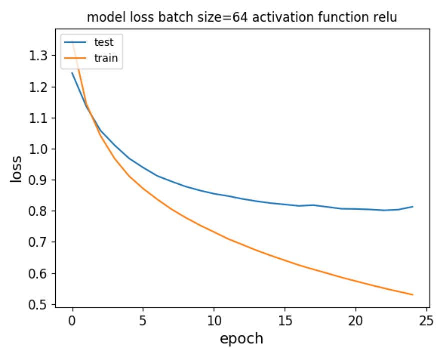
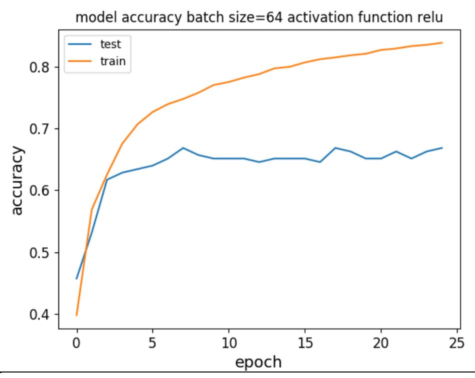
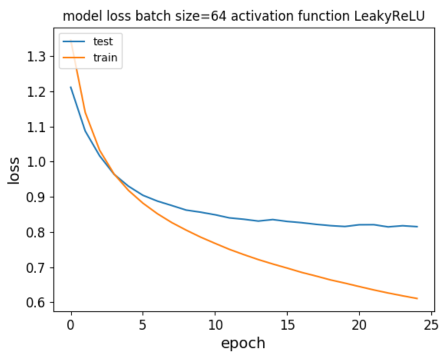
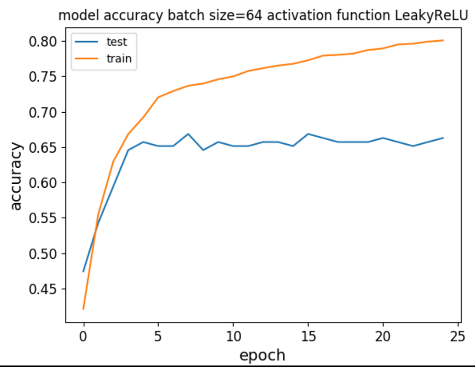
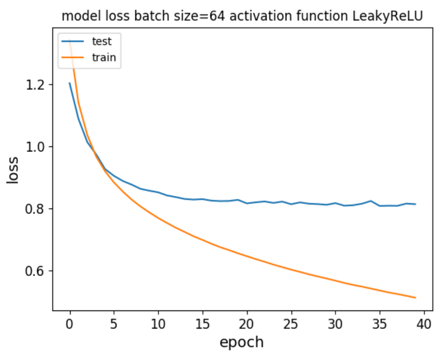


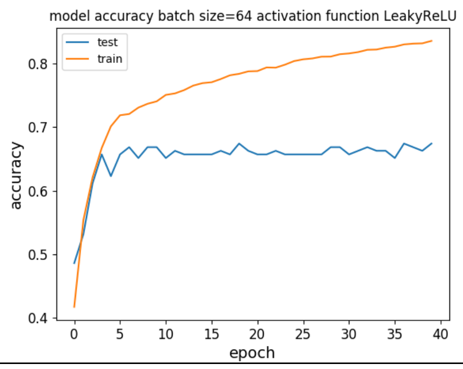




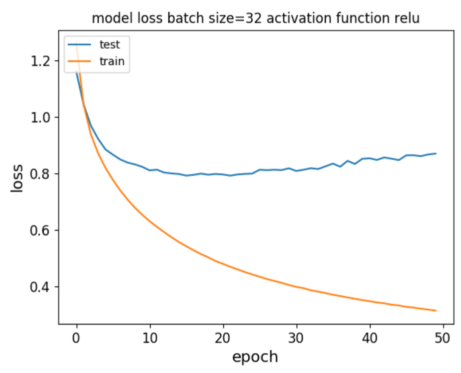


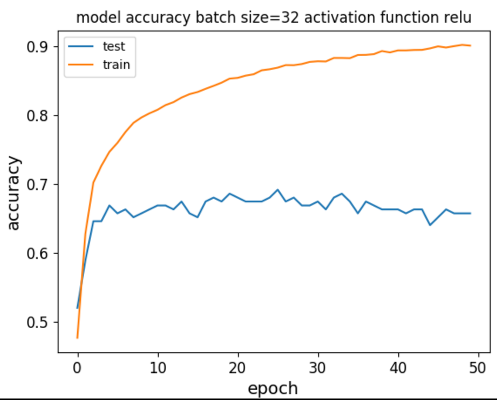
Task 2: Leaky ReLU multiplies the negative input by 0.01 instead of zero as in regular ReLU. This extends the range of ReLU so there are less dead neurons. This enhances performance of the network, as it improves gradient flow.



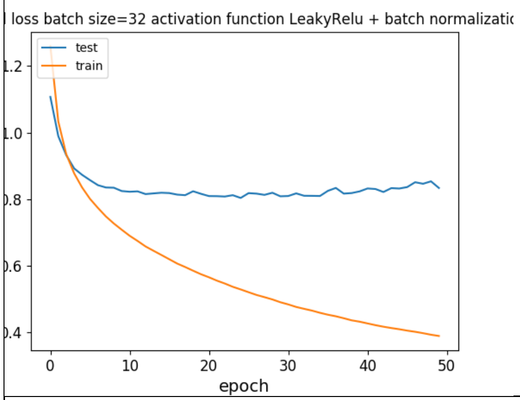


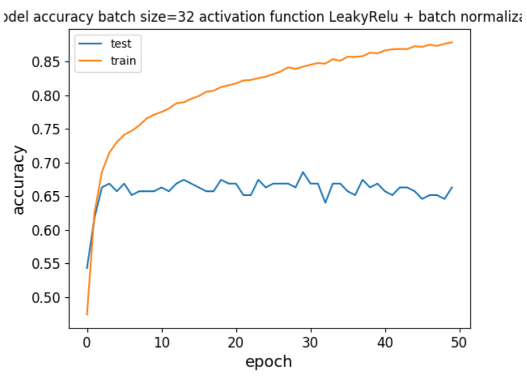
Task 3: The loss in the training set with 40 epochs is reduced, meaning the performance is improved in comparison to the 25. On the other hand, the test set is the same. The risk with training more epochs is overfitting on the training set that can eventually reduce the accuracy on the test set.





Task 4: Minibatch deals better with a smaller amount of data then the SGD. Because minibatch uses a small amount of data for each batch we keep the high accuracy with small fluctuation in the cost, while the SGD has more cost fluctuation. The mini batch is also more effective computationally, it takes less memory because it doesn't run on each sample separately like the SGD. As we can see in the graphs, when using a smaller batch (32) there is a performance enhancement in the training set, the accuracy increased in comparison to the 64.





As we can see in our results, the Batch normalization makes the networks work faster. In our graph we can see after very few epochs we got to an accuracy of over 0.65 and stays stable in the training set. We can see that in the test set we also have high accuracy. In previous examples it took longer to receive such accuracy.

Part 2:

The model has 8 layers, 5 convolutional and 3 dense layers.

In the convolutional layers we have filters of 64, 128, 128, 256, 256.
In the dense layers we have a filter of 512, 128, 4.

In a fully connected we will have all the parameters of each layer multiplied by the other layers. Here the advantage is that the number of parameters matches the number of filters times the kernels, meaning we get a smaller amount of parameters. Each filter is applied to all the layers in the picture vs each pixel separately. The parameters aren't dependent on the size of the input.

Yes, it is regulized by 'l2'

Filter = [64, 128, 128, 256, 256] Total params: 3,271,492 Trainable params: 3,271,332 Non-trainable params: 160 test loss, test acc: [7.5722535106113975, 0.41714287]	Filter = [32, 62, 62, 128, 128] Total params: 1,392,772 Trainable params: 1,392,612 Non-trainable params: 160 test loss, test acc: [4.9384700884137835, 0.25142857]

In the first one (with larger filters), the filters have a greater depth and can capture more information (it has a greater number of parameters as well), this is why we have a higher accuracy. Since we are filtering more features we can also see that the larger data has more test loss. Another problem with larger number of filters is that it can take longer to compute.