**Lab 2 - Report**

**Task 6 A**

**Answer)** The best results we were able to achieve was a validation accuracy of 0.83 and a validation loss of 0.4635 after 13 epochs. After that, the model starts to overfit which worsens the results.

**Task 6 B**

**Answer)** The model exhibits the same behavior for n\_base = 16. With n\_base = 8 the model performs slightly worse with the following best results: validation accuracy of 0.8 and a validation loss of 0.4941, at epoch 10. Thereafter it overfits, as the last two models.

Adding dropout improves the general performance of the network. The best results of the network is the following: validation accuracy of 0.835 and a validation loss of 0.4371, at epoch 34. This is comparable to previous models. However, the model does overfit slightly.

The model with 150 epochs overfits at epoch 32 with the following best results: validation accuracy of 0.8250 and a validation loss of 0.4284. The drop out makes the model overfit later.

**Task 6 C**

**Answer)** The model converges slower with decreased with a decreased learning rate. After 150 epochs, the model still had not converged. However, it did not overfit either.

Best results were a validation accuracy of 0.75 and a validation loss of 0.6636. The model exhibits the same behavior when increasing the number of epochs to 350. Then, the best results were achieved at epoch 349 with the following statistics: validation accuracy of 0.78 and validation loss of 0.5714.

**Task 6 D**

**Answer)** We could observe that the network trains much slower with bs (batch size) = 2 since half of the images are used in each epoch of training. The results improve somewhat, peaking at epoch= 149 with validation accuracy of 0.765 and validation loss of 0.5814. The gain is however not good enough to motivate the more than doubled computational time. The challenge here is to find a sweet spot between a high performance in terms of accuracy, and a fast network.

Bs = 4 experiences the same problems, with best results being a validation accuracy of 0.765 and a validation loss of 0.6363. Bs=8 was run in the previous task.

Thus, we could infer that larger batch size implies a slower network with increased performance.

**Task 6 E**

**Answer)** Using the following hyperparameters: base = 32, learning rate = 0.00001 and a bs = 8, with dropout applied, we get the following results for the different optimizers:

**Adam:** validation accuracy = 0.8350 and validation loss = 0.4481

**SGD**: validation accuracy = 0.5 and validation loss = 0.6931

**RMSprop:** validation accuracy = 0.8350 and validation loss = 0.4526

**Task 6 F**

**Answer)** The model seems to have difficulty learning from the hinge loss function. Validation accuracy and validation loss remains low and respectively high during the training. The best results were a validation accuracy of 0.4100 and a validation loss of 0.6958.

**Task 7 A**

**Answer)** See the submitted code.

**Task 7 B**

**Answer)** The optimal learning rate was found to be 0.00001 after testing.

**Task 7 C**

**Answer)** With a learning rate of 1 \* e^-5, the models train slower. Thus, it is unable to converge in 150 epochs due to it being so slow. However, it converges faster than the AlexNet model. The best results achieved was a validation accuracy of 0.7350 and a validation loss of 0.66.

**Task 7 D**

**Answer)** Increasing the number of feature maps lead to a decrease in performance with the results being: validation accuracy = 0.605 and validation loss = 0.6787.

Adding dropout increased the performance slightly with results being: validation accuracy = 0.76 and validation loss = 0.6654.

**Task 7 E**

**Answer)** The complexity of the networks is the main difference. LeNet being the simplest, then AlexNet which is a bit more complex and finally VGG16 that is the deepest.

VGG16 is the most modern of them all and has the best performance.

**Task 8**

**Answer)**

**Task 9**

**Answer)**

**Task 10**

**Answer)**