v) Today I have completed 4 tests that aim to get closer to this answer. From what I can ascertain, this network seems to be no better than the previous one. There is some improvement in overfitting, but the overall accuracy rate is not getting any higher. This obfuscates the original point of these tests, but I have 2 more that I think will give a better conclusion on this topic.

(vi) There were some developments today that resulted in improvement over the previous 4 attempts today:


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
50
100
150
200
250
300

(vii)


Training and Validation Loss
Training loss
Validation Loss
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250
300

(viii)

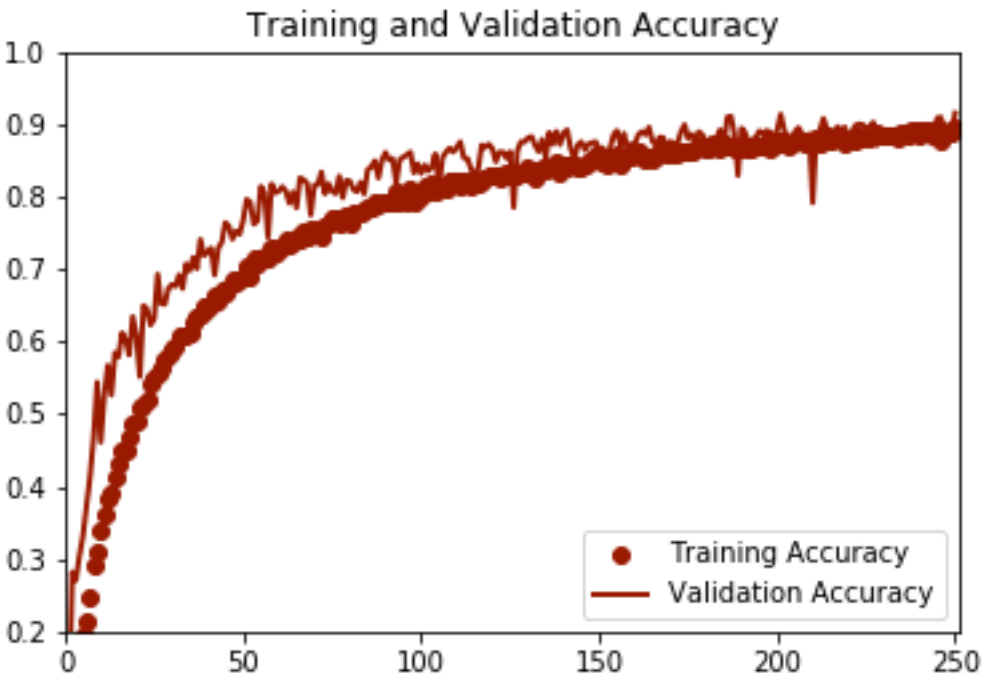
(ix) While these results are not our best they show that we may be suffering from a lack of data.

(x) The final test is currently happening, this test uses our most successful large scale model so far, medset4 on a larger amount of data.

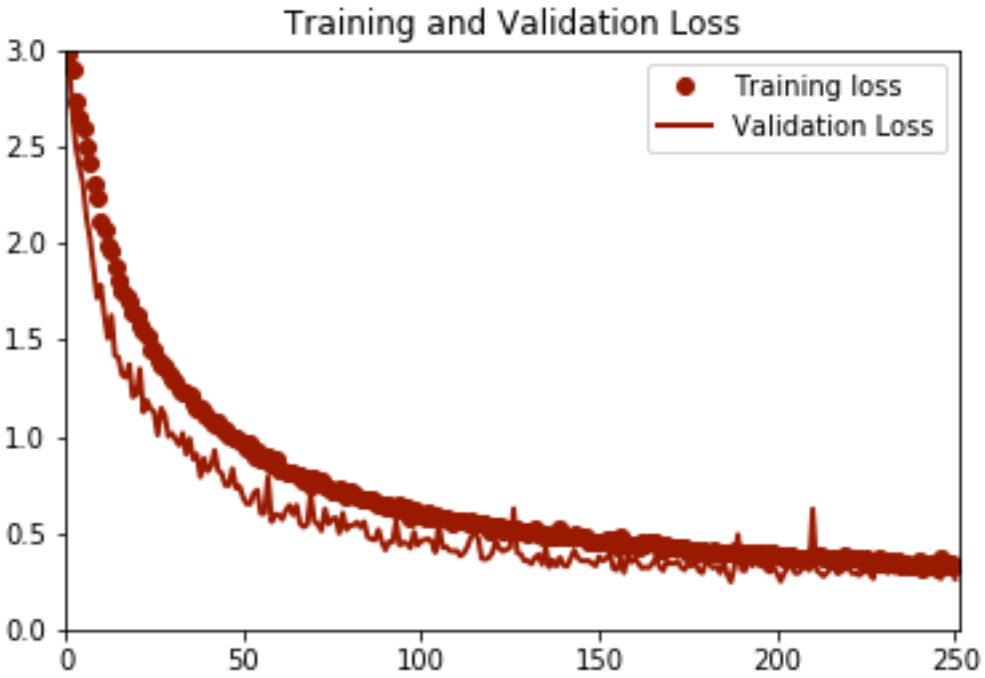
(xi) The results from the final test will be present in tomorrow's update. The test will over run into tomorrow and will have to be paused.

1.26 July 16th, 2019

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



(ii)



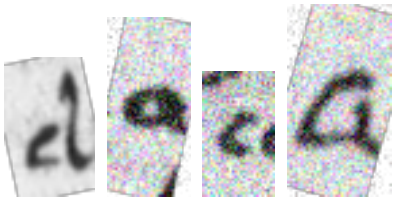
(iii)

(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.

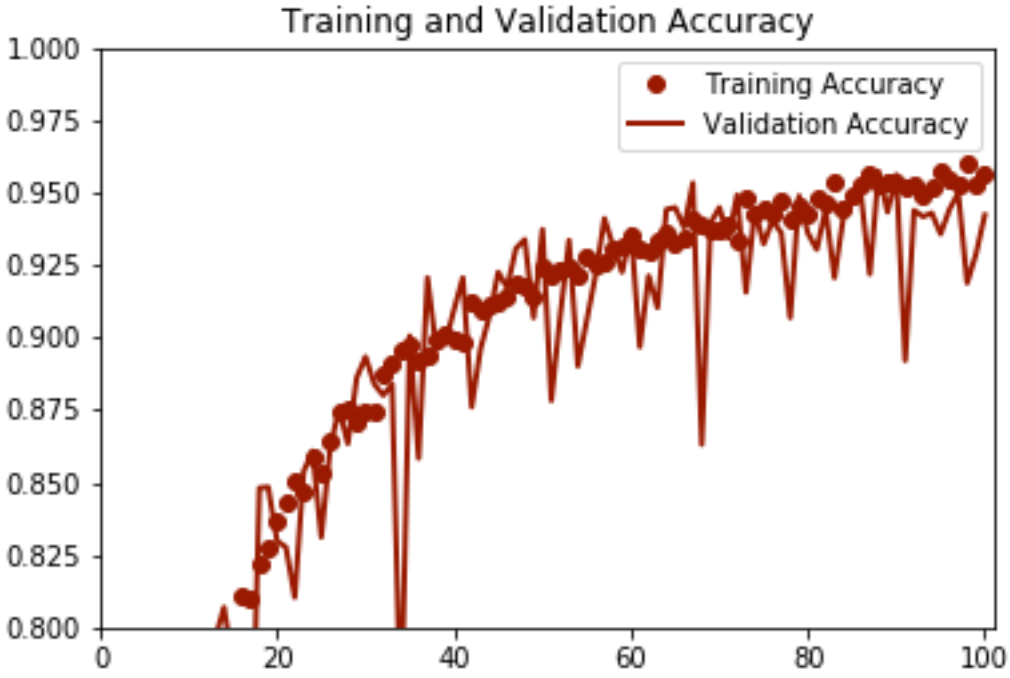
(vi) We went back to the original smallset test, which has 4 classes with 1,539 images. Through a program called image transform that I wrote, I augmented them and got 3,053 images total from the base set.

(vii) Examples of augmentation:

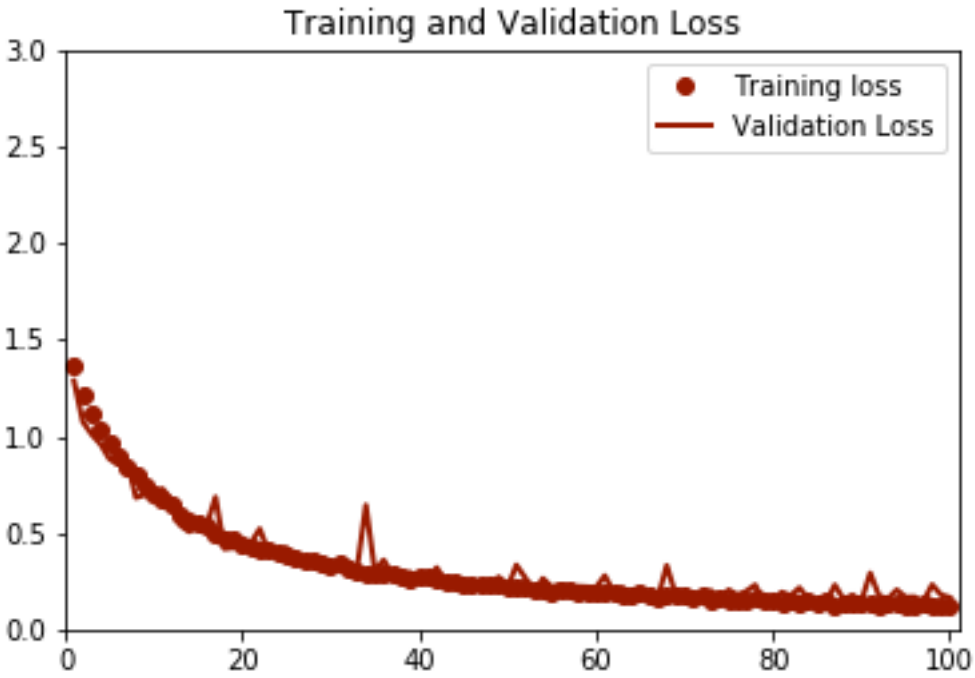
(viii) 

(ix) The results were as follows:

(x) Baseline:

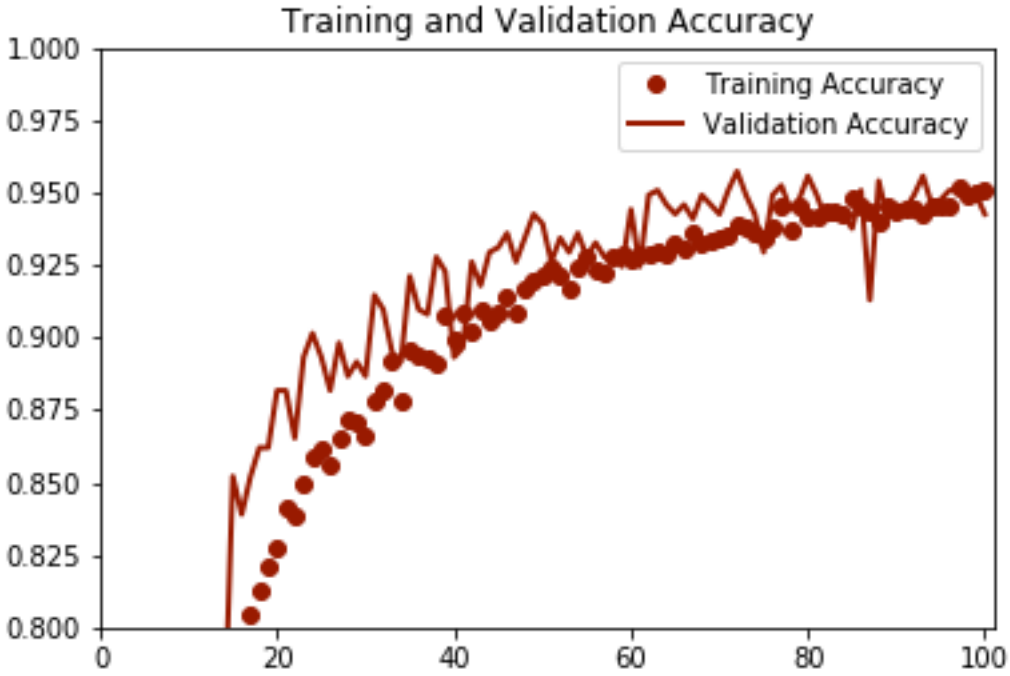


(xi)

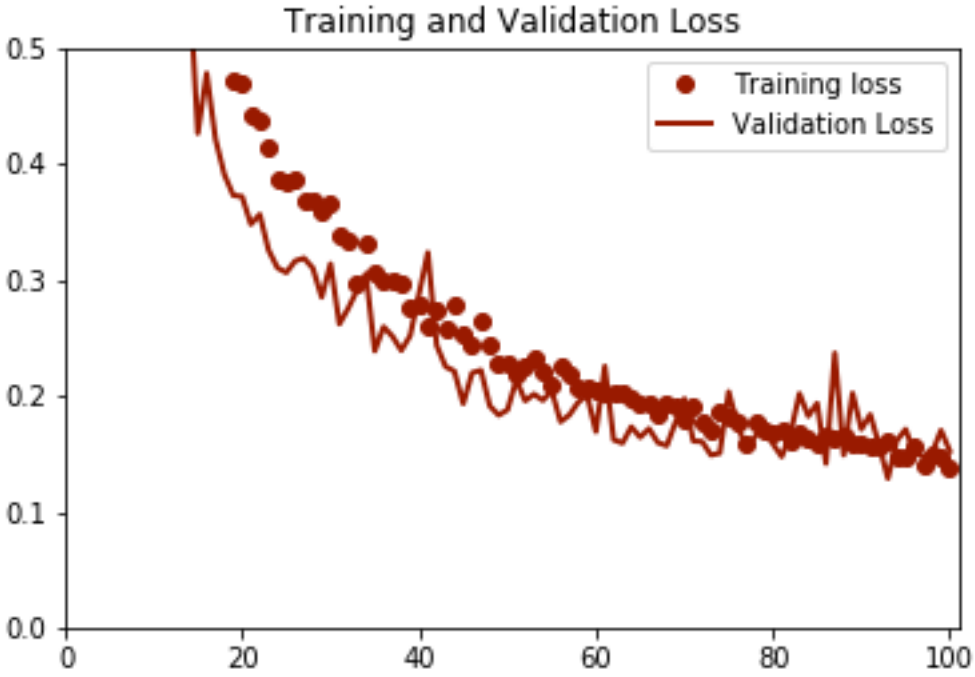


(xii)

(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

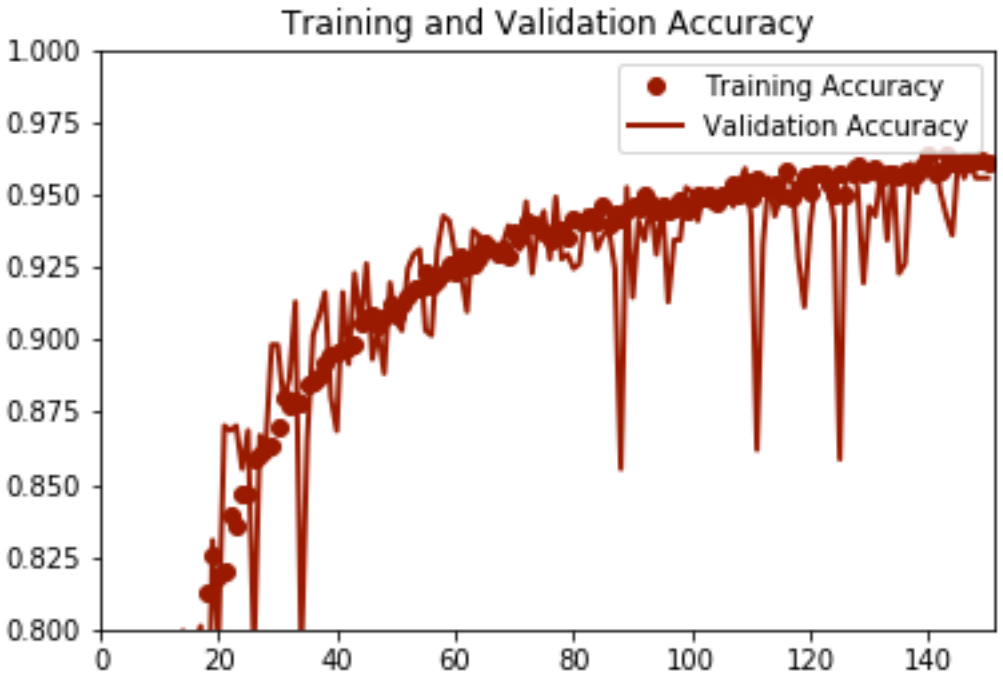
(xiv)


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

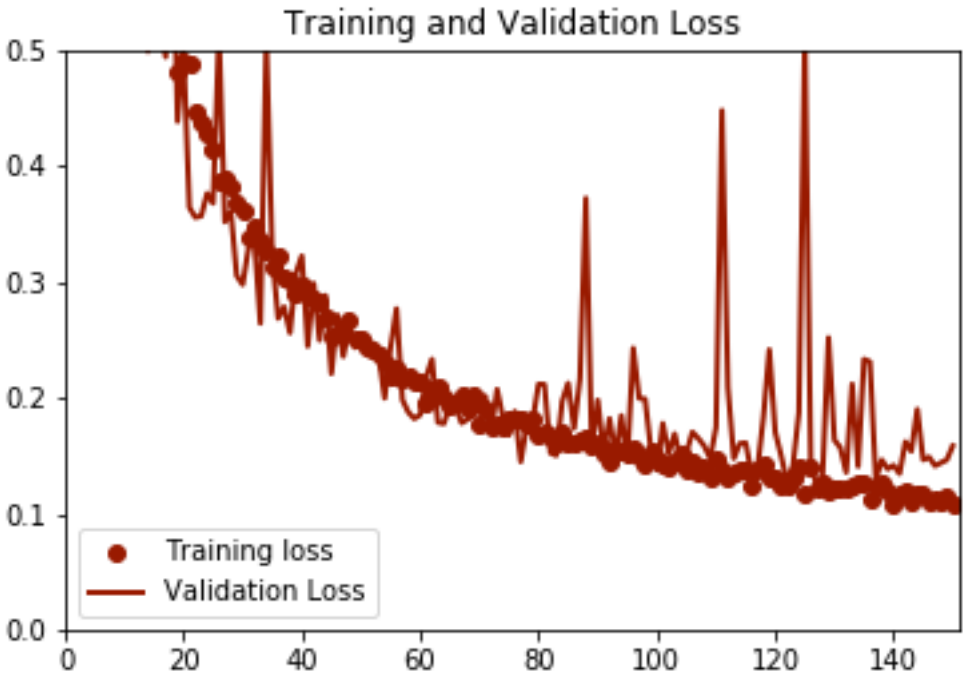
(xv)

(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)



(xix)

1.27 July 17th, 2019

(i) Yesterday I created a script that does large scale testing in order to compile data into csv files for visualization.

(ii) The script takes a path to the test files, a path to the model, and a dictionary of labels. The script then tests, and subsequently saves the results to a results.csv file.

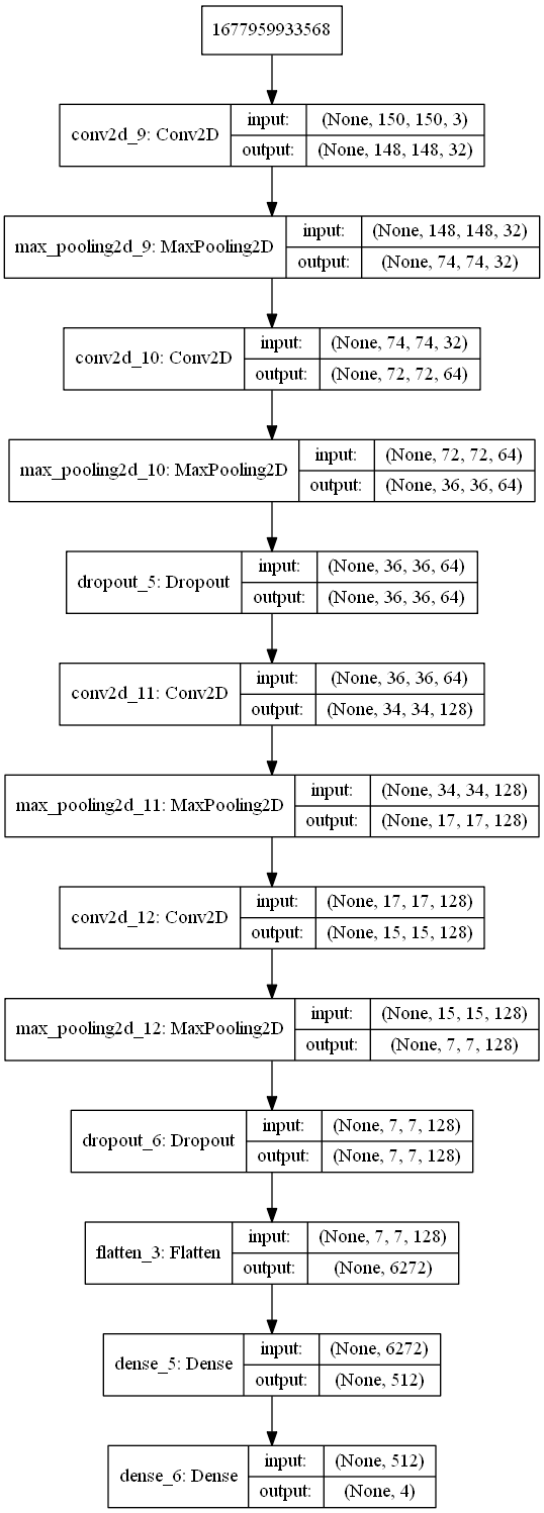
(iii) This will be useful to document our results with.

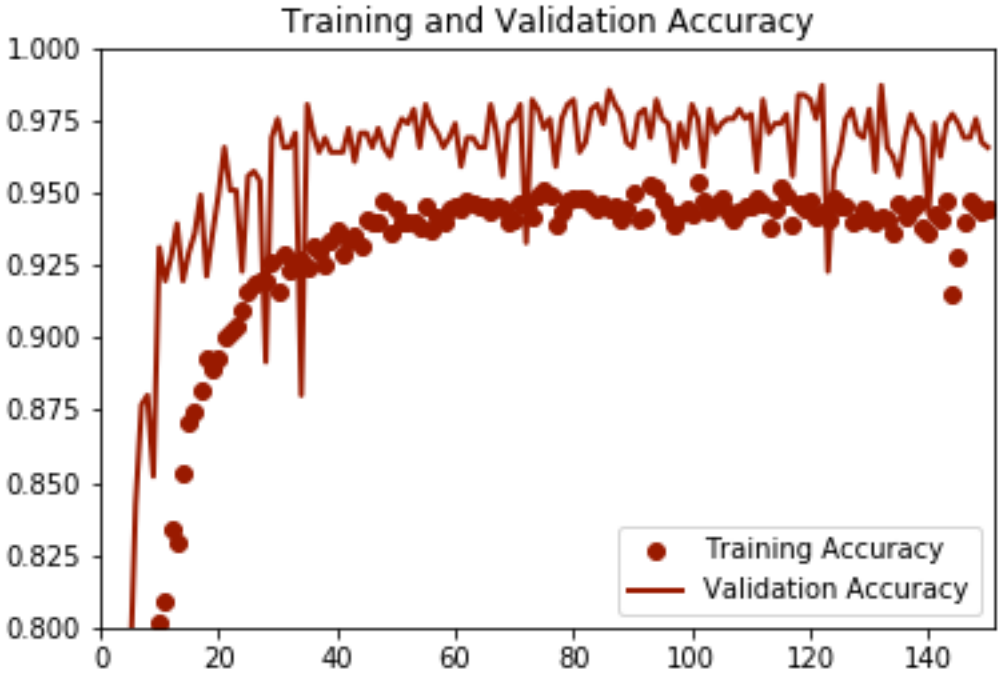
(iv) There have been several important developments at the end of yesterday on the project. For one I rewrote the data manipulation script to be random and the results seem to be an increase of 5% over the previous version.

(v) I have also modified the optimization function we use by adding an epsilon parameter of .9.

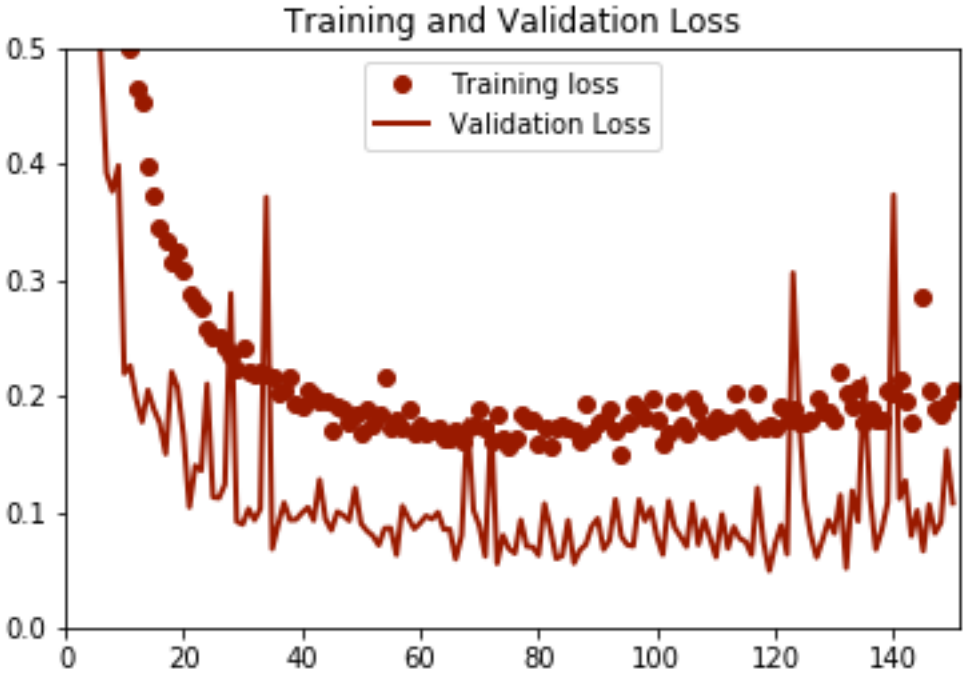
(vi) I have decided as well to increase dropout where possible to continue to limit our model's tendency to over fit. The results are great:

(vii)


1677959933568
conv2d_9: Conv2D
input: (None, 150, 150, 3)
output: (None, 148, 148, 32)
max_pooling2d_9: MaxPooling2D
input: (None, 148, 148, 32)
output: (None, 74, 74, 32)
conv2d_10: Conv2D
input: (None, 74, 74, 32)
output: (None, 72, 72, 64)
max_pooling2d_10: MaxPooling2D
input: (None, 72, 72, 64)
output: (None, 36, 36, 64)
dropout_5: Dropout
input: (None, 36, 36, 64)
output: (None, 36, 36, 64)
conv2d_11: Conv2D
input: (None, 36, 36, 64)
output: (None, 34, 34, 128)
max_pooling2d_11: MaxPooling2D
input: (None, 34, 34, 128)
output: (None, 17, 17, 128)
conv2d_12: Conv2D
input: (None, 17, 17, 128)
output: (None, 15, 15, 128)
max_pooling2d_12: MaxPooling2D
input: (None, 15, 15, 128)
output: (None, 7, 7, 128)
dropout_6: Dropout
input: (None, 7, 7, 128)
output: (None, 7, 7, 128)
flatten_3: Flatten
input: (None, 7, 7, 128)
output: (None, 6272)
dense_5: Dense
input: (None, 6272)
output: (None, 512)
dense_6: Dense
input: (None, 512)
output: (None, 4)


Training and 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
120
140
Training Accuracy
Validation Accuracy

(viii)


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

(ix)

(x) The small discrepancy between training and validation acc/loss is due to using dropout on the model, this difference would converge over time.

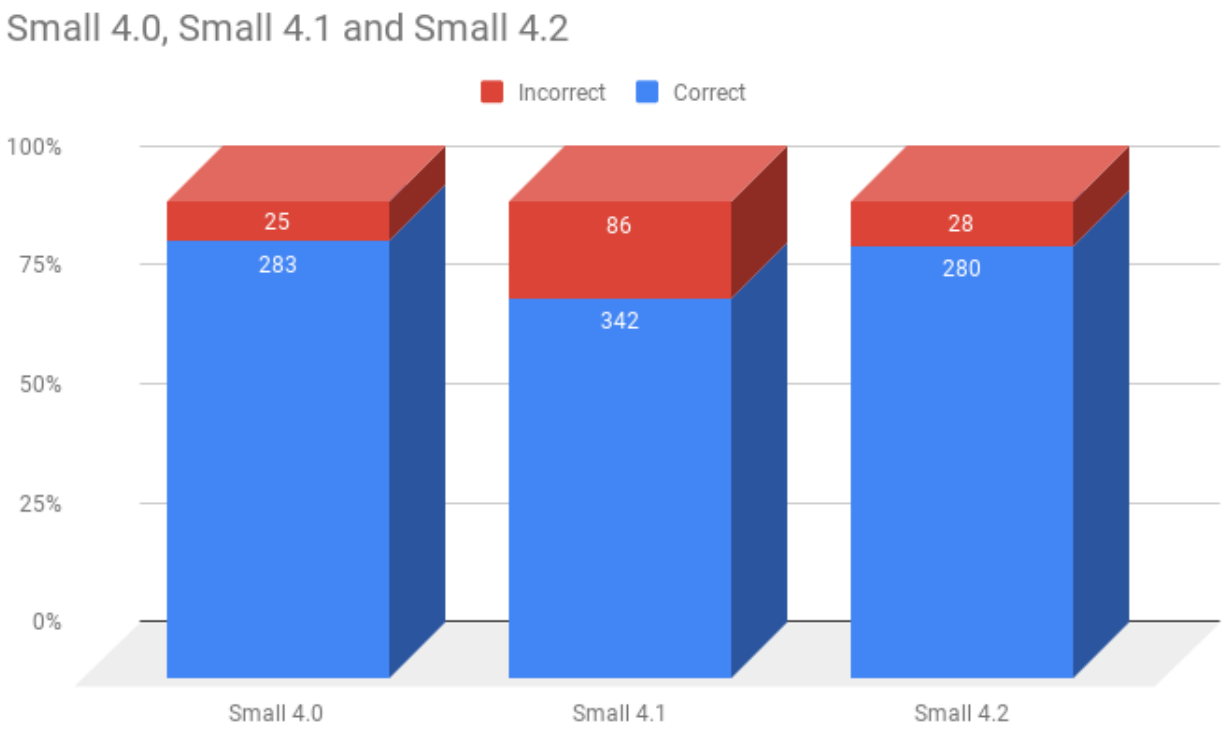
(xi) The most successful test yet has just been run with the current stats:

(xii) Test accuracy: Correct: 293 Incorrect: 15 Ratio: 0.951298701

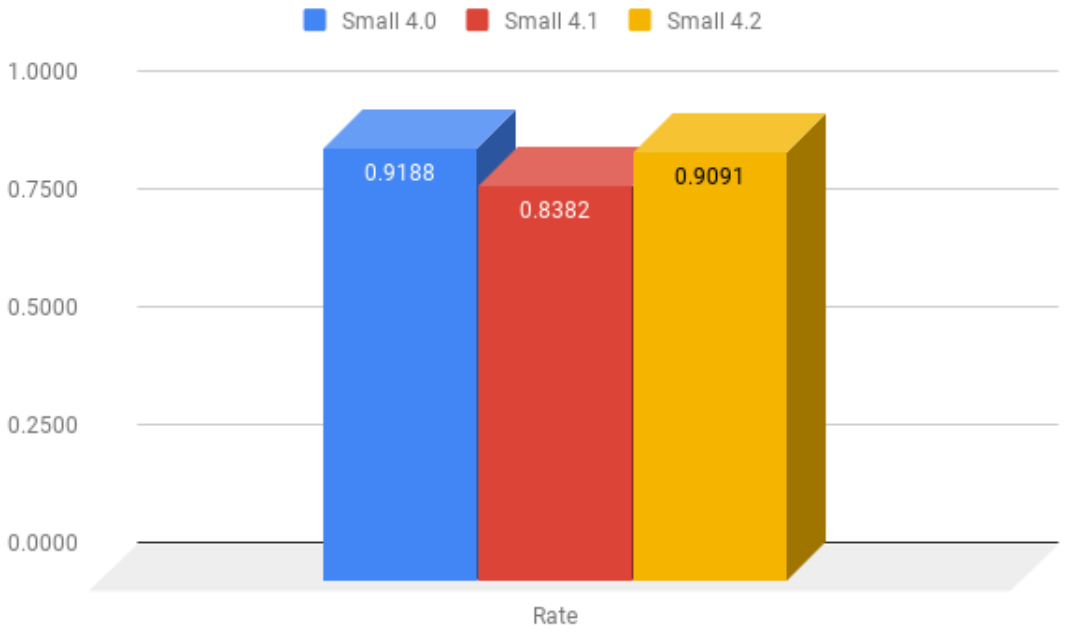
(xiii) This model is called convnet-smallset-test3-model-TEST3.h5

1.28 July 25th, 2019

(i) Today's tests were run in order to decide the future of the project as far as trajectory. After several tests I think that it is clear that our data set must be increased. If I had to give an estimation I would say that to realistically solve this problem at high sucess rates, we would need something on the order or 3-5x more images per sample. I believe this is due to the inter-class noise that our model is picking up. It is having a hard time deciding between certain letters because their characteristics are not defined different enough through the data. That being said we still have great success rates given the reality of this problem. For example, our 19 class test currently runs around 80% accurate. To put this into perspective, it is still 15.2 times better than random guessing. To me, this means our model is built correctly, yet it still doesn't have enough data to properly discern classes. Below I have shown the conclusions of the tests to support my hypothesis.


Small 4.0, Small 4.1 and Small 4.2
Incorrect
Correct
100%
75%
50%
25%
0%
25
283
86
342
28
280
Small 4.0
Small 4.1
Small 4.2

(ii)


Small 4.0
Small 4.1
Small 4.2
1.0000
0.7500
0.5000
0.2500
0.0000
0.9188
0.8382
0.9091
Rate

(iii)

(iv) Notes: Small 4.0 is the base neural network for 4.1 and 4.2. The differences are as follows:

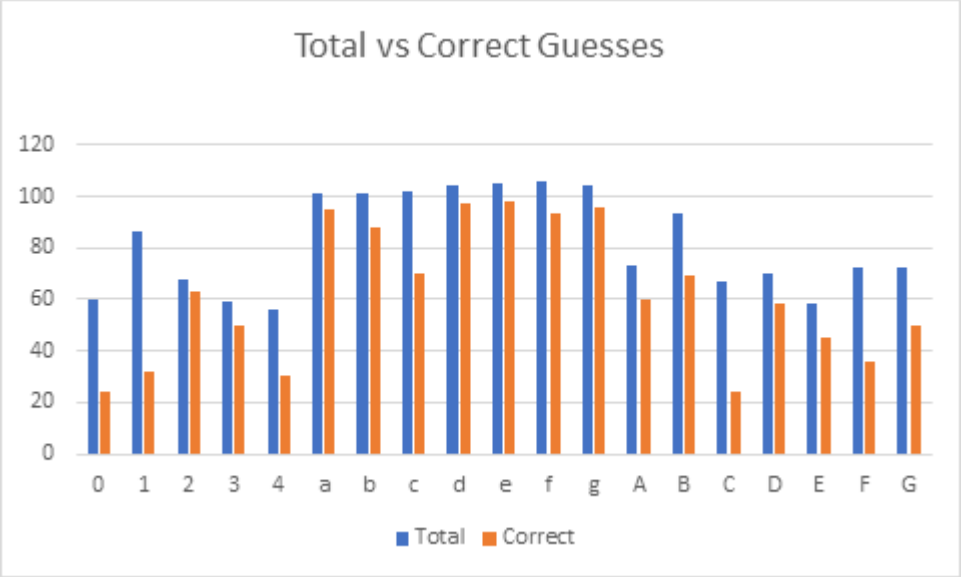
(v) 4.0 vs 4.1: 4.1 increased the augmentation from 250 photos per class to 500 photos per class, which is reflected in the increase in overall numbers for 4.1. This was not a positive improvement.

(vi) 4.1 vs 4.2: 4.2 retained the same model as 4.1 and 4.0. 4.2 Used a augmentation size of 300 as well as removed the horizontal flip option from the augmentation. I did this as I thought that flipping a d or b horizontally could lead to classification issues. It does not seem that this had a positive effect.

(vii) I also compiled some useful data regarding to the medium (19 class) set.

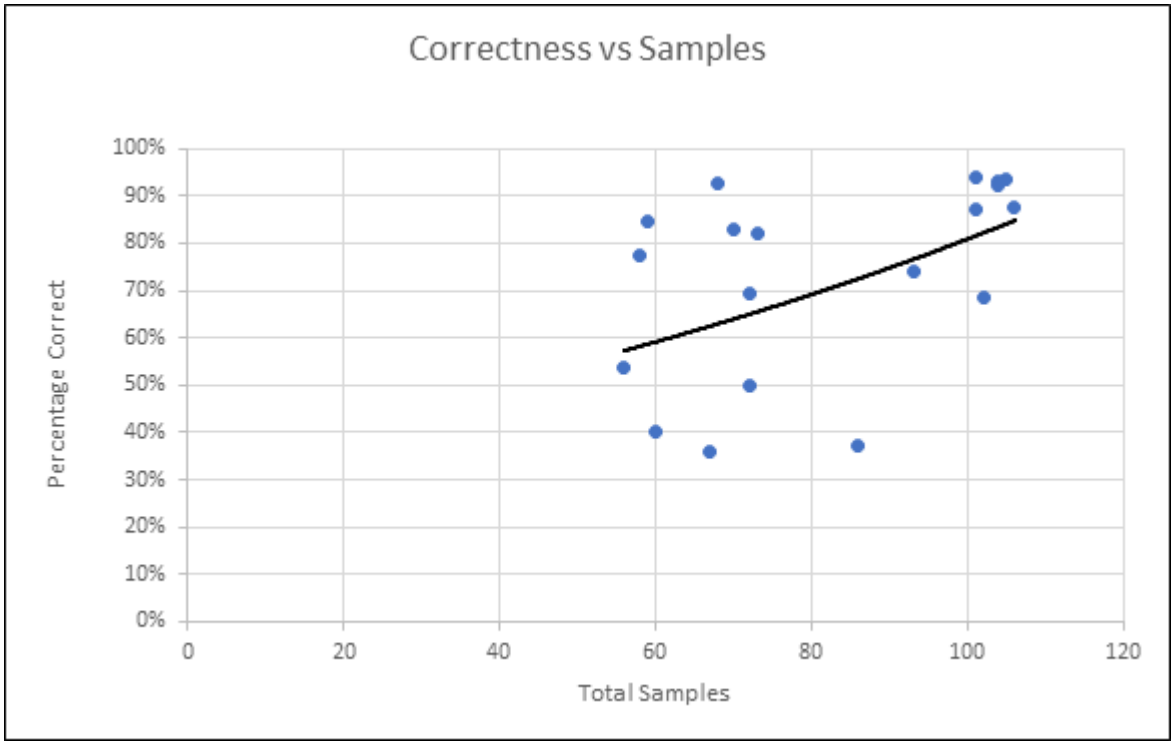
(viii) The following is a graph showing the total number of samples vs correct guesses that was achieved during a test on the 7th iteration of the medium set model.

(ix)



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

(xi)



(xii) 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.