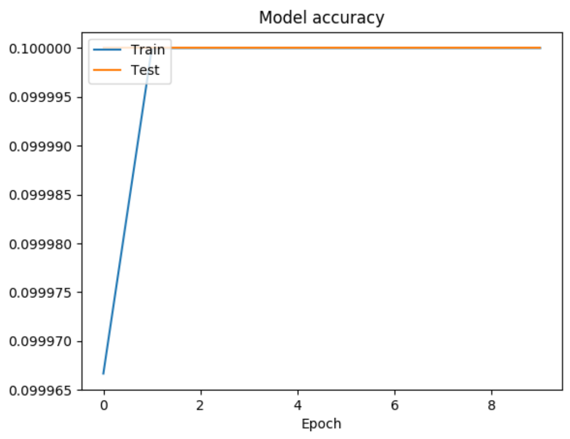
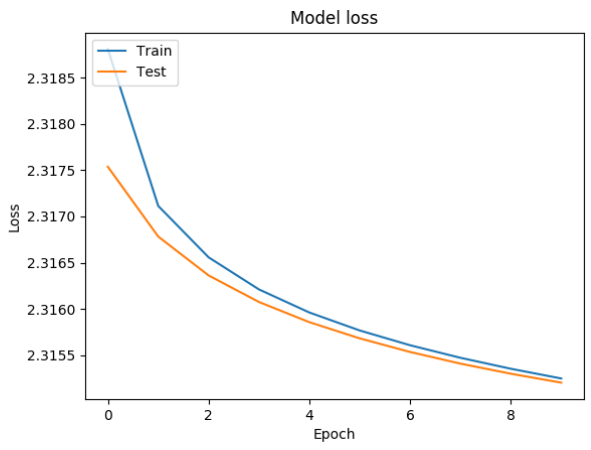
Harshdeep Goraya
421 : Project 1

MLP Net

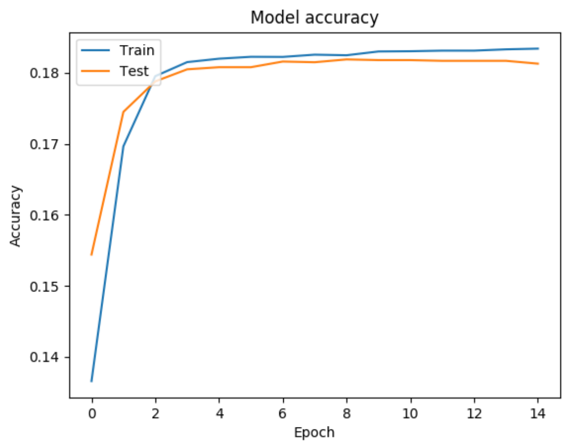
V1

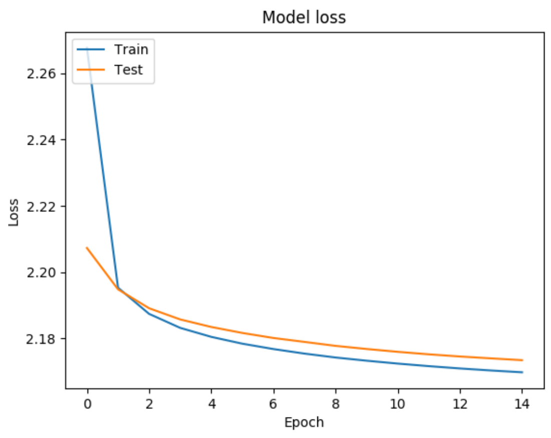




These graphs represent the performance of a 2 layer MLP. For each epoch, accuracy is shown by the blue train line which is 10%; as the loss value is shown to be approximately 2.32. Note this was done for 10 epochs, as the we will increase the number of epochs I suspect the accuracy to increase since the model will *see* the data more times.

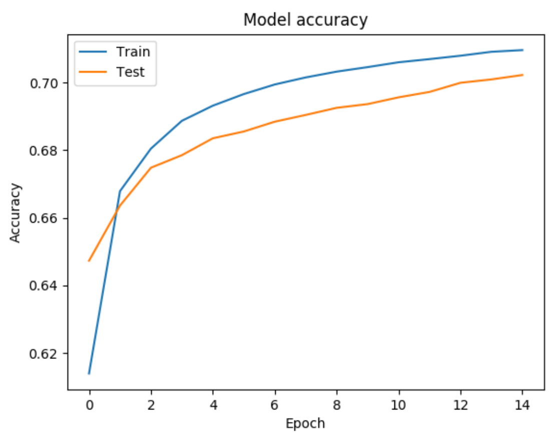
V2

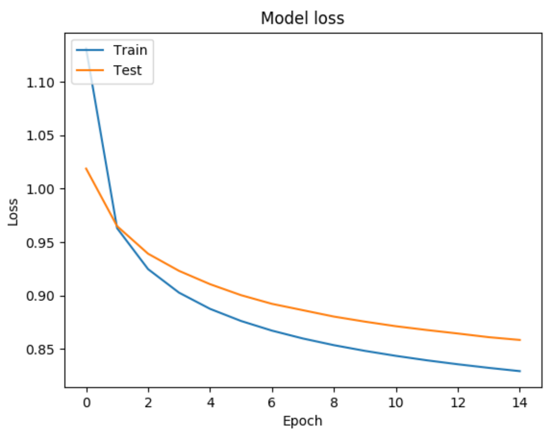




These graphs represent the performance of a 5 layer MLP. Training for a total of 15 epochs, each one had an accuracy of about 18% (increased from last run), and a loss value of about 2.17. In this network, units were taken out without us causing it to lose any valuable info.

V3

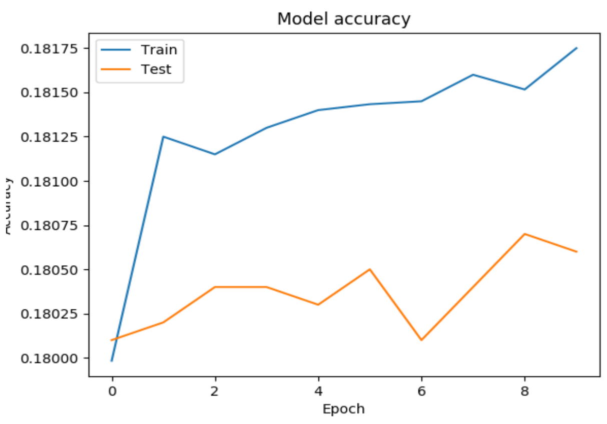


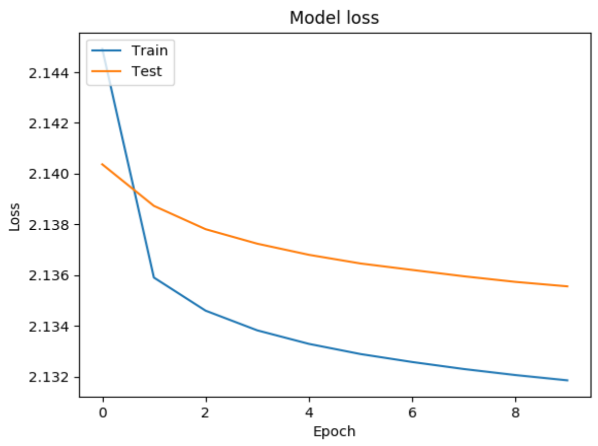


These graphs represent the performance of a 8 layer MLP. As the layers increased, the number of units consistently were around 20-40 for each layer (only two low unit layers throughout the total 8); this gave us an approximate accuracy of about 62-70 % for each epoch. The loss value for each epoch was about 0.83-1.1. With this increased number of units for each added layer compared to the last run, it was able to produce a much higher training/testing accuracy overall. (Highest one so far)

Conv Net

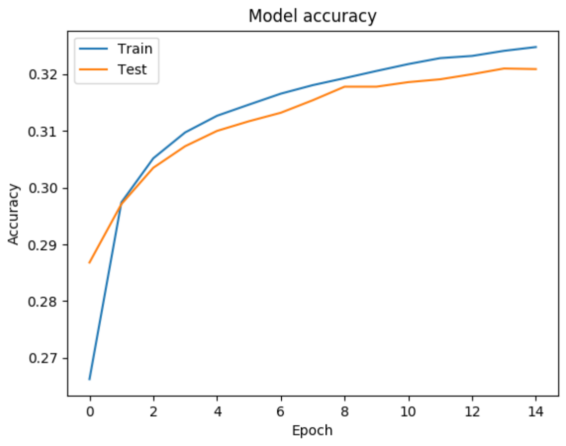
V1

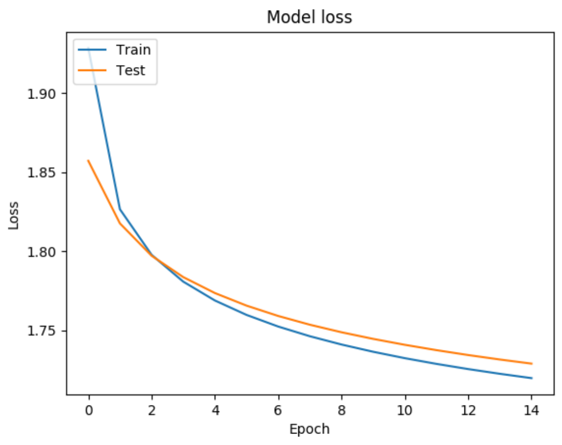




This represents adding 2 CNN layers right before and after the max-pool layer in the network. The accuracy per epoch was around 18%, as the loss value looked to be around 2.13 all around. The max-pool layer reduced our x-y dimensions by half, as it seems this layer assisted in reducing the total number of computations for the network.

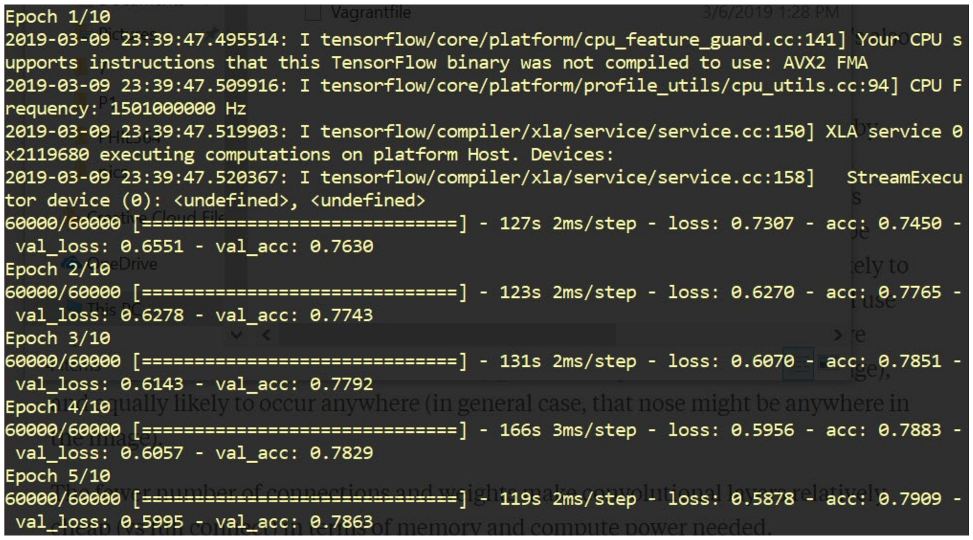
V2





After increasing the units in the Dense layer following the 2 CNN layers, the accuracy per epoch increased to about 29-32%. The loss value for each epoch was approximately 1.86-1.73 (decreasing manner). Just by changing the last Dense layer units, we increased our accuracy. The units will further be increased, and another Dense layer will be added to see if this increase is consistent.

V3



```
Epoch 1/10
2019-03-09 23:39:47.495514: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU s
upports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2019-03-09 23:39:47.509916: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU F
requency: 1501000000 Hz
2019-03-09 23:39:47.519903: I tensorflow/compiler/xla/service/service.cc:150] XLA service 0
x2119680 executing computations on platform Host. Devices:
2019-03-09 23:39:47.520367: I tensorflow/compiler/xla/service/service.cc:158]   StreamExecu
tor device (0): <undefined>, <undefined>
60000/60000 [==============================] - 127s 2ms/step - loss: 0.7307 - acc: 0.7450 -
 val_loss: 0.6551 - val_acc: 0.7630
Epoch 2/10
60000/60000 [==============================] - 123s 2ms/step - loss: 0.6270 - acc: 0.7765 -
 val_loss: 0.6278 - val_acc: 0.7743
Epoch 3/10
60000/60000 [==============================] - 131s 2ms/step - loss: 0.6070 - acc: 0.7851 -
 val_loss: 0.6143 - val_acc: 0.7792
Epoch 4/10
60000/60000 [==============================] - 166s 3ms/step - loss: 0.5956 - acc: 0.7883 -
 val_loss: 0.6057 - val_acc: 0.7829
Epoch 5/10
60000/60000 [==============================] - 119s 2ms/step - loss: 0.5878 - acc: 0.7909 -
 val_loss: 0.5995 - val_acc: 0.7863
```

(Best Working Net)

This represent placing the Max-pool layer after our 4 CNN layers, as well as increasing the amount of units in the Dense layer followed by another Dense layer with a similar (slightly smaller) number of units. Per epoch, the accuracy looked to be approximately 75-80%, as the loss value for each was about 0.66-0.60.

Analysis:
It seems as the convolutional network was much more efficient than the fully connected. More so, since in the fully connected every unit will be connected to every single unit in the layer before; it will be expensive in terms of the weight as well as computation.

While in a convolutional net, each unit may only be connected to a few nearby units. The fewer the number of connections make convolutional nets cheaper. This is basically a locally shared weight connection.