Jeff Hildebrandt

PA4 Report

1:

N/A

2:

Test accuracy: 0.9908

3:

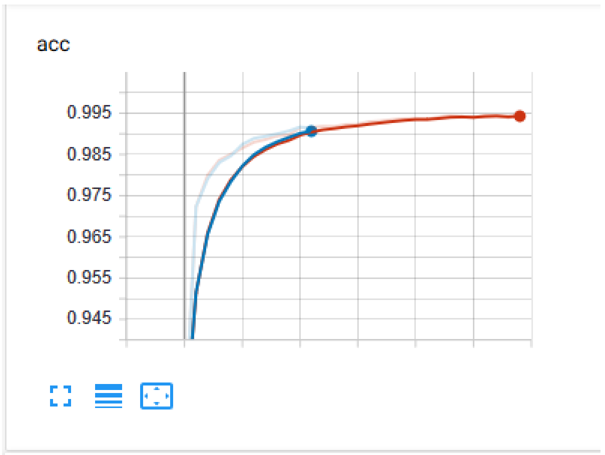
Layer	**Number of Filters/nodes**	**Filter Dimensions**	**Number of Parameters**	**Activation Function**
Conv	32 filters	3x3	320	Relu
Conv	64 filters	3x3	18,496	Relu
Max pooling	64 filters	2x2	0	N/A
Flatten	9216	N/A	0	N/A
Fully Connected	128 nodes	N/A	1,179,776	Relu
Fully Connected	10 nodes	N/A	1,290	Softmax

4:

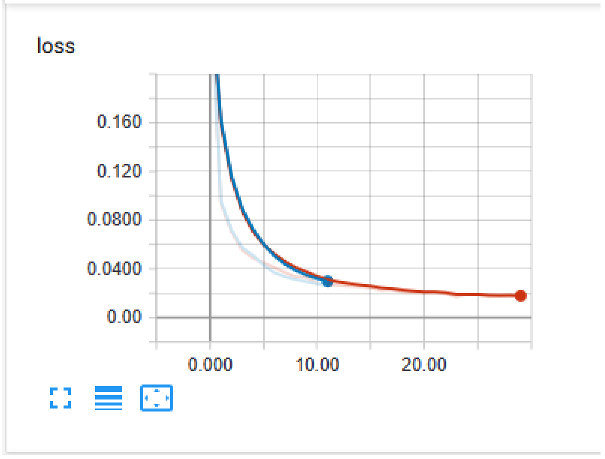
The blue line represents 12 epochs

The red line represents 30 epochs

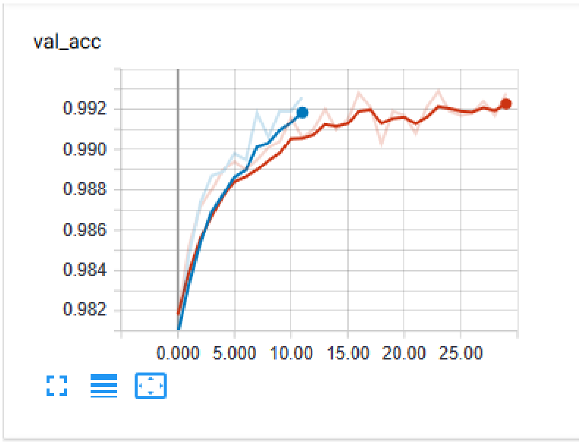
For training:



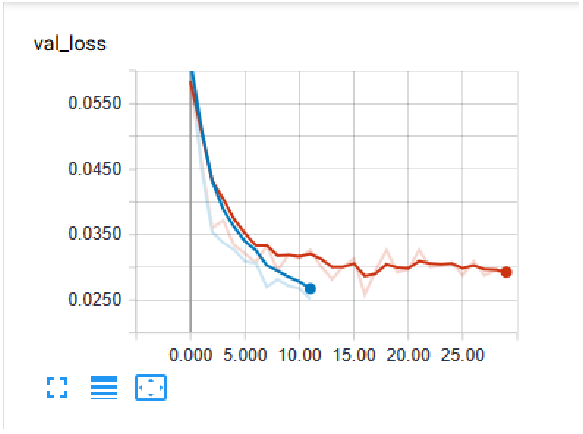
loss



For test:



val_loss

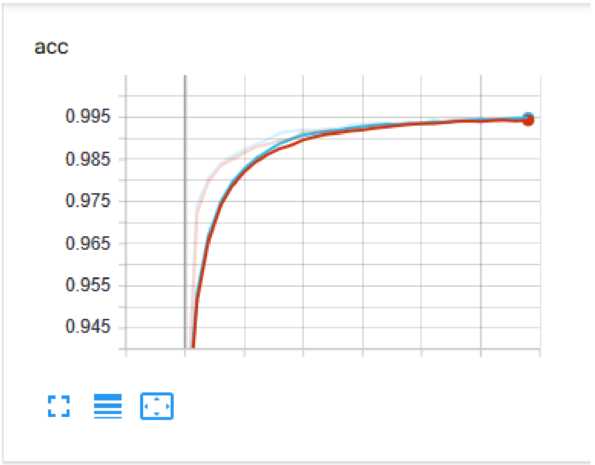


<u>5:</u>

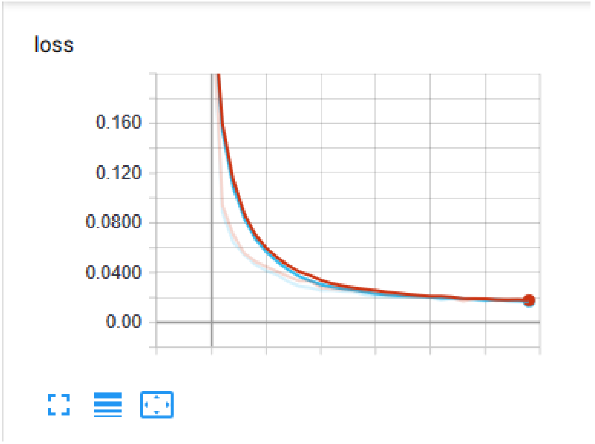
The red line represents having test data the model has not seen before

The light blue line represents having the same test data as training data

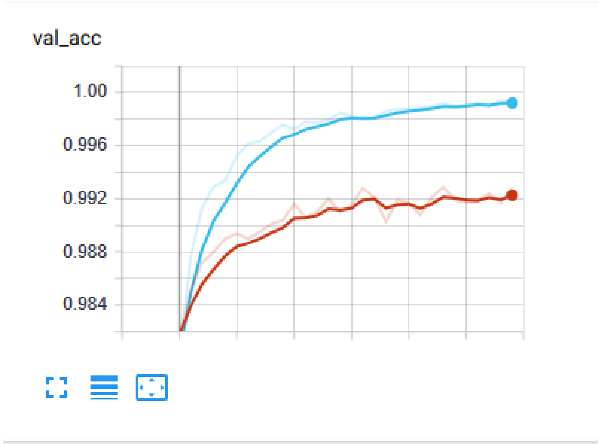
For training:



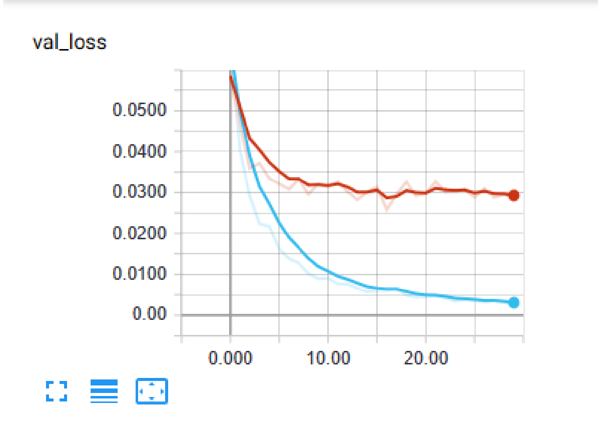
loss



For test:



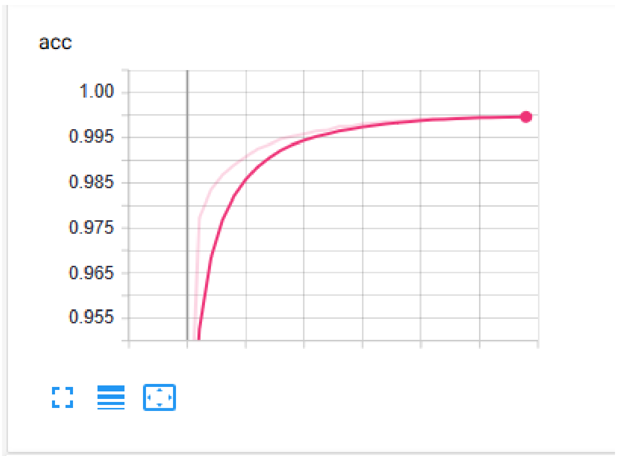
val_loss

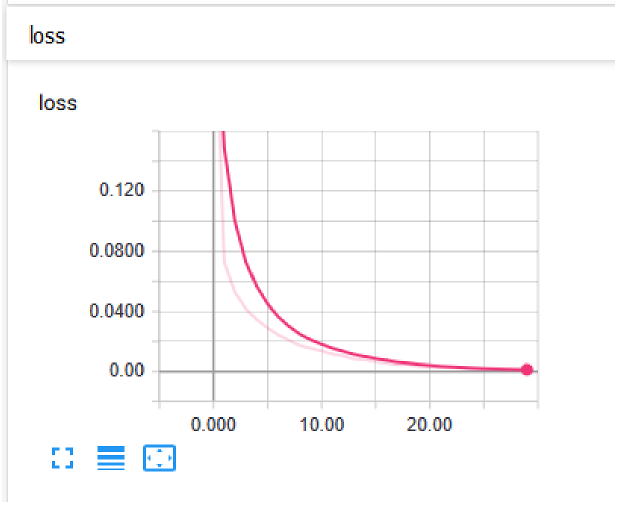


6:

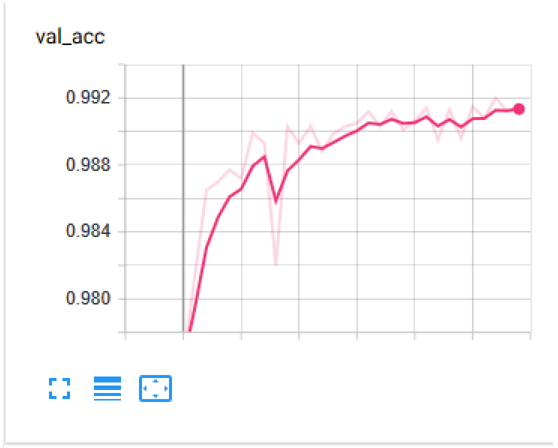
LeNet5

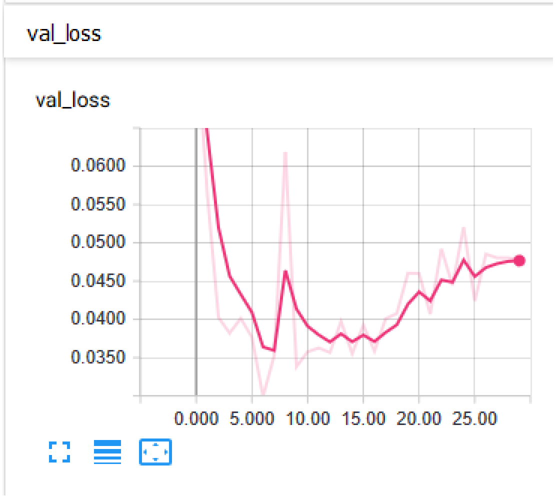
For training:





For test:





7:

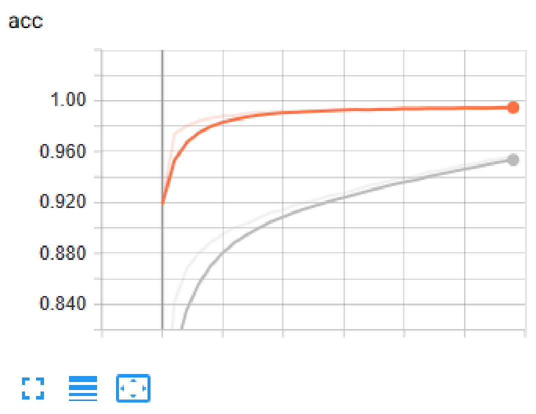
The gray line represents the LeNet5 model

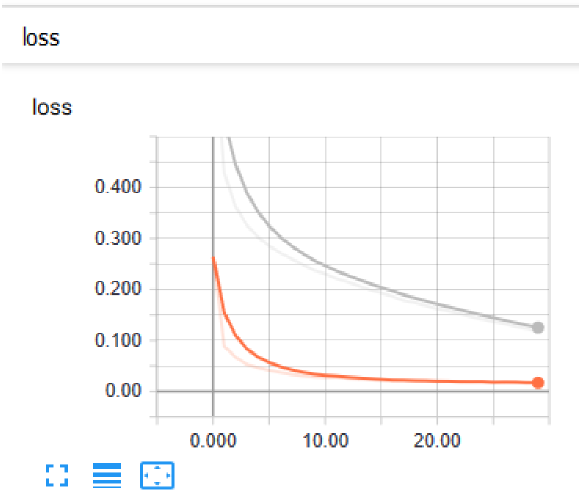
The orange line represents the simple ConvNet model

LeNet5 had an average training time of **3.22s per epoch**

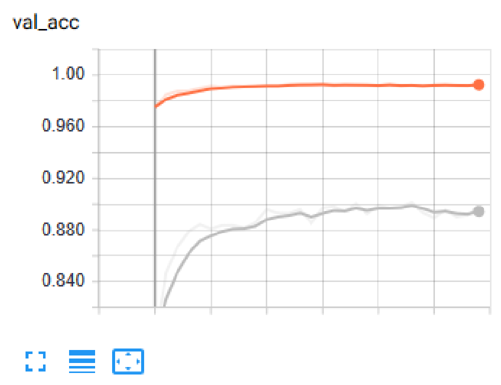
Simple ConvNet had an average training time of **7.24s per epoch**

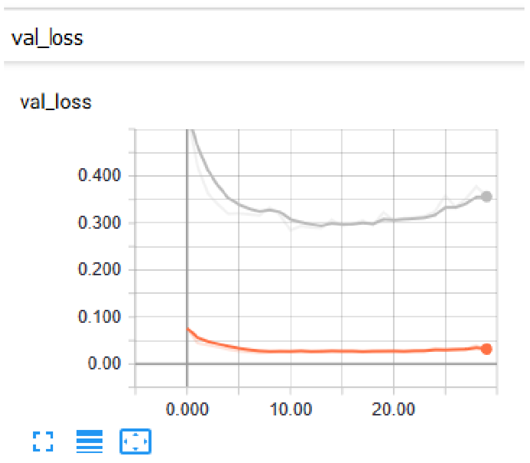
For training:





For test:





The data shows the deeper model (LeNet5) performed worse than the simple ConvNet model in terms of accuracy.

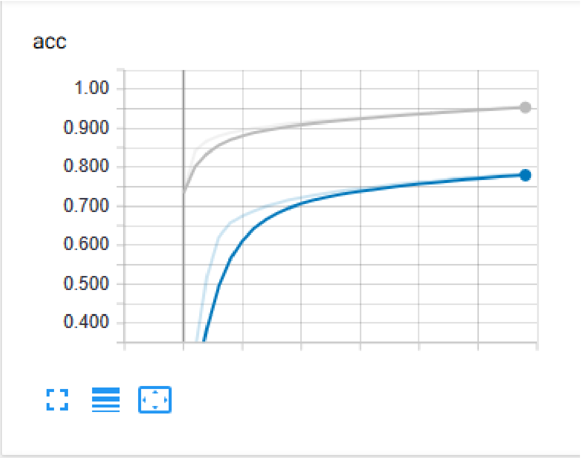
8:

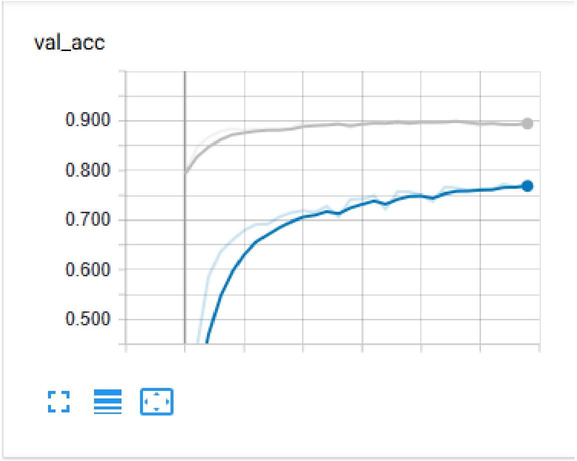
The gray line represents the LeNet5 model with the Adadelta optimizer over the Fashion MNIST dataset

The blue line represents the LeNet5 model with the SGD optimizer over the Fashion MNIST dataset

For training:

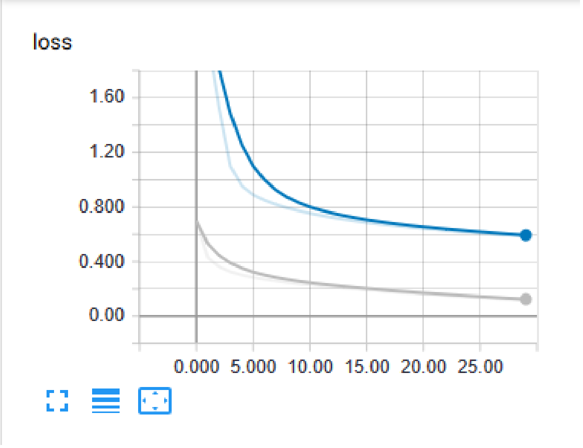
For test:

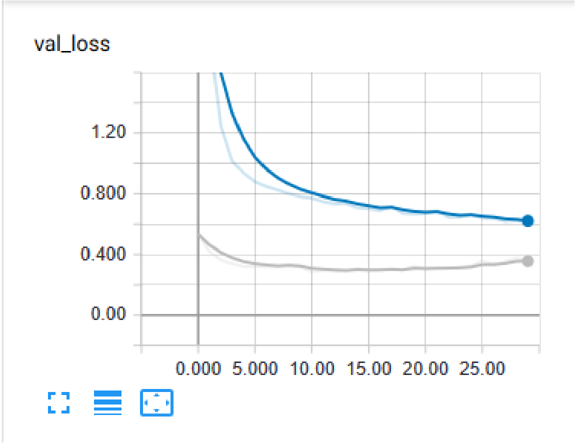




loss

val_loss





<u>9:</u>

The red line represents a learning rate of 0.1

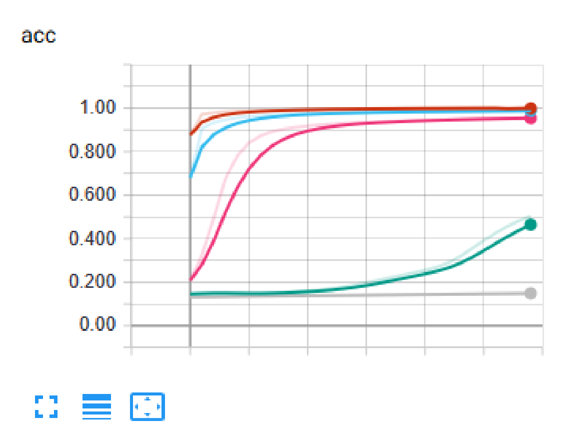
The light blue line represents a learning rate of 0.01

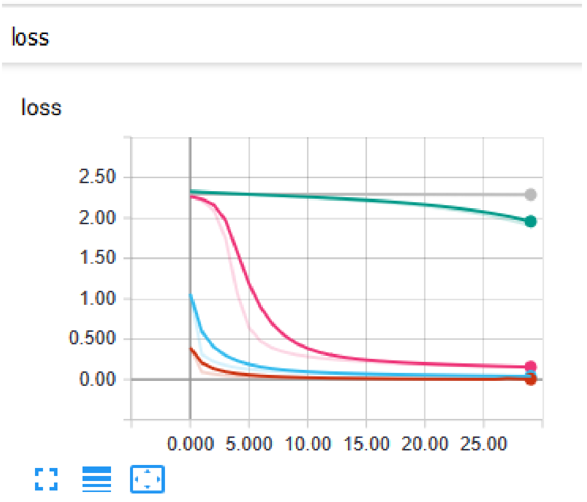
The pink line represents a learning rate of 0.001

The green line represents a learning rate of 0.0001

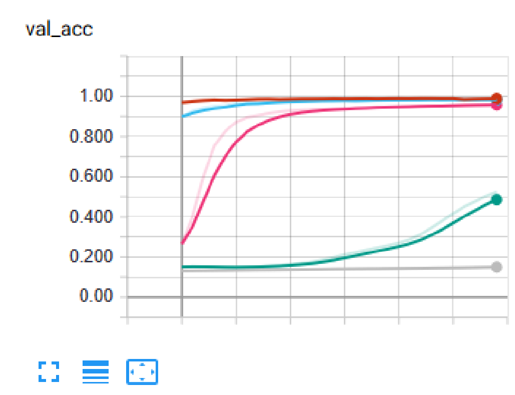
The gray line represents a learning rate of 0.00001

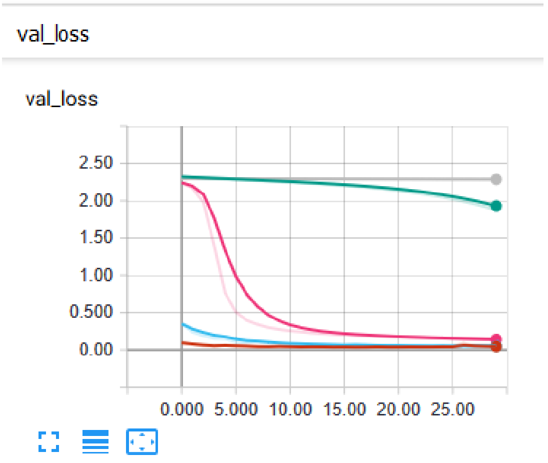
For training:





For testing:





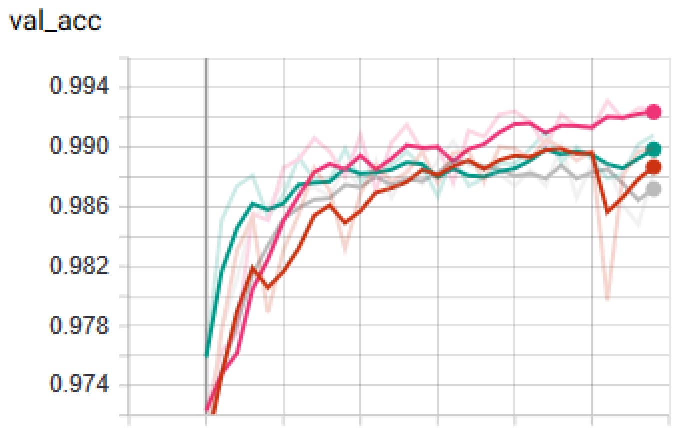
The chosen learning rates perform well for 0.1, 0.01, and 0.001. However, 0.0001 seems to be too small for the data available. The learning rate that performed the best was 0.1 with an accuracy of 0.9887 for the test data. 0.01 was fairly close with an accuracy of 0.9829, same with 0.001 with an accuracy of 0.9582. However, there was a very large dropoff with 0.0001 with an accuracy of 0.5248 and an even larger dropoff with 0.00001 with an accuracy of only 0.1503.

10:

In addition to the data collected from the earlier questions, I did a few additional tests to determine whether AdaDelta or SGD worked better. It seems conclusive that AdaDelta performed better than SGD, when the learning rate for AdaDelta was not defined, and the learning rate for SGD was 0.1 (the highest it received from testing, including leaving undefined). Where AdaDelta had an accuracy of 0.8976 for 30 epochs and SGD had an accuracy of 0.8709 for the mnist-fashion dataset.

Since the learning rates that I chose had 0.1 has the highest, I decided to do some additional testing with higher learning rates to see when the learning rate would be too high, and it would drop-off. The following graph shows that the optimal learning rate was 0.2, where 0.3 started to have a decline.

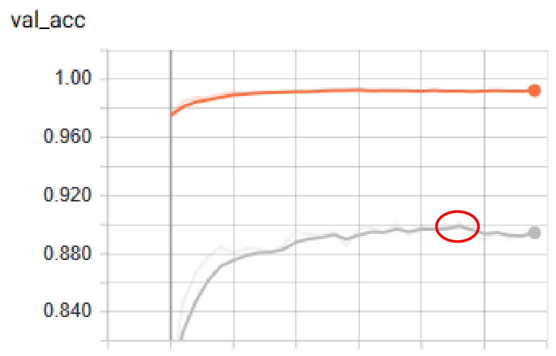
Red=0.1 Pink=0.2 Green=0.3 Gray=0.4



So, I learned that learning rates above 0.2 and learning rates below 0.2 would have lower and lower accuracy the further the learning rate strayed from 0.2.

The higher the learning rate the faster the data would converge into a value. Where with low learning rates, you notice a very steep accuracy rate, where it would continuously get better the more iterations there were.

I did notice that in some plots such as the plot in question 7 where the LeNet5 was trained using the fashion dataset, the data seemed to peak around the 25th epoch. Where after that point the accuracy dropped. The peak is marked with a red circle.


val_acc