Deep Learning in Data Science
DD2424

Assignment 1 (Basic)

Report

Prashant Yadava

yadava@kth.se

05.04.2024

Gradient Descent implementation

Assignment 1's implementation was done in Python. The analytical gradient computation function was also successfully implemented. Using NumPy's in-built assert *assert_array_almost_equal*, the computed analytical gradient was found to be 'almost' equal to the provided numerical gradient function, as the asserts did not throw any errors.

```
grad_W_test_ana, grad_b_test_ana = ComputeGradients(train_X_norm[:, :2], train_Y[:,:2], P, W, lamda=0)
grad_W_test_num, grad_b_test_num = ComputeGradsNum(train_X_norm[:, :2], train_Y[:,:2], P, W, b, lamda=0, h=1e-6)


npt.assert_array_almost_equal(grad_W_test_ana, grad_W_test_num)
npt.assert_array_almost_equal(grad_b_test_ana, grad_b_test_num)
```

Classification Results

Next, we present the results from the training runs, including the loss computations in training and validation runs.

Table 1 shows the parameters chosen for model training and the plots showing the corresponding training and validation loss.

Table 1: Model Parameters settings

Figure	Run #	Parameter Settings
Figure 1	Run 0	lambda=0, n epochs=40, n batch=100, eta=.1
Figure 3	Run 1	lambda=0, n epochs=40, n batch=100, eta=.001
Figure 4	Run 2	lambda=.1, n epochs=40, n batch=100, eta=.001
Figure 5	Run 3	lambda=1, n epochs=40, n batch=100, eta=.001

Figure 1 demonstrates a very unstable model learning happens when a high learning rate is chosen.

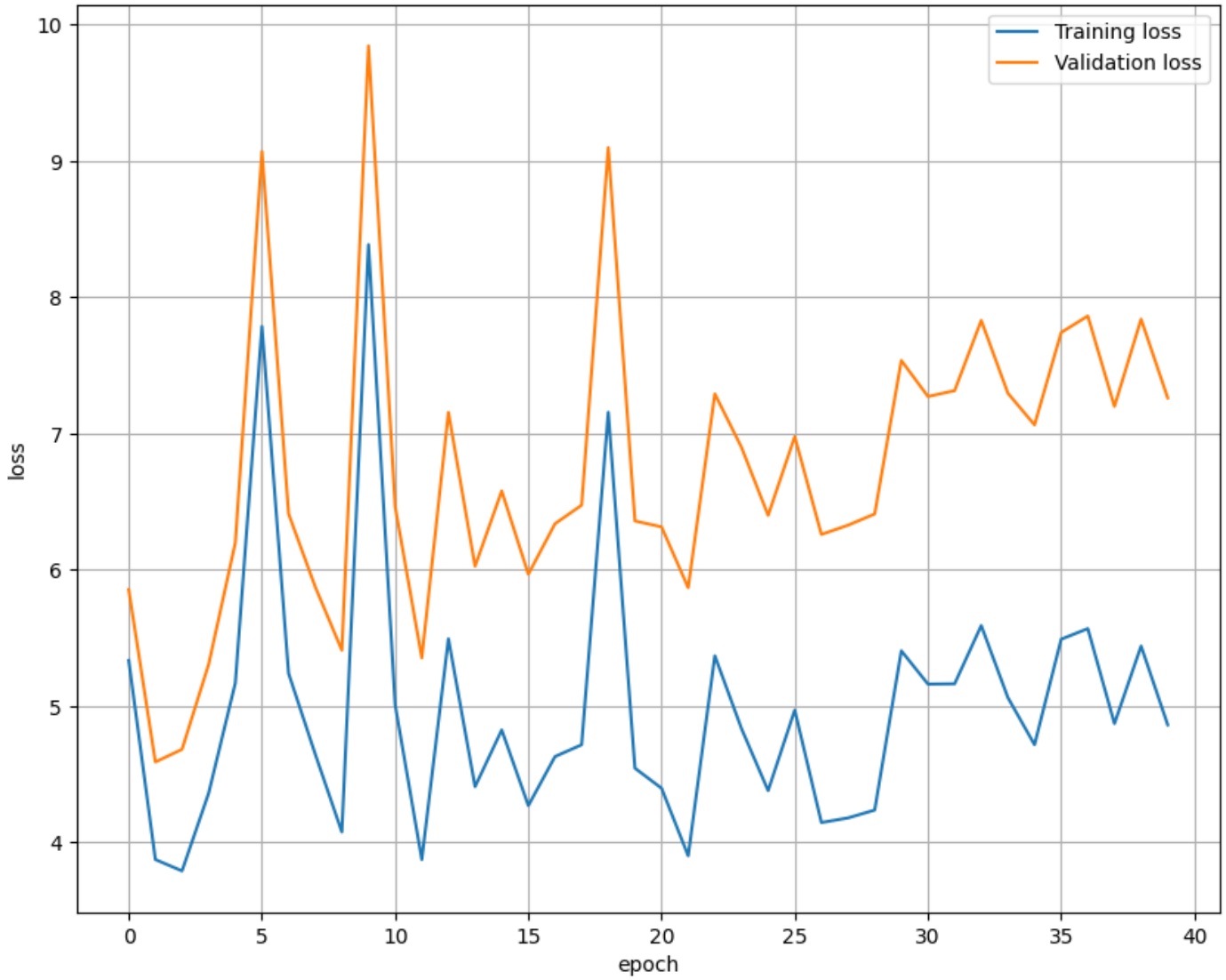

Figure 1: Model training and validation errors from training run 0

After this round, the following accuracy related statistics were observed:

Network parameters settings round: 0
Mean Train accuracy: 0.3584
Mean Validation accuracy: 0.23559999999999998
Mean Test accuracy: 0.23789999999999994
Std-dev Train accuracy: 0.0
Std-dev Validation accuracy: 2.7755575615628914e-17
Std-dev Test accuracy: 5.551115123125783e-17

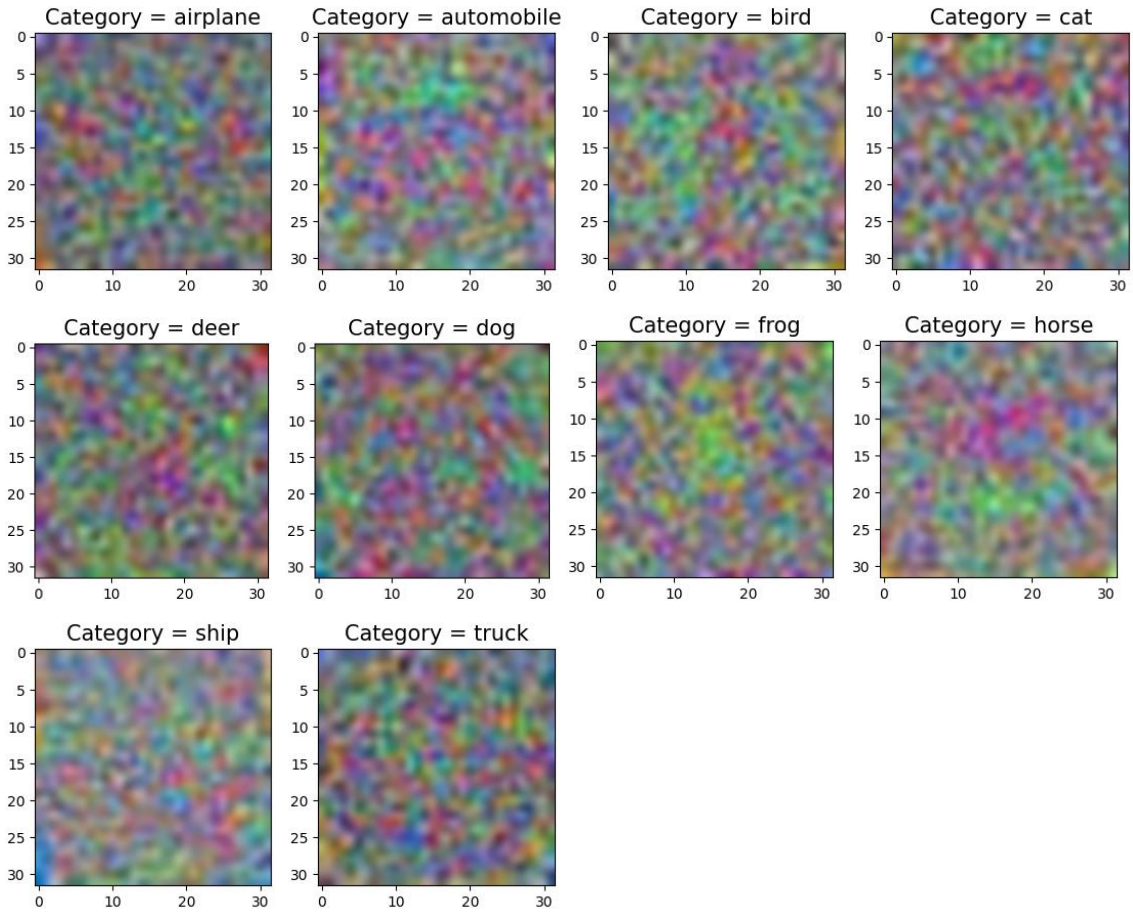

Figure 2: Images showing the learnt weight matrix after training run 0

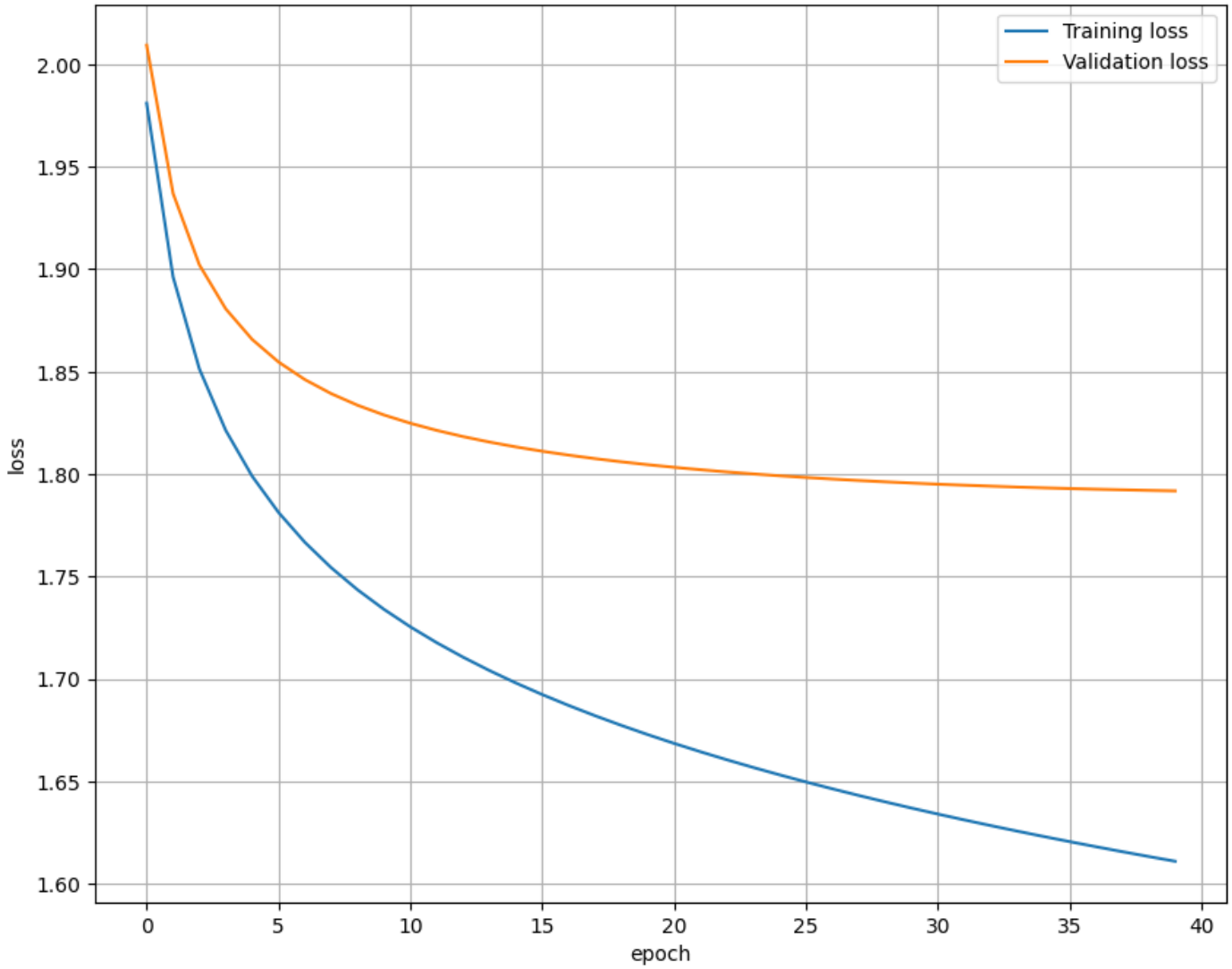

Figure 3: Model training and validation errors from training run 1

After this round, the following statistics for model accuracies were observed:

Network parameters settings round: 1
Mean Train accuracy: 0.4566
Mean Validation accuracy: 0.38710000000000006
Mean Test accuracy: 0.3921
Std-dev Train accuracy: 0.0
Std-dev Validation accuracy: 5.551115123125783e-17
Std-dev Test accuracy: 0.0

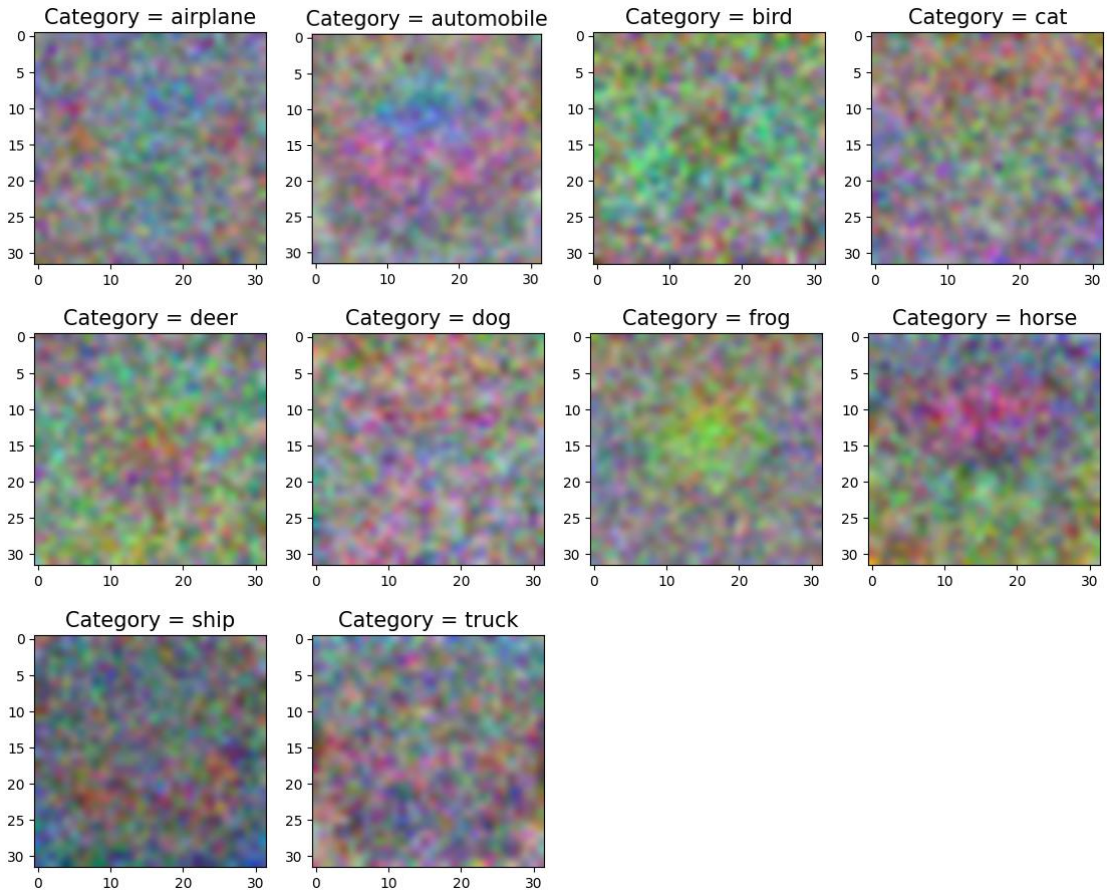

Figure 4: Images showing the learnt weight matrix after training run 1

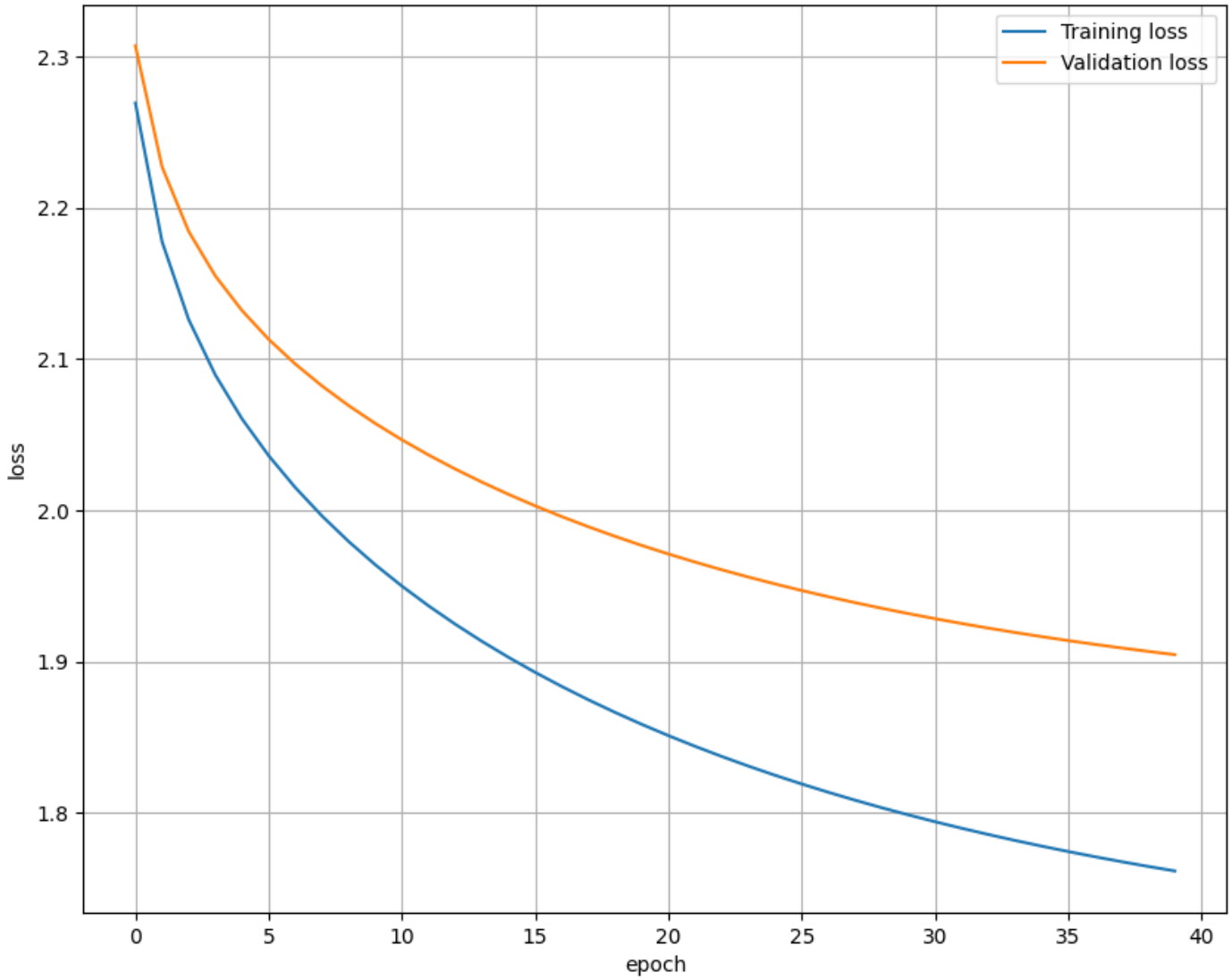

Figure 5: Model training and validation errors from training run 2

Network parameters settings round: 2
Mean Train accuracy: 0.4492
Mean Validation accuracy: 0.38520000000000004
Mean Test accuracy: 0.392
Std-dev Train accuracy: 0.0
Std-dev Validation accuracy: 5.551115123125783e-17
Std-dev Test accuracy: 0.0

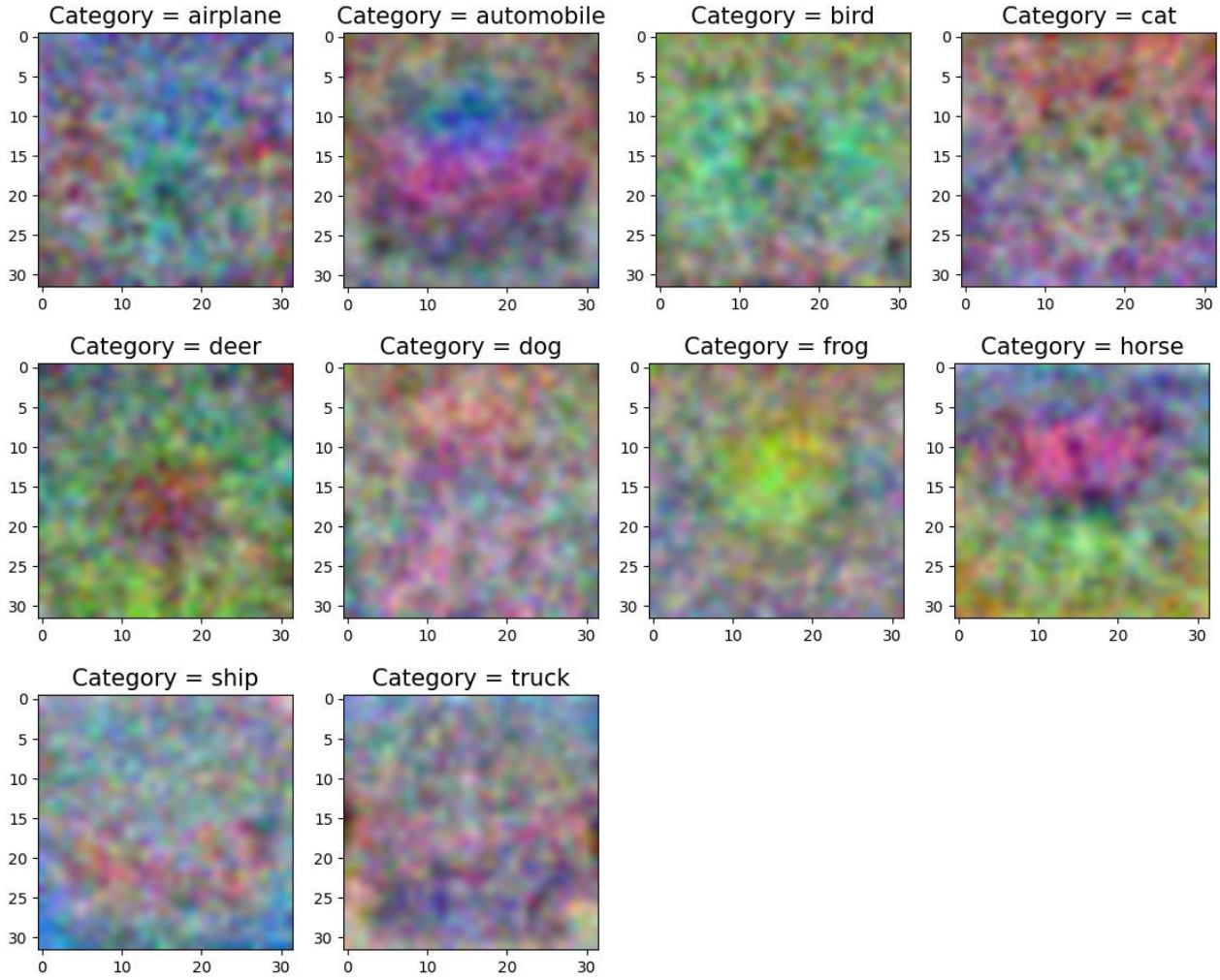

Figure 6: Images showing the learnt weight matrix after training run 2

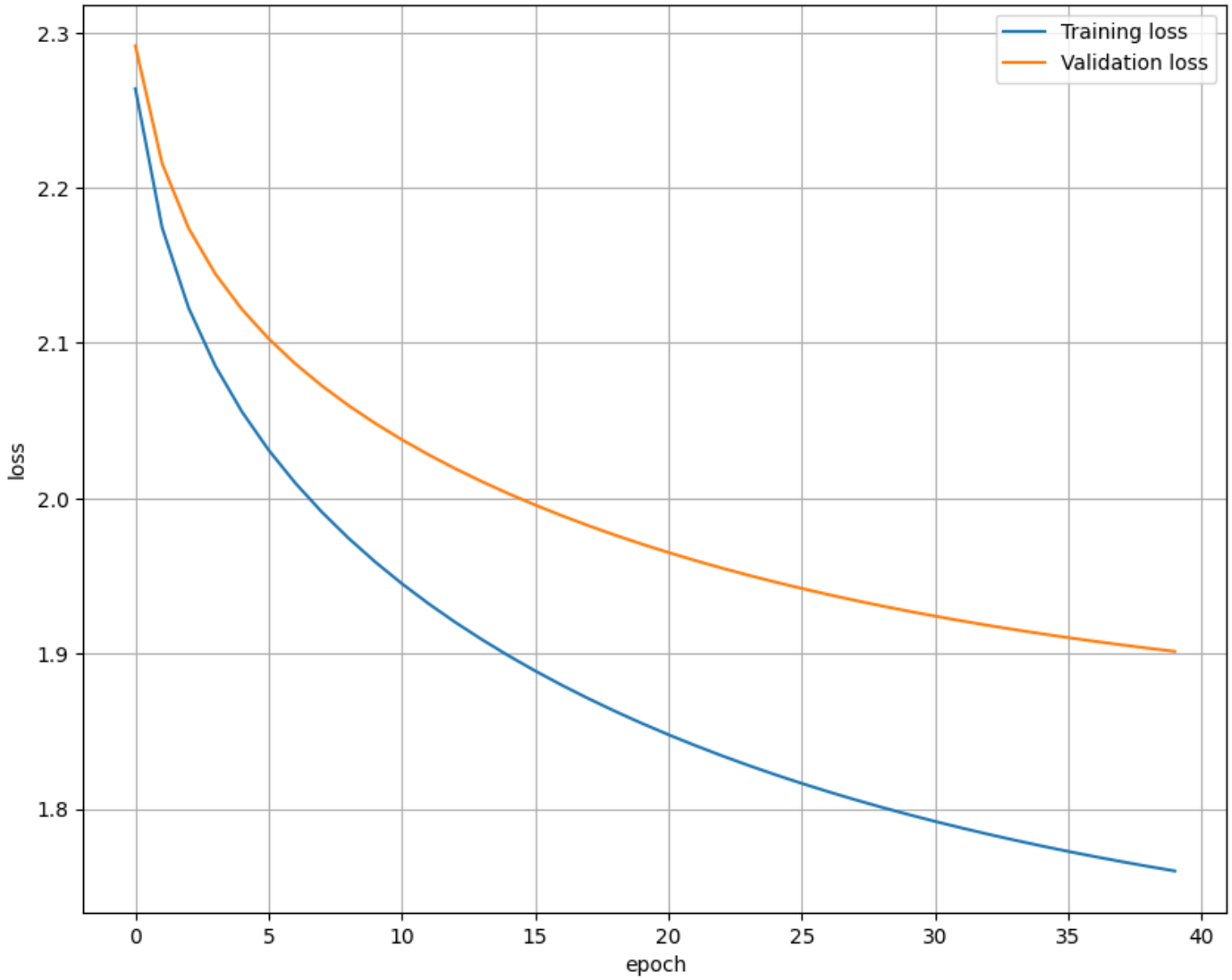

Figure 7: Model training and validation errors from training run 3

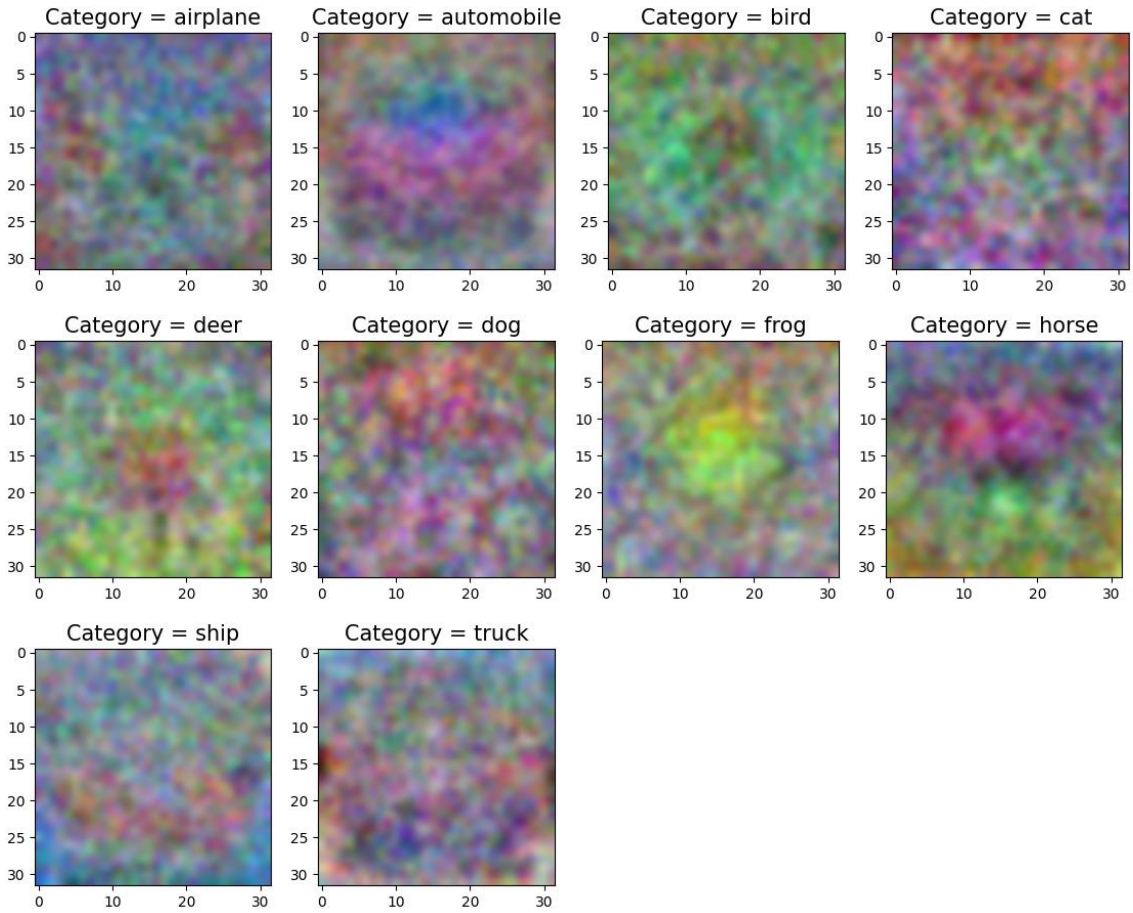

Figure 8: Images showing the learnt weight matrix after training run 3

Network parameters settings round: 3
Mean Train accuracy: 0.4483
Mean Validation accuracy: 0.3872999999999999
Mean Test accuracy: 0.39109999999999995
Std-dev Train accuracy: 0.0
Std-dev Validation accuracy: 5.551115123125783e-17
Std-dev Test accuracy: 5.551115123125783e-17

Increasing the regularization is seen to lead to a smoother learned image.

However, one must be careful with setting the value of regularization is set too high. This is because it heavily penalizes higher weights and has detrimental effects on the model's performance. It could lead to underfitting where the model's complexity is significantly reduced, or it's generalization capacity decreases on unseen data, causing the model not to be able to learn much.