Jacob Dahlkvist, jad7543
Mervan Kaya, mk45596

HW1

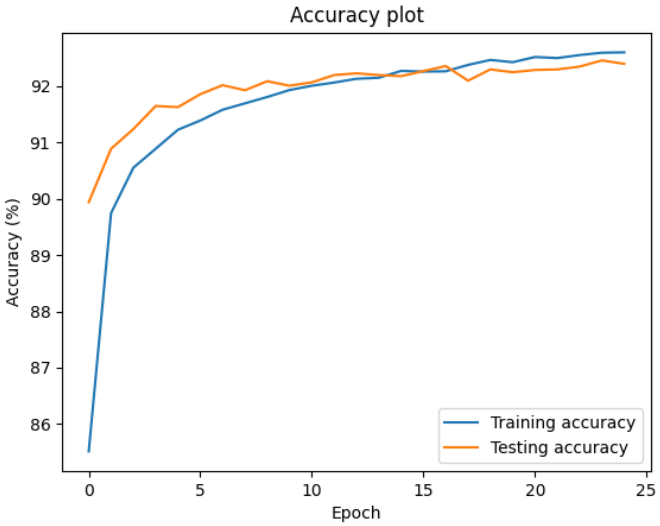
The work has been split between both group members fairly. We have learned what and how to use the convolutional layer, as well as how to communicate with super communities and how to run batch jobs.

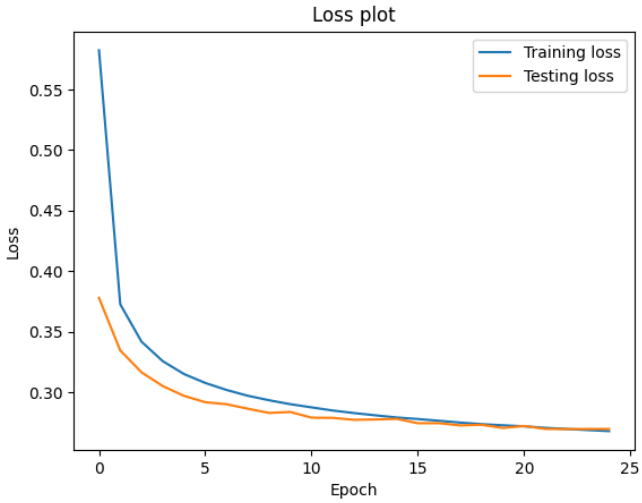
Problem 1

Q1.

This is done by creating a torch device variable called “device” which is set to “cuda:0” (line 34). This is then passed to the model using model.to() (line 56) and the images and labels using images.to() (line 83 and 126) and labels.to() (line 84 and 127).

Q2.





Jacob Dahlkvist, jad7543
Mervan Kaya, mk45596

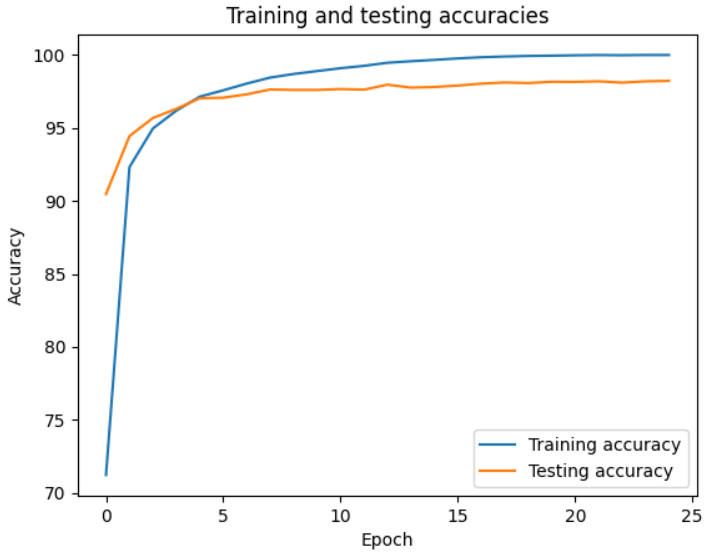
Q3.

Table 1

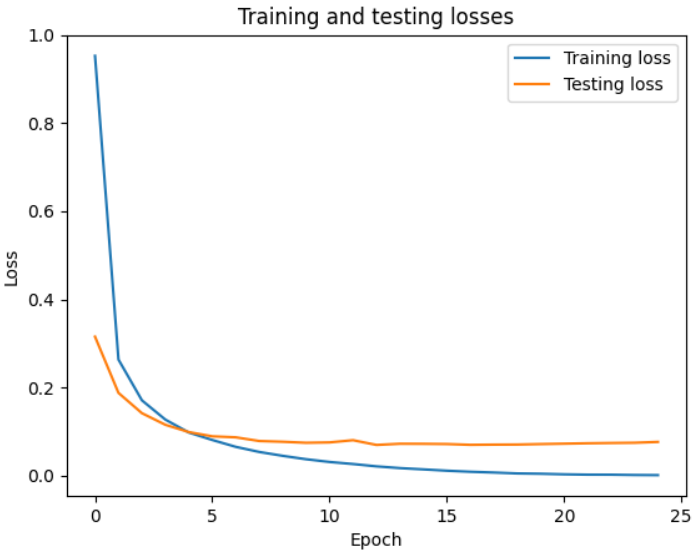
Training accuracy [%]	Testing accuracy [%]	Total time for training [s]	Total time for inference [s]	Average time for inference per image [ms]	GPU memory during training [MB]
92.62	92.40	188.09	0,82	0.08	655

Problem 2

Q1.

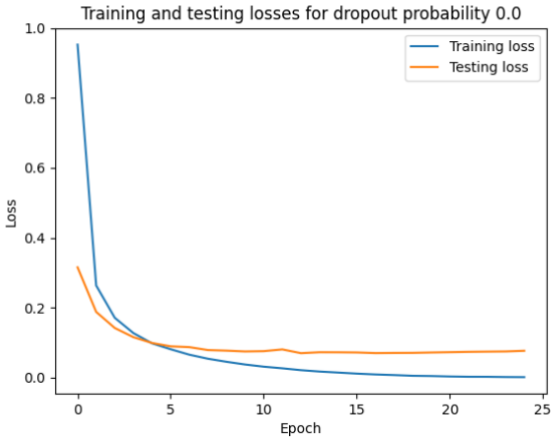


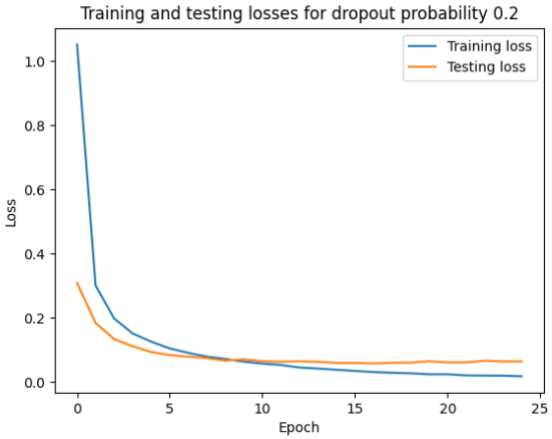
Jacob Dahlkvist, jad7543
Mervan Kaya, mk45596

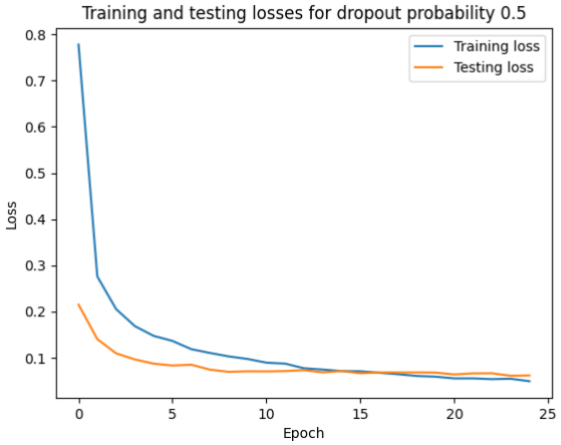


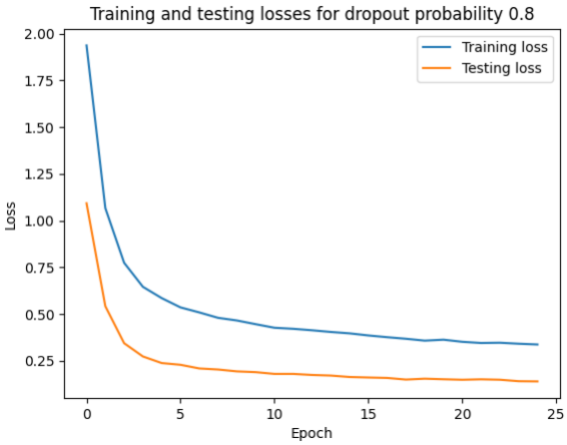
Yes, the model does overfit as the testing losses diverge from the training losses, while the training accuracy still improves.

Q2.









Jacob Dahlkvist, jad7543
Mervan Kaya, mk45596

From the plots above, we can see that the best dropout level will be for probability 0.5. This is because this is the level where the testing and training losses are almost the same. In the other plots, either the training losses are higher than the test losses which is indicative of underfitting (0.8) or the test losses are higher than the training losses which is indicative of overfitting (0.0, 0.2).

Q3.

Dropout	Training Accuracy [%]	Testing Accuracy [%]	Total time for training [s]	First epoch when the model reaches 96% training accuracy
0.5	98.27	98.33	261.21	8
0.5 + Norm	98.52	98.35	369.64	6

From the table above, we see that normalizing the data will generate better training and testing accuracies, and it reaches a higher accuracy quicker. However, normalizing the data greatly increases the total training time.

Problem 3

Q1.

Model name	MACs (M)	FLOPs (M)	#params (M)	Model size(MB)	Saved model size (KB)
SimpleCNN	3.87	1.93	5.02	0.48	203

There is a big difference between estimated size of the model (0.48MB) and the saved version of the file (203KB=0.203MB). The difference in size between estimated size of the model and the saved version of the model is likely due to the fact that the torchsummary function is reporting the total size of the model in memory, while the saved version of the model Is just the serialized model. It is also compressed.

Q2.

Model name	MACs (M)	FLOPs (M)	#params (M)	Model size(MB)	Saved model size (KB)
SimpleCNN	3.87	1.93	5.02	0.48	203
SpiderNET	7.46	3.73	104	0.74	419

Jacob Dahlkvist, jad7543
Mervan Kaya, mk45596