*DD2424/FDD3424 Deep Learning in Data Science, Assignment 1, 2021*
*Wai-Hong Anton Fu, whafu@kth.se*

## Baseline Implementation

| Cases | Test Accuracy (%) |
|---|---|
| 1 | 27.88 |
| 2 | 38.80 |
| 3 | 39.29 |
| 4 | 37.57 |

Table 1: The final test accuracy for each case. Case 3 is the best one, while case 1 is the worse.

The baseline implementation was successful for my case, which was verified by comparing the analytic gradients (my implementation) to the numerically calculated gradients. *Python* was the programming language of choice.

For the assignment, we were asked to investigate four distinct cases:

- Case 1. lambda=0, n_epochs=40, n_batch=100, eta=0.1
- Case 2. lambda=0, n_epochs=40, n_batch=100, eta=0.001
- Case 3. lambda=0.1, n_epochs=40, n_batch=100, eta=0.001
- Case 4. lambda=1, n_epochs=40, n_batch=100, eta=0.001

The test accuracy for each cases can be seen on table 1. Note that cases 1 and 2 lack regularization terms, which allows the weights of the network to be as complex as possible. Furthermore, case 1 have a very high eta (learning rate) of 0.1 which prevents the learning from converging across the epochs, whereas Case 2 did not. This can be seen on figure 1.

Cases 3 and 4 have both better regularization terms of 0.001. However, Case 3 have a more healthy regularization term (lambda) of 0.1, which will prevent the network from overfitting. But the lambda of case 4 is too large, which caused the network to be very simply and underfit (we can see that case 4 converges very fast but to a subpar result). This is reflected on the accuracy. Case 4 have an accuarcy of 37.45% which is worse than case 2 (38.70%) and case 3 (39.29%) due to underfitting. Case 3 is best because of the balance between the lambda and learning rate. Case 1 is worse, likely due to its inability to converge.

The learnt W matrix for each case is visualised on figure 3. See see the better the model is, the more distinct each image gets.

Note the differences between the cost function and loss function (the cost function is the loss function but with the regularization term added to it) on figures 1 and 2. We see that the cost function shows better generalization, which can be seen on the fact that the training loss and validation loss is closer to eachother. Naturally, the cost function and loss function is the same for case 1 and case, due to the lack of lambda.

**Conclusion**: The results highlights the importance of a well picked eta and lambda. If the eta is to high, then the learning might overshot and never diverge, while a lower eta is more likely to converge, although

slower. Adding a lambda term will help regularize the network, which will prevent it from overfitting, but if the lambda term is to large, then the network might be too simple and underfit.

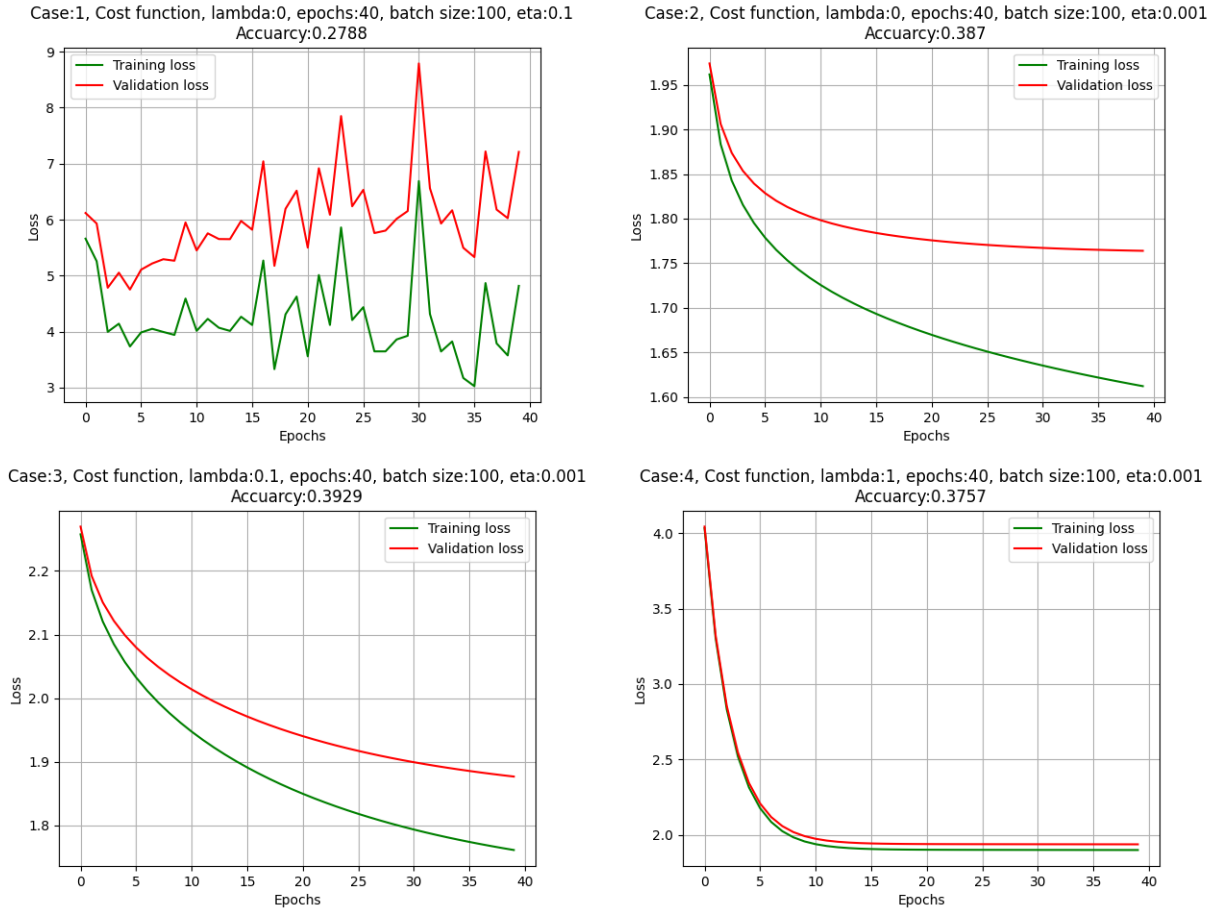


Figure 1: Shows the cost function across the epochs for each cases.

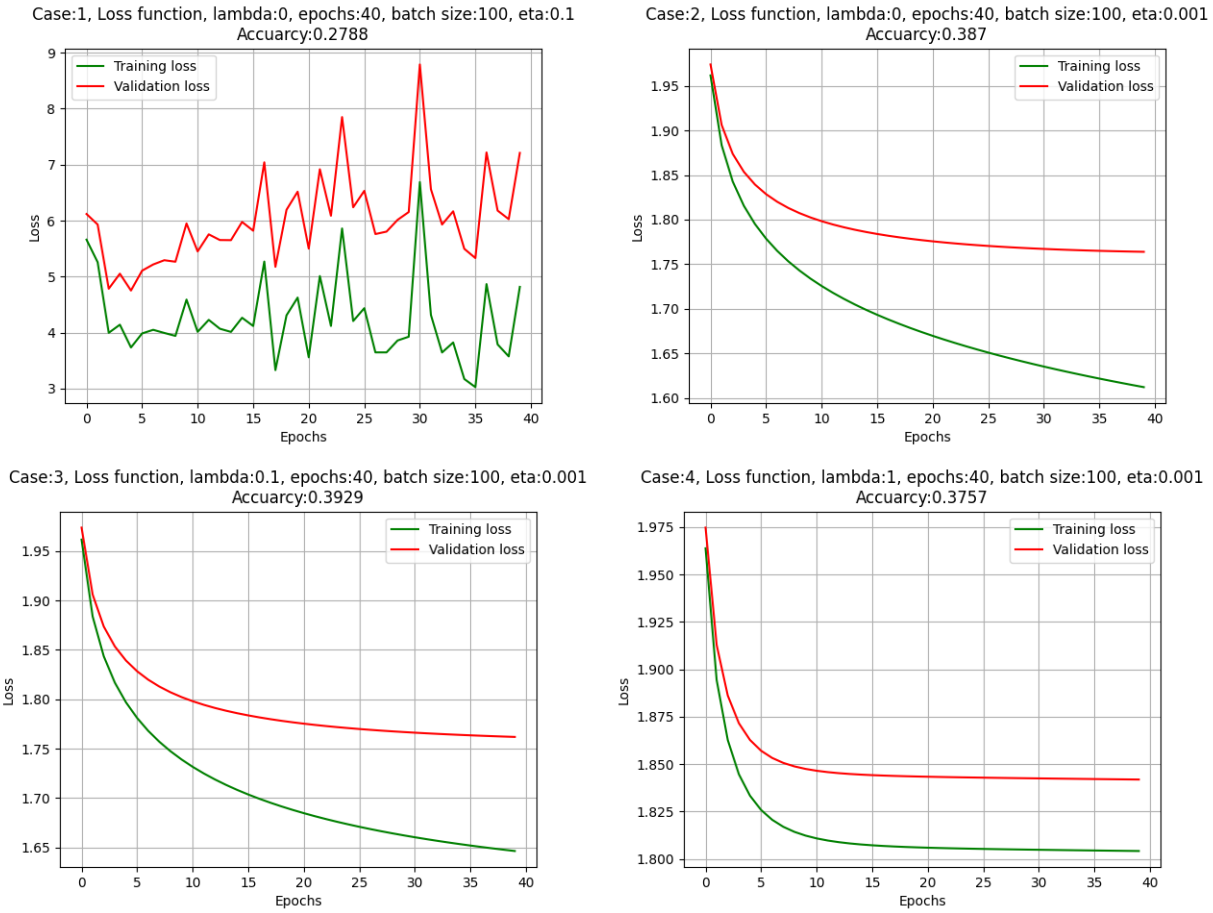


Figure 2: Shows the cost function across the epochs for each cases.

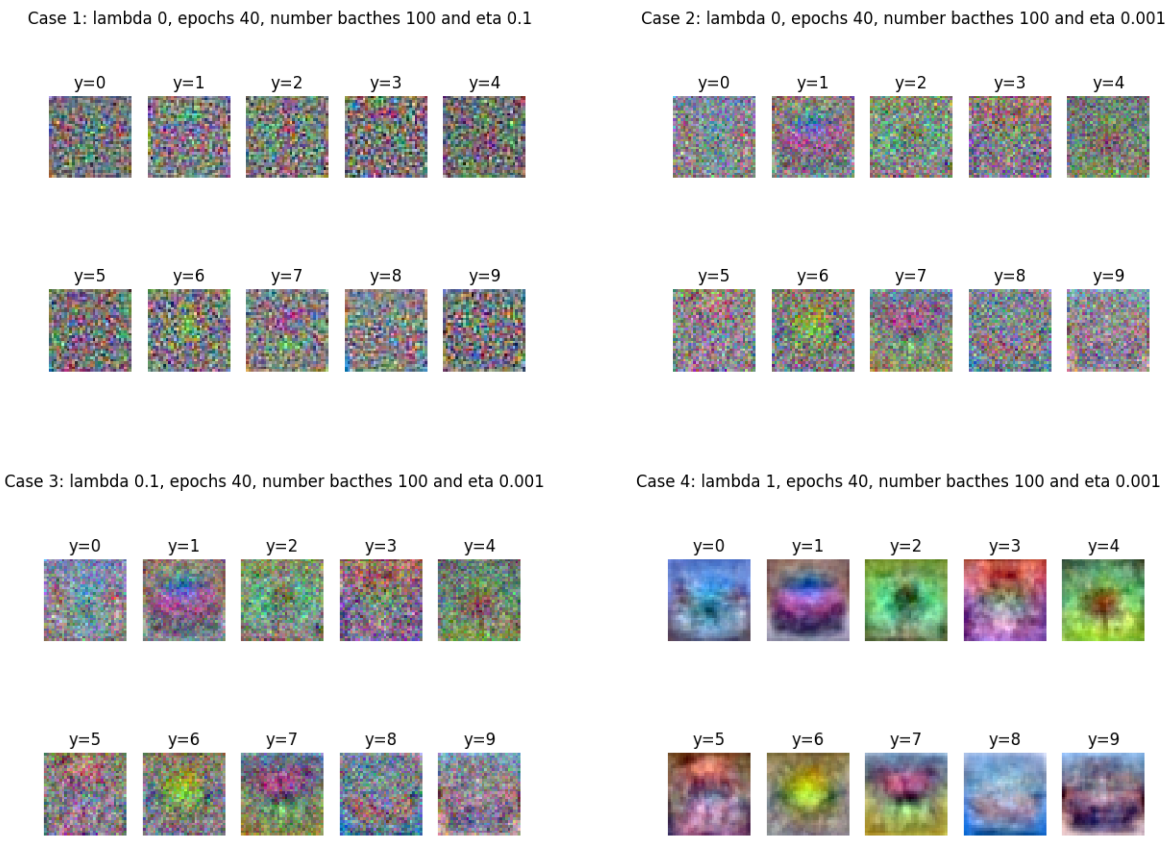


Figure 3: The learnt W matrix visualized as class template images. This is shown for all cases.