Week 8 Assignment

Efeosa Eguavoen - 17324649

December 1, 2020

1 i

1.1 A

1.2 B

2 ii

2.1 A

2.2 B

2.2.1 i

Keras says the model has 37,146 parameters. The dense layer has the most parameters with 20,490 parameters. This is because the dense layer is a fully connected layer with each output being a function of the weighted sum of all the inputs. Since this layer can have multiple outputs, it means the number of parameters that need to be learned here increases rather quickly. It has n*m parameters with n being the length of the input vector and m being the number of output channels. The test data performs significantly worse then the training data in terms of accuracy. Both classifiers don't perform particularly well, but when compared to a random classifier, they both still perform significantly better. I used a random classifier as my baseline to test if the model is actually working and learning the the data properly versus randomly selecting numbers.

Accuracy of Test Data: 0.49
Accuracy of Training Data: 0.64
Accuracy of Random Classifier: 0.10

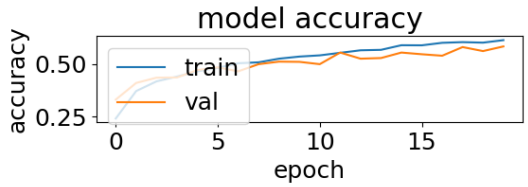
2.2.2 ii

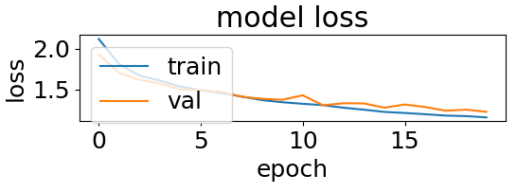
From the charts the history variable outputs, it seems that as time goes one the model captures the trends in the data more and more. Initially it under-fits both the training dataset and the testing dataset, but the test data set performs better. But after 20 epochs, this switches with the training data performing better than the test data. It seems that the model is much better at predicting to the training data than the testing data. This is probably due to the model possibly over-fitting to the training data and not being general enough to be able to predict on new data accurately.

2.2.3 iii

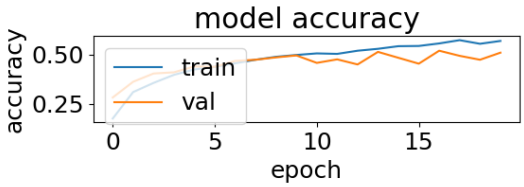
The prediction accuracy seems to increase slightly with more training data. The 5K dataset has a lower overall prediction accuracy compared to any of the other datasets. .The training data accuracy only increases by 10% with more training data, with the 40K dataset having an accuracy of 0.74 compared to 0.64 with the 5K dataset. But the test dataset performs significantly better going to 0.7 with the 40K dataset from 0.49 with the 5K dataset, an improvement of about 20%. From the history plots below, we can see that as we increase the amount of training data, our accuracy increases while our loss decreases. But it seems the returns decrease more and more as we increase the size of the dataset.
In terms of time taken per step:
5K dataset: 7.2ms/step
10K dataset: 6.5ms/step
20K dataset: 1s 06ms/step

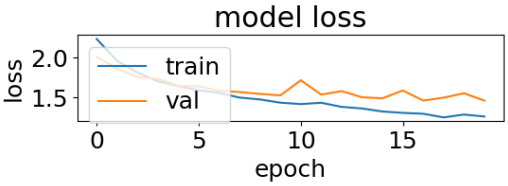
40K dataset: 2s 7ms/step.
As the size of out dataset grows, the time taken per step increases, taking about 3x as long for a setp for a dataset that's 8x as big. The tradeoff here seems to be worth it given the jump in performance for little extra computing time. (I used a GTX 1070 GPU to run this hence the times being low).



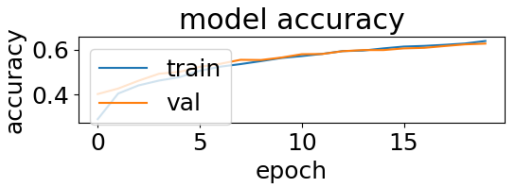


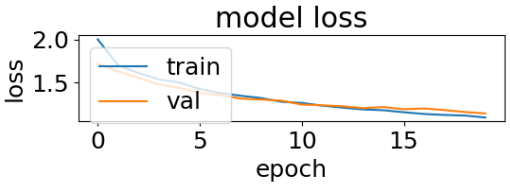
(a) 5k Dataset



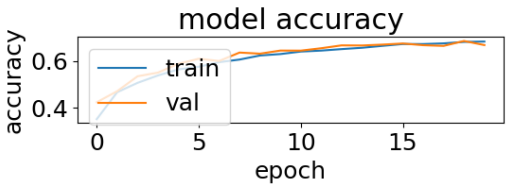


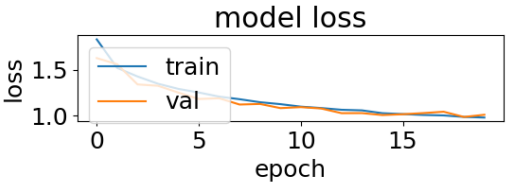
(b) 10k Dataset





(c) 20k Dataset

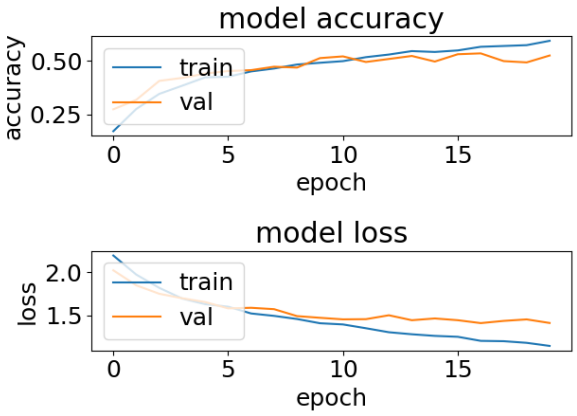




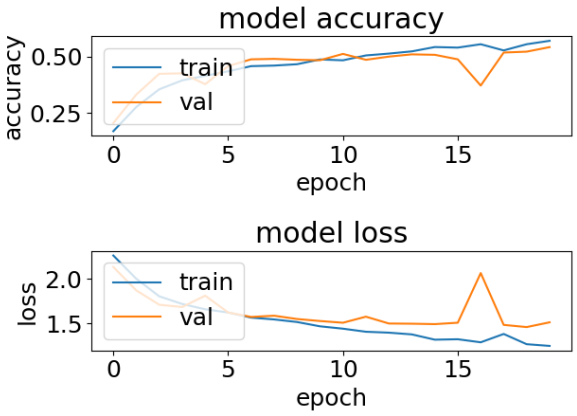
(d) 40k Dataset

2.2.4 iv

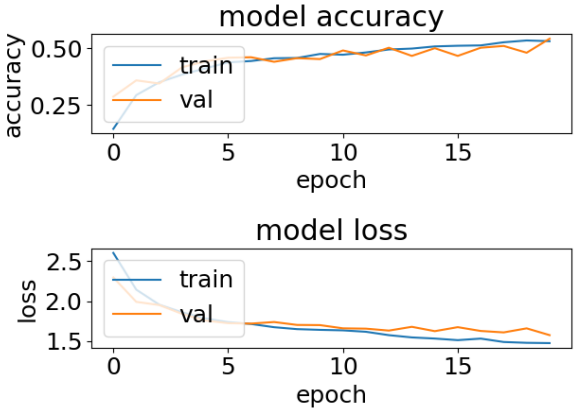
As we vary the L1 penalty, the accuracy fluctuates quite a bit. With 0, our accuracy actually increases in terms of the training and test data but as we increase the penalty, the accuracy become worse and worse. Using a large L1 value such as 10, out accuracy is terrible got both our training and test datasets. This could be due to the fact that by increasing th L1 penalty, the CNN attempts to prevent overfitting more and more and hence begins underfitting the data with larger values of L1 since the weights are so high.



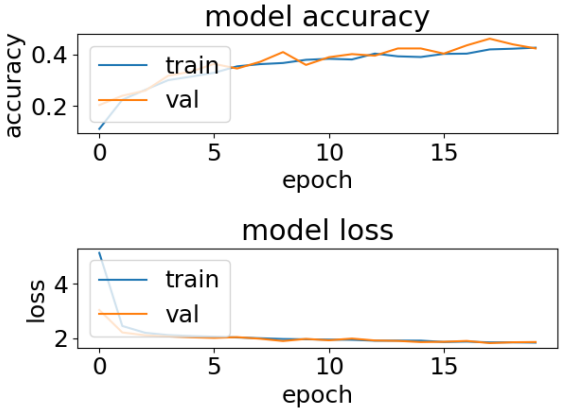
(a) L1 = 0



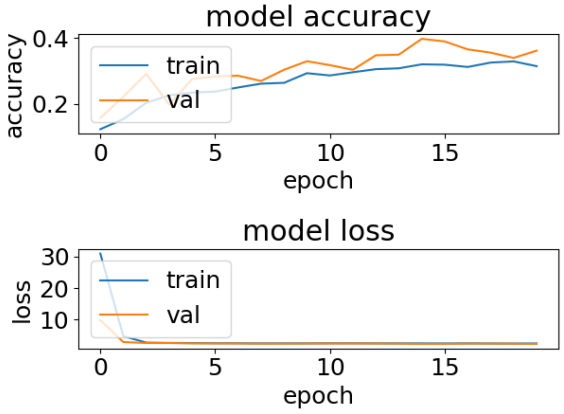
(b) L1 = 0.0001



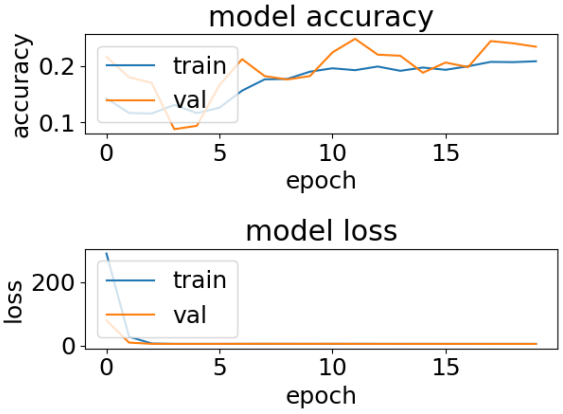
(c) L1 = 0.001



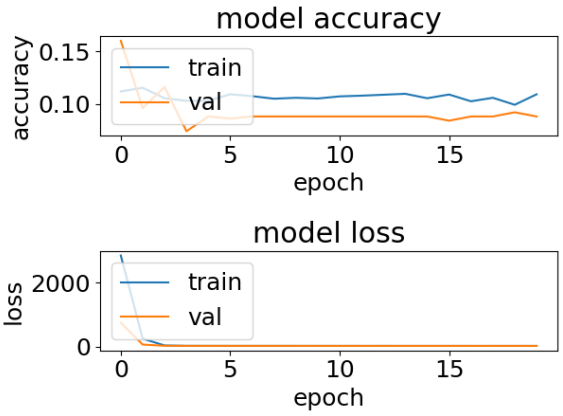
(d) L1= 0.01



(e) L1 = 0.1



(f) L1 = 1



(g) L1 = 10

2.3 C