DD2424 Deep Learning – Assignment 3

Testing Gradient Implementation

To test and confirm that the analytical computations of the gradient were correct, the analytical gradient was compared to the numerical version as in assignment 1 and 2 (computed with the centered difference method). The relative error was computed, according to the formula in figure 1, for six different number of layers, images and image dimensions. The results of the tests are shown in table 1.

$$\frac{|g_a - g_n|}{\max(\text{eps}, |g_a| + |g_n|)}$$

Figure 1. The relative error between a numerically computed gradient value g_n and an analytically computed gradient value g_a

Layers	Nr of images	Image dimension	λ	Relative error (W1,W2,..)	Relative error (b1, b2,..)	Relative error (beta1…, gamma1…)
2	10	100	0	1.2207e-08, 4.3927e-10	6.1402e-13, 5.1678e-10	3.3081e-11, 4.4113e-11
2	20	200	0	5.8966e-09, 5.7114e-10	4.5698e-13, 4.4589e-10	6.5896e-11, 9.4194e-11
2	10	100	0.5	9.6673e-09, 2.6463e-10	6.1402e-13, 5.7485e-10	1.6429e-10, 1.3693e-10
3	10	100	0.5	8.0718e-09, 5.7982e-09, 5.2872e-10	1.1220e-12, 1.2781e-12, 7.9331e-10	7.5455e-10, 2.9468e-10, 1.1967e-09, 3.0305e-10
4	10	200	0.5	0.0109, 5.9118e-09, 2.4525e-09, 4.5829e-10	8.9996e-13, 8.5331e-13, 5.9825e-13, 4.4584e-10	0.0032, 7.9234e-10, 6.5280e-10, 1.2072e-09, 8.5557e-10, 4.8464e-10

Table 1. The conducted gradient tests for a network (50 hidden nodes in all hidden layers) with batch normalization. Note that step length of h=0.0001 was used in the numerical computations.

One may observe that the relative error is < 1e-7 in practically all test cases, which indicate that the gradient implementation is correct.

To further check if the gradient implementation was correct, a 3-layer network with batch normalization was trained on a small amount of training data (100 images) without regularization (λ = 0) for one cycle with batch size 100 and cyclical learning rate (n_s = 2250, eta_min = 0.00001, eta_max = 0.1). The resulting training- and validation loss is shown in figure 2. As can be observed, the overfitting is significant, which indicate that the gradient computations and mini-batch gradient decent algorithm are both okay.

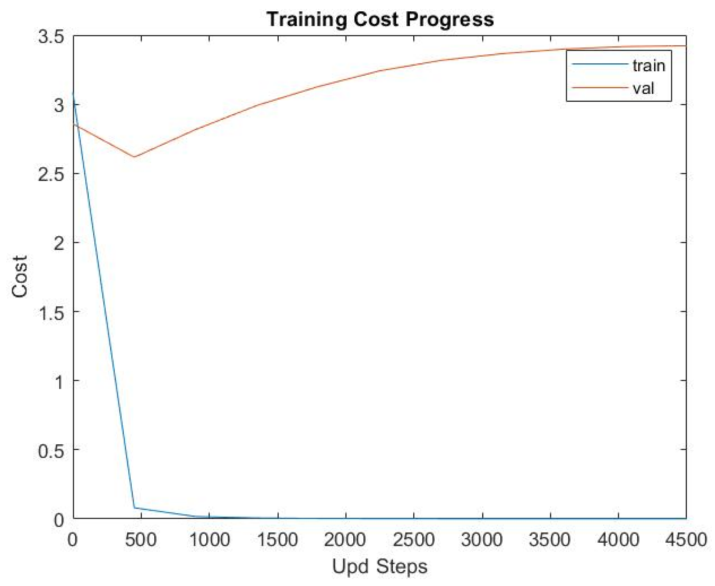

Figure 2. Training the 3-layer network without regularization to see if it can be overfitted.

Three Layer Network with and Without Batch Normalization

A three layer network with 50 and 50 nodes in the first and second hidden layer respectively, was then trained with the following hyper-parameter setting: batch size (n_batch) = 100, eta_min = 1e-5, eta_max = 1e-1, lambda=0.005, two cycles of training and n_s = 5 * 45,000 / n_batch. 45 000 images were used during training and 5000 images for validation. He initialization was used, and training data was randomly shuffled after each epoch. This configuration achieved 52.37% test accuracy and the cost during training is shown in figure 3. Using the same hyperparameter configuration but with batch normalization, yielded a test accuracy of 53.79 % and the corresponding cost plot can be seen in figure 5.

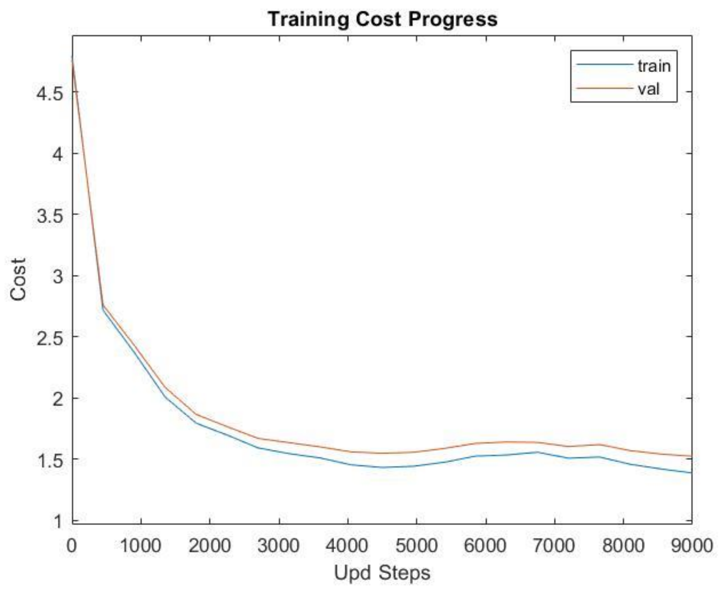

Figure 3. An illustration of the training- and validation cost for the 3-layer network without batch normalization

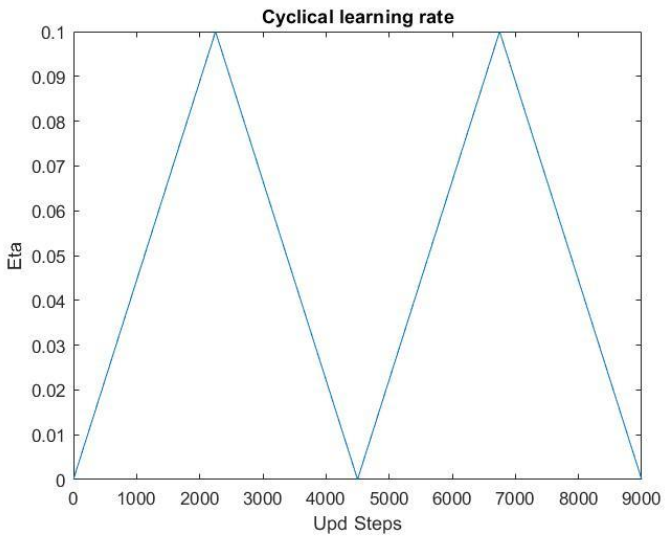

Figure 4. An illustration of the cyclical learning rate for the 3-layer network

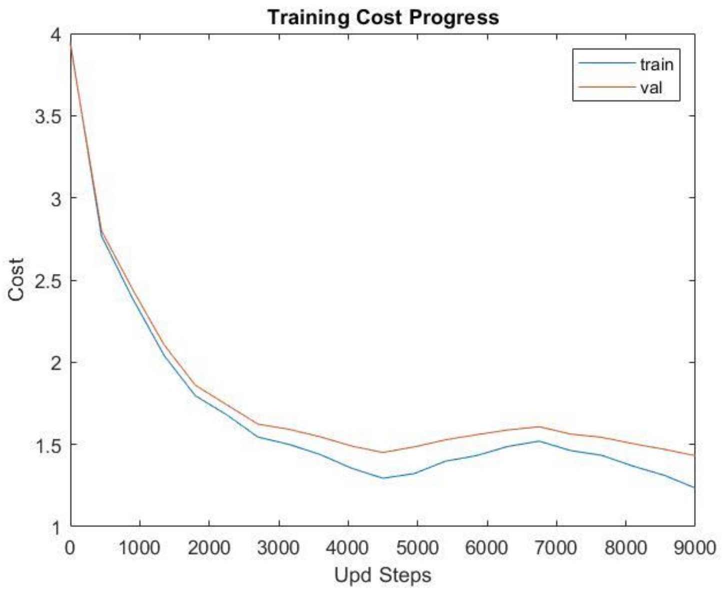

Figure 5. An illustration of the training- and validation cost for the 3-layer network with batch normalization

Nine Layer Network with and Without Batch Normalization

A 9-layer network was then trained with hidden nodes according to [50, 30, 20, 20, 10, 10, 10, 10] and the same hyperparameter settings as the above 3-layer network. This model achieved a test accuracy of 51.51 % with batch normalization and 48.95 % without. The difference is noticeable, as it should be for such a deep network. As can be observed in figure 6 and 7, the cost is generally higher without batch normalization.

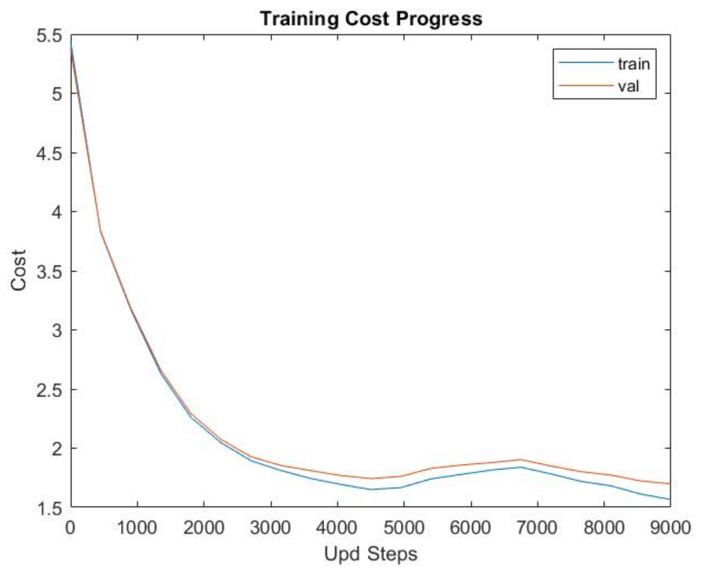

Figure 6. An illustration of the training- and validation cost for the 9-layer network without batch normalization

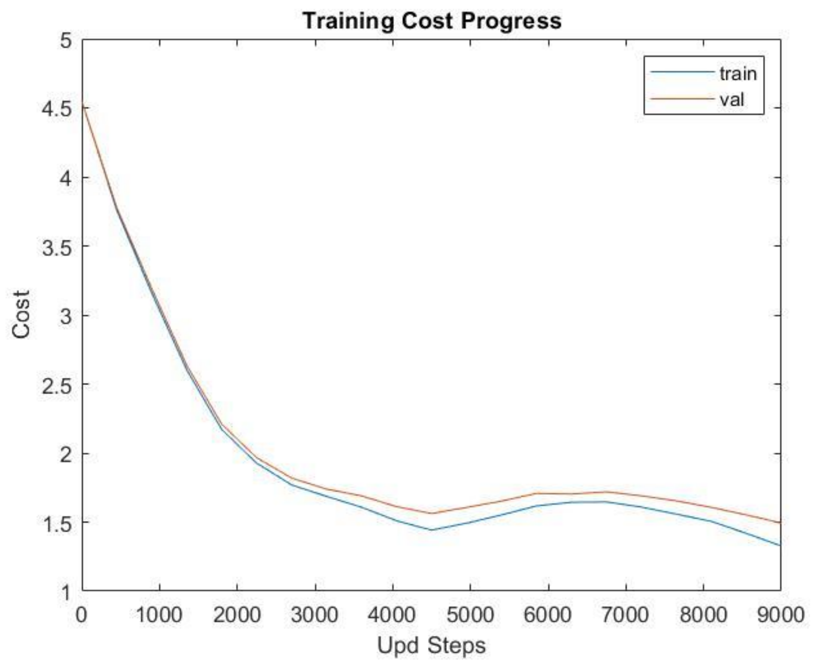

Figure 7. An illustration of the training- and validation cost for the 9-layer network with batch normalization

Random Search

In order to optimize the performance of our 3-layer network with batch normalization, a random search was then performed as in assignment 2. The validation accuracy was registered while the value of λ varied. A searched was performed in three rounds, each with decreasing range of the λ values. All 5 training batches of the *cifar-10 dataset* were used for training, except for 5000 images in a validation set. One cycle of training with n_s = 2*floor(n / batch size), where n is the number of samples in the training set, was conducted for each λ value. The search for λ was done on a log scale according to the following randomized sampling algorithm:

```
l = l_min + (l_max - l_min)*rand(1, 1);
lambda = 10^l;
```

Round 1

Ten different λ values with l_min=-5 and l_max=-1, i.e. on the range [0.00001, 0.1] yielded:

Validation Accuracy	λ
0.5054	0.020211
0.4996	4.521130E-5
0.5032	0.020952
0.5036	0.002977
0.48	0.047889
0.503	0.001886
0.4968	4.738164E-5
0.501	0.000444
0.4912	0.000354
0.4678	0.085438

Table 2. Round 1 yielded maximum validation accuracy of 50.54 %. The best values found are highlighted in green.

Round 2

Based on the green highlighted values in round 1, a search on the range [0.001, 0.1] for 8 different λ values, yielded the following result:

Validation Accuracy	λ
0.481	0.044956
0.5054	0.002126
0.484	0.045773
0.5128	0.017254
0.4756	0.069202
0.514	0.013732
0.5036	0.002177
0.508	0.006663

Table 3. Round 2 yielded maximum validation accuracy of 51.40 %. The best values found are highlighted in green.

Round 3

Based on the green highlighted values in round 2, a search on the range [0.01, 0.1] for 8 different λ values, yielded the following result:

Validation Accuracy	λ
0.4772	0.067049
0.5096	0.014582
0.4754	0.067656

0.496	0.041538
0.469	0.083188
0.4988	0.037057
0.5124	0.014754
0.5008	0.025812

Table 4. Round 3 yielded maximum validation accuracy of 51.24 %. The best values found are highlighted in green.

The top three scores from all rounds were:

Validation Accuracy	**λ**
0.514	0.013732
0.5128	0.017254
0.5124	0.014754

Table 5. Top three lambdas from all rounds

Final Training and Evaluation

The highest achieved validation accuracy of 51.40 % was obtained for λ = 0.013732 in round 2. Therefore, the 3-layer network was trained using that λ for 3 cycles using all the training data (50 0000 images) apart from 1000, which were used as a validation set. A full specification of the hyperparameters follows:

eta_min = 1e-5, eta_max= 1e-1, n_s = 5 * 45000/n_batch, batch size = 100, λ = 0.013732, 3 cycles, 50 hidden nodes in both layers.

This setting yielded a test accuracy of 54.54 %, slightly higher than when the default parameters were used earlier, although a few more training examples were used now. The same hyperparameters but with the 9-layer network, yielded a test accuracy of 51.07 %, slightly lower than earlier unfortunately.

Sensitivity to initialization

To explore the pros with batch normalization some elaboration with the weight initialization was done. For each training regime instead of using He initialization, each weight parameter W_i was initialized to be normally distributed with sigmas equal to the same value *sig* at each layer. For three runs sig was set to sig=1e-1, 1e-3 and 1e-4 respectively. The 3-layer network was trained with and without batch normalization. The network had 50 nodes at each hidden layer and basic hyper-parameter setting was eta_min = 1e-5, eta_max = 1e-1, lambda=.005, two cycles of training with n_s = 5 * 45,000 / n_batch.

The results of all the experiments are shown in table 6 and the evolution of the cost in figures 8-13. It can be observed that the test accuracy decreases as sig gets smaller when batch normalization is not used but is stable when batch normalization is used. Using a small standard deviation in the weight initialization and mean 0, means that each element of W will be close to zero. This may work okay for small networks but can lead to non-homogeneous distributions of activations across the layers of a deep network. The activations will decrease with each layer, which in turn cause the gradients computed in back propagation to be close to zero as well. This problem is mitigated by normalizing each batch with the batch normalization algorithm. That is, training becomes more stable and less sensitive to weight initialization. These experiments provide evidence for this idea.

Experiment	**Test Accuracy (%)**
sig = 1e-1 with BN	54.10
sig = 1e-1 without BN	53.17
sig = 1e-3 with BN	54.27
sig = 1e-3 without BN	50.32
sig = 1e-4 with BN	53.91
sig = 1e-4 without BN	10.00

Table 6. Results from the six different combinations of normalization and standard deviation on weight initialization

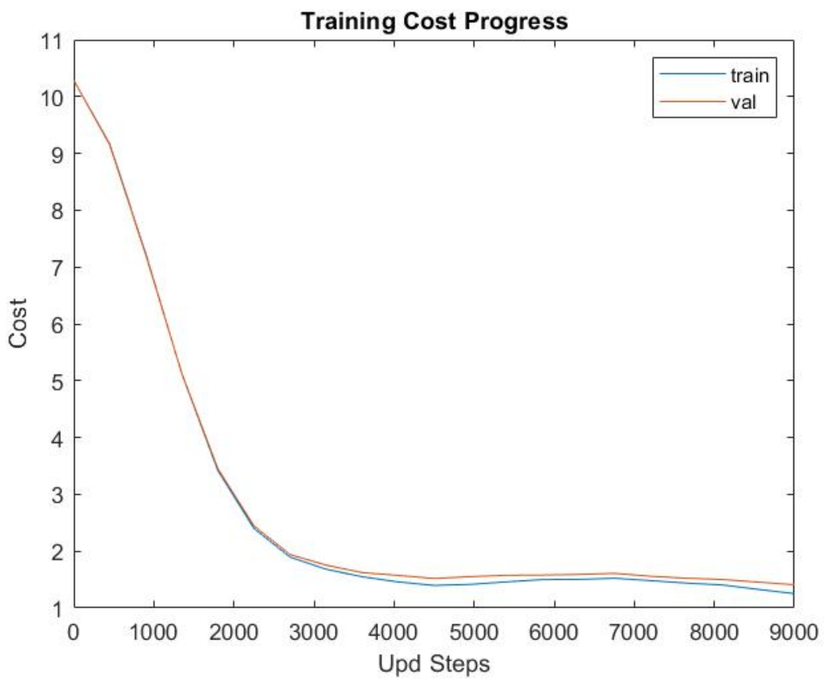

Figure 8. sig = 1e-1 with BN

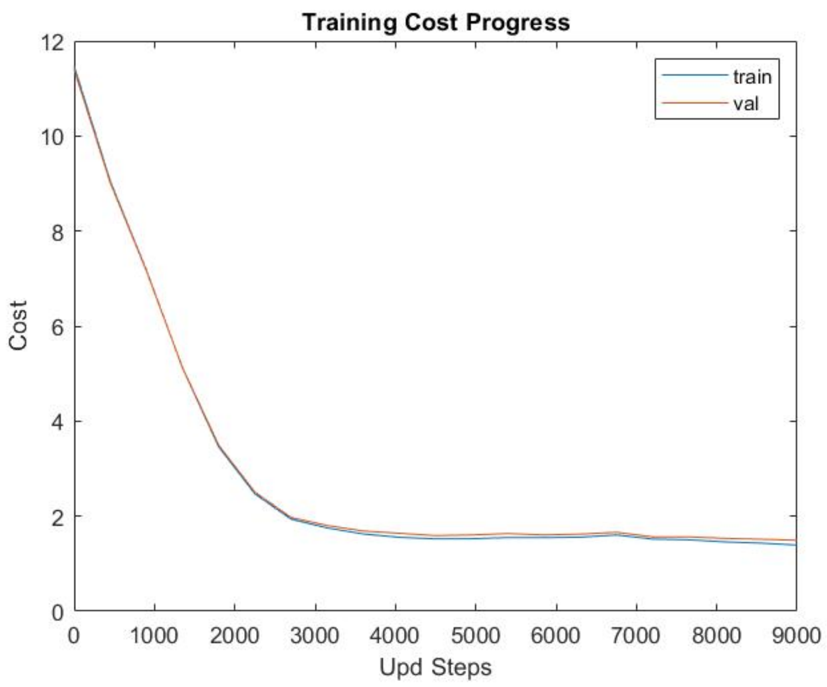

Figure 9. sig = 1e-1 without BN

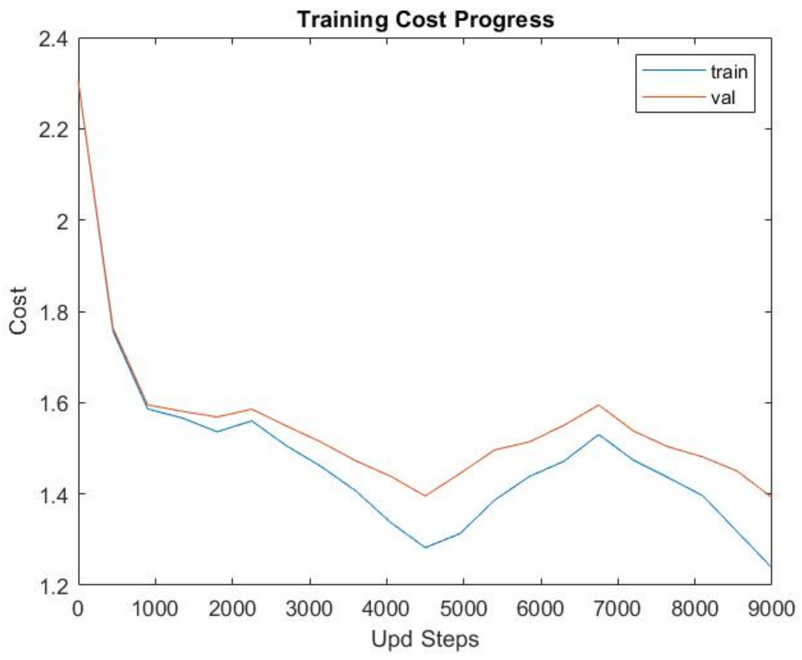

Figure 10. sig = 1e-3 with BN

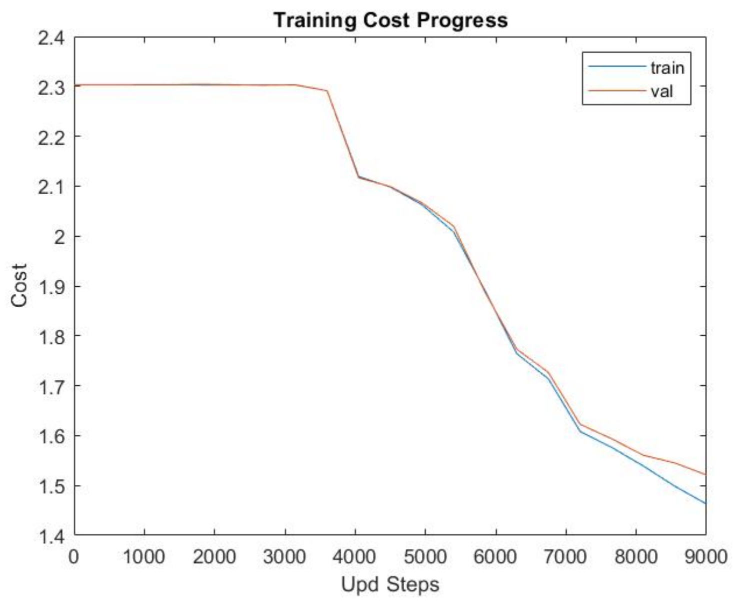

Figure 11. sig = 1e-3 without BN

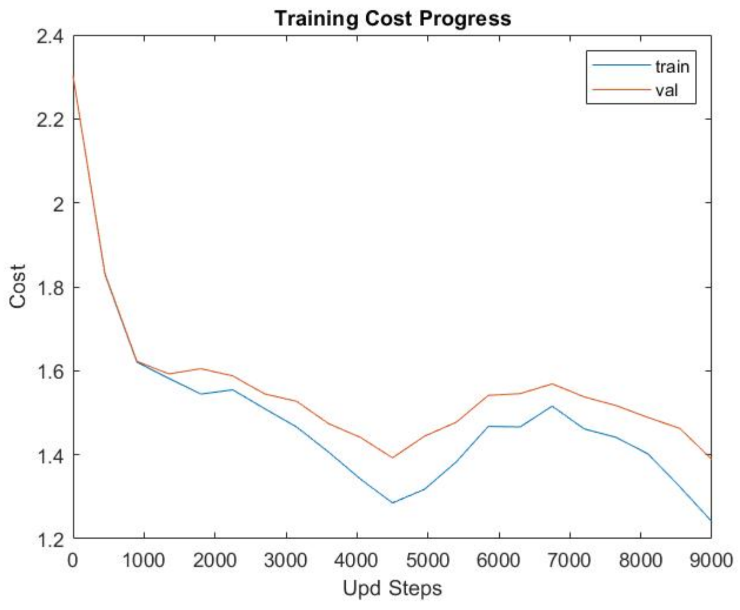

Figure 12. sig = 1e-4 with BN

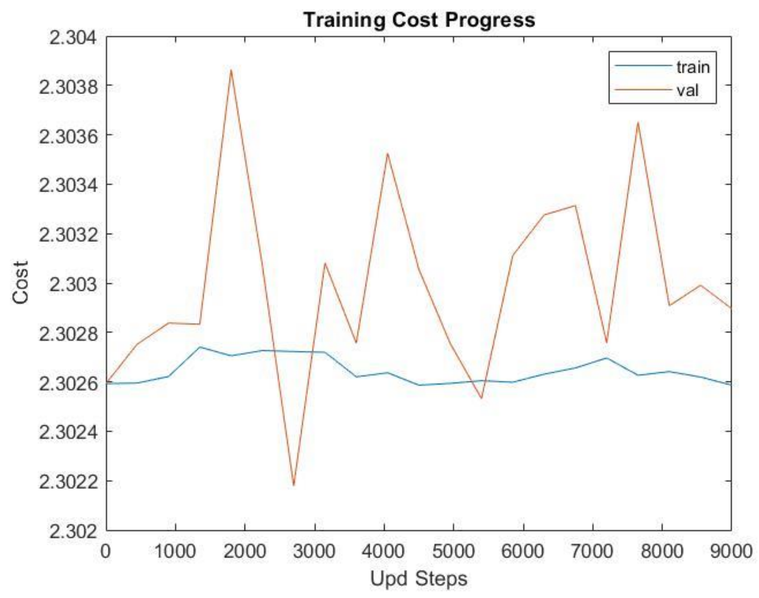

Figure 13. sig = 1e-4 without BN. It can be seen that the small gradients are causing instability in the learning process.