The code was written successfully, and the results produced from running the code closely resembled that which was shown in the assignment description.

The data loaded by the script was manipulated to be zero-mean, and was also divided by the standard deviation of the set (and not 255) as described in the assignment pdf. And for all experiments in this assignment, the random seed was not fixed, and data was shuffled between each epoch during training.

1 Correctness of analytical gradient

To verify that the analytical gradient was indeed correct, the analytically calculated gradient was compared against the numerically computed gradient, using the method

$$\text{alignment} = \frac{|g_a - g_b|}{\max(1e-6, |g_a| + |g_b|)} \tag{1}$$

and the computed alignment was 4.88e-07 for W_1 and W_2 respectively, using the first full 100 samples in the `data_batch_1` dataset. The computed loss using this batch of data was $2.298051095211594e-07$ and $3.821336874730926e-07$ respectively, which is an acceptable number, if compared to what was achieved in the first assignment. Additionally, the analytical gradient was used to train a network using learning rate $\eta = 0.001$ and no regularisation ($\lambda = 0$) and over 200, which resulted in plots seen in Figure 1. As expected the network overfits to the training data, and the training data accuracy keeps increasing, in comparison to validation and test accuracy that saturates.

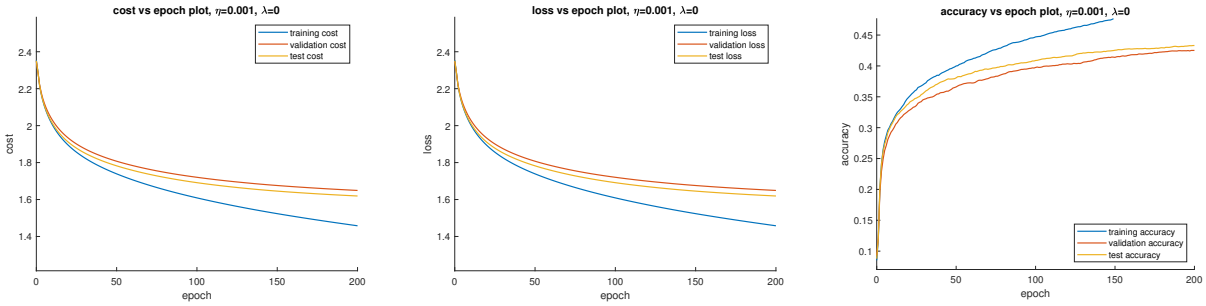


Figure 1: Plots of cost, lost and accuracy as function of trained epochs. Trained using a 2 layer network with $\eta = 0.001$, $\lambda = 0$, over 200 epochs.

2 Cyclical learning rates

The script was modified for cyclical learning rates in triangular form, with default settings $\eta_{min} = 10^{-5}$, $\eta_{max} = 0.1$. A cost/loss/accuracy data point was saved for every completed epoch, thus a run with $\eta_s = 500$, batch size 100 and one full cycle would correspond to $10+1$ saved data points (including the origin).

The plots for $\eta_s = 500$, batch size 100, 1 cycle, can be seen in Figure 2. All three plots look largely the same as those supplied in Figure 3 in the assignment PDF. We clearly see that the network begins to overfit the training data, as its accuracy keeps on increasing, while the validation and testing accuracies start to plateau. The reason the curves are not smooth is because only 11 data points are used to plot the curves.

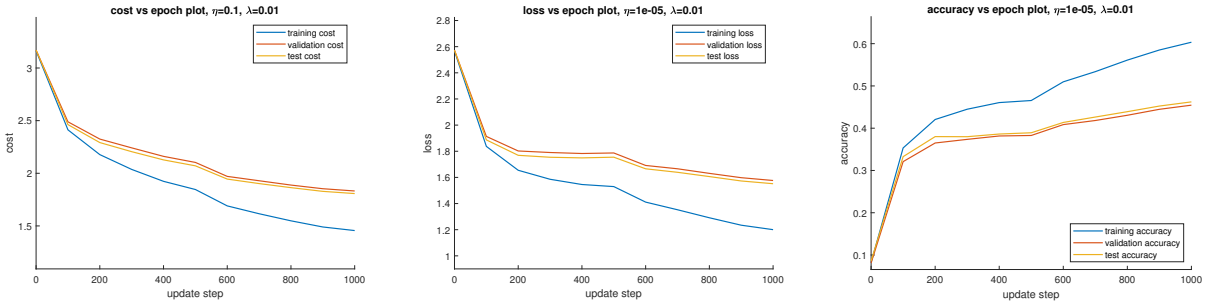


Figure 2: Plots of cost, lost and accuracy as function of number of updates, using cyclical learning. Trained using a 2 layer network with $\eta_{min} = 10^{-5}$, $\eta_{max} = 0.1$, $\eta_s = 500$, $\lambda = 0.1$. The final test accuracy was 46.23%.

3 Coarse search to set λ

To do the coarse grid search for the regularisation parameter λ, the training algorithm was run on 50 different λs, and then the best validation accuracy corresponding to the λ was saved. The λs were generated using the `logspacel_min, l_max`, where in my case for the coarse search $l_{min} = 10^{-5}$, $l_{max} = 0.1$. As before, the range for η was $[10^{-5}, 0.1]$, and $n_s = 2N//n_{batch}$, $n_{batch} = 100$, and the network was trained using 2 cycles.

Table 1: Best 3 accuracies when running a coarse grid search for regularisation parameter λ.

λ	accuracy
0.00232995181051537	51.74 %
0.00719685673001152	51.60 %
0.00339322177189533	51.58 %

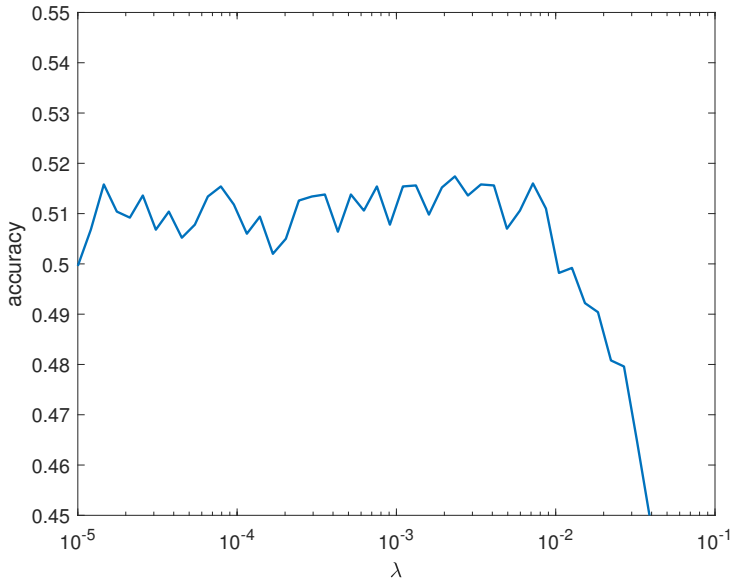


Figure 3: Accuracy vs λ plot when doing a coarse grid search for best regularisation parameter λ.

Clearly, the coarse search states that the network performs the best for λ in the magnitude of 0.001. Most accuracies are above 50%, and close to 52%, which conforms to what was stated in the assign-

ment pdf. The jaggedness of the accuracy vs λ plot is likely explained by the random initialisation, and shuffling of data in the mini batch GD.

4 Fine search to set λ

The fine search was performed using the same parameters as the previous section, with the exception of course being that the range of λs being examined being different. For this task, the interval for the λ grid search was set to a linear space for between 0.001 and 0.008, with a total of 50 examined lambdas. The training was performed over 2 cycles.

Table 2: Best 3 accuracies when running a fine grid search for regularisation parameter λ.

λ	accuracy
0.00157142857142857	51.98 %
0.00428571428571429	51.94 %
0.00457142857142857	51.82 %

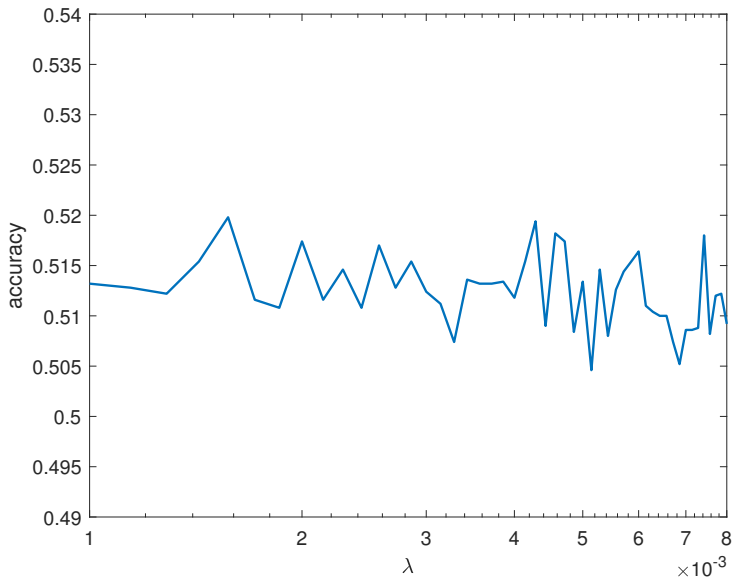
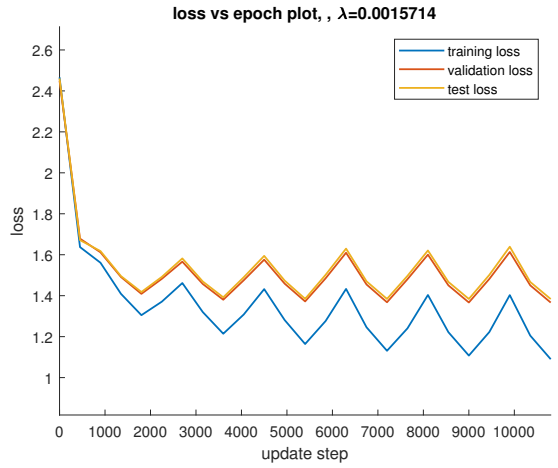


Figure 4: Accuracy vs λ plot when doing a fine grid search for best regularisation parameter λ.

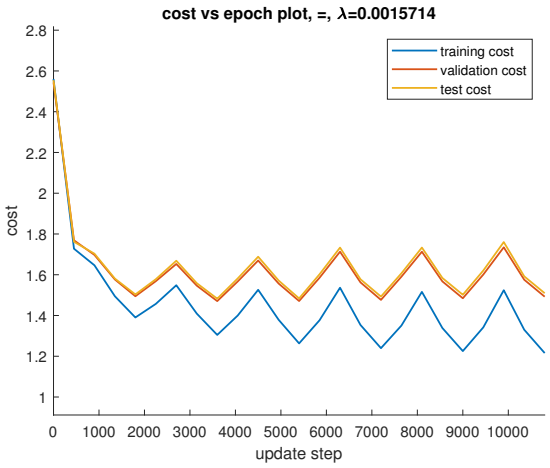
As expected, the accuracies in the examined interval did not differ too much, and the peak performance was reached at 51.98 % accuracy. As earlier, the jaggedness of the curve is explained by the random initialisation, and shuffling of data in miniGD. Thus, the optimal value of λ might not really be the best possible, but we shall use it for the last task in the assignment anyway.

5 Performance using the best λ

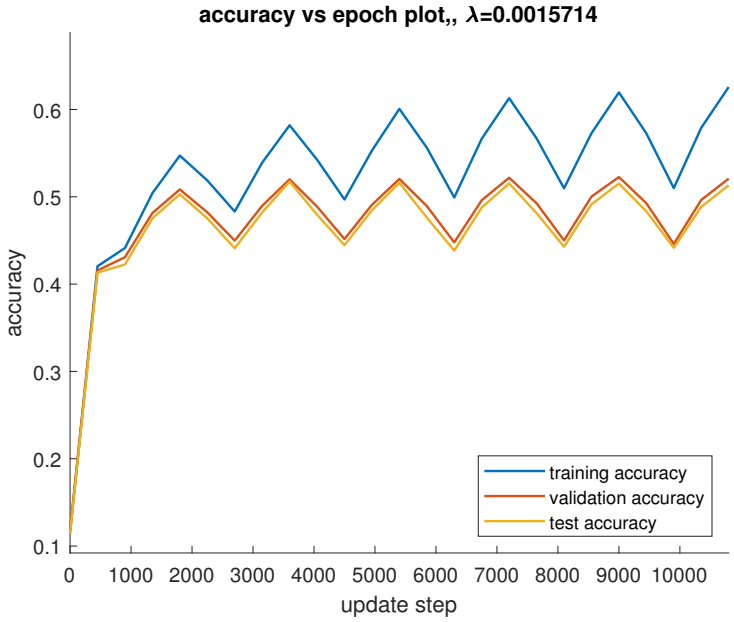
Using the result from the previous section, the network was trained using 6 cycles, and $\lambda = 0.00157142857142857$. The final test accuracy became 51.46 %.



(a) Loss l as function of number of epochs for training.



(b) Cost J as function of number of epochs for training.



(c) Accuracy as function of number of epochs for training.

Figure 5: Plots of loss and cost function, as well as accuracy of trained network over 6 cycles with cyclical learning rate, and $\lambda = 1.5714E-3$.

As can be observed in Figure 5, the accuracy peaks at the end of the cycle, and cost and loss has a minimum at each cycle end. The accuracy achieved on the test set is somewhat smaller than what was achieved on the validation set, but still very good.