Deep Learning in Data Science, DD2424

Short report on assignment 2

Two layer classification network for CIFAR-10

Simon Jarvers

April 20, 2023

1 Main objectives and scope of the assignment

My goals in the assignment were:

- to implement a basic two-layer classification network,
- to use the gradient decent method to train the network,
- to test different regularization methods like data augmentation or Nesterov Momentum,
- to train the network on the well-known CIFAR-10 data set.

To study the points of interest, the experiments and plots were created in Python using the widely known libraries, i. e. NumPy and Matplotlib. This report was written without the use of ChatGPT.

2 Classification with a two-layer classification network

Many implementations could be reused from the first assignment. I adapted the numerical calculation of the gradients to verify my analytical computations. The assessment can be seen in Figure 1.

As another sanity check I let the network overfit to 100 samples. After 75 epochs the training accuracy reached 1 as illustrated in Figure 2.

2.1 Cyclic learning rates

Next, I implemented the cycling learning rate scheme with `eta_min = 1e-5` and `eta_max = 1e-1`. Similarly to the assignments plots, I let the network run

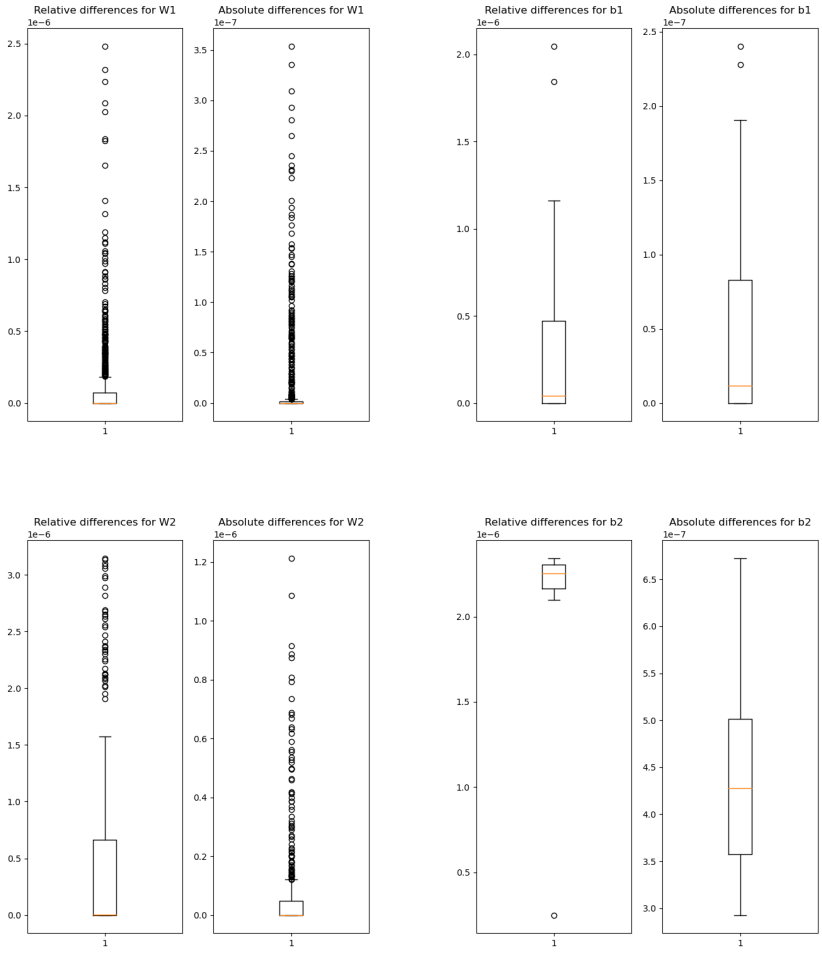

Figure 1: Box plots for W1, b1, W2 and b2

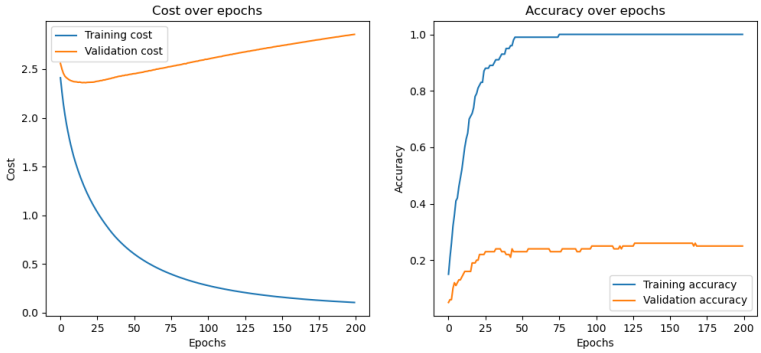

Figure 2: The network is overfitted to 100 samples and reaches accuracy of 1 after 75 epochs.

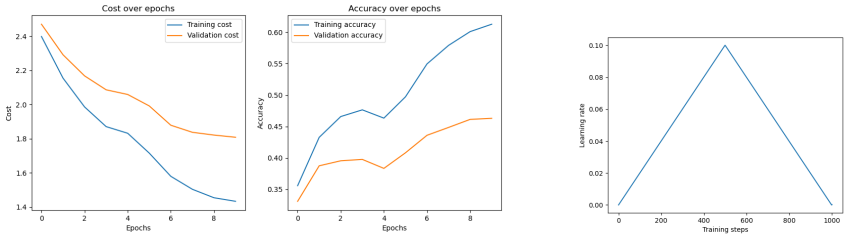

Figure 3: Performance for 1 cycle.

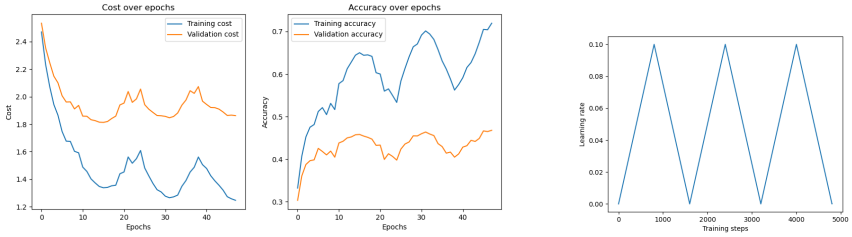

Figure 4: Performance for 3 cycles.

with 1 cycle with `n_s = 500` and 3 cycles with `n_s = 800` resulting in 10 epochs and 48 epochs respectively as the training set consists of 10000 samples. The regularization factor λ was kept at 0.01 and the test accuracies were 46.26% and 47.83%. The resulting cost and accuracy plots can be seen in Figure 3 and 4. Especially in the 3 cycles plot it can be obtained that the smaller learning rates increase the accuracy whereas higher learning rates deteriorate the performance. However, higher learning rates might help to converge faster, particularly, at the beginning of the training. This observation inspired me to test an annealing cyclic learning rate which I will further discuss in Section 3.

2.2 Weight decay optimization

First, I chose a logarithmic grid search with 9 evenly spaced values from `1e-5` to `1e-1` and two cycles in the aforementioned cyclic learning rate scheme. The validation accuracies can be seen in Table 1. In the region from `1e-5` to `1e-3` the network performed quite similarly. This made me run a random search in the aforementioned interval obtaining the best validation accuracy of 51.75% for `lambda = 4e-4` which was the value I used for the Bonus assignment tasks.

With all these optimization steps I finally achieved a test accuracy of 52.25% for `lambda = 4e-4`, `hidden_size = 50`, `n_s = 1000` for 6 cycles and 49000 training samples. Cost, accuracy and learning rate plot can be seen in Figure 5.

$\log_{10}(\lambda)$	-5	-4.5	-4	-3.5	-3	-2.5	-2	-1.5	-1
val_acc in %	50.8	50.6	51.1	50.9	50.4	50.8	49.9	46.1	37.9

Table 1: Grid search for the optimal regularization factor

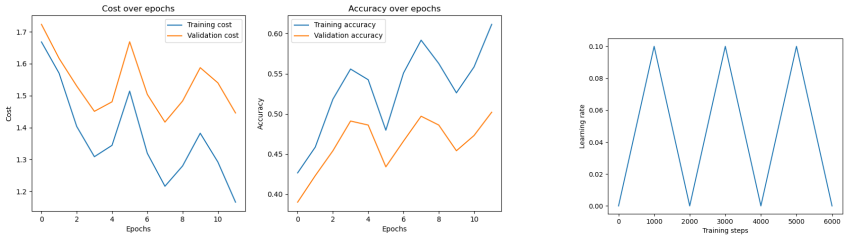

Figure 5: Best performing network with final test accuracy 52.25%

3 Bonus assignment

3.1 Optimize the performance of the network

To further improve the performance of the network, I chose (1) data augmentation, (2) to implement Nesterov Momentum, (3) to increase the number of hidden nodes and (4) to use linear annealing in combination of the previous cyclic learning rate.

Data augmentation

Depending on the new hyperparameter `augment_p` every image is augmented with a probability p. The augmentation itself can be flipping or translation up to 3 pixel in every direction or any combination of those. Typically I set $p = 5\%$

Nestrov Momentum

I followed the proposed paper and implemented the Momentum according to this scheme. I set the hyperparameter $\mu = 0.5$

```
v_prev = v # back this up
v = mu * v - learning_rate * dx # velocity update stays the same
x += -mu * v_prev + (1 + mu) * v # position update changes form
```

Number of hidden nodes

I tried 100, 200, 400 and even 800 hidden nodes. However, I could not identify any significant improvement over 400 nodes. Additionally, these large number of hidden nodes made testing of network configurations unfeasible as training took up to 20 minutes or more. The best performing network I found uses 400 nodes.

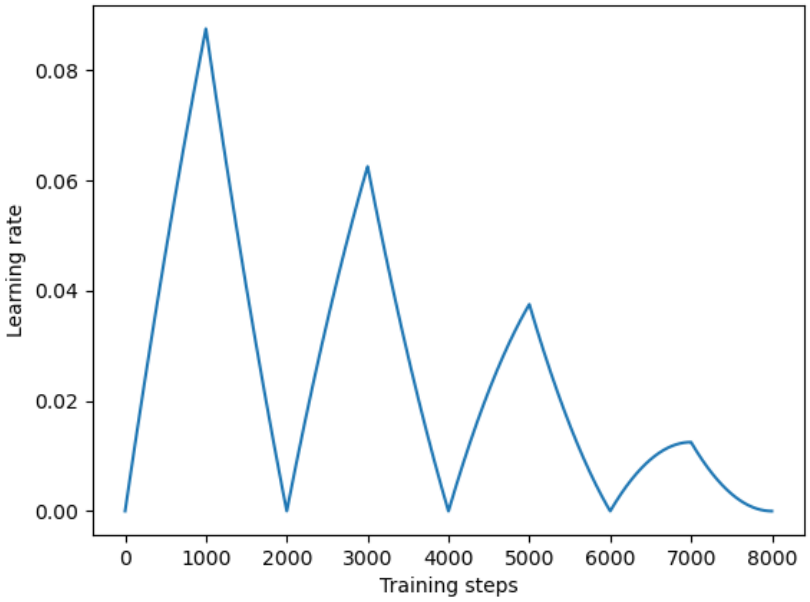

Figure 6: Cyclic learning rates in combination of linear annealing.

	100 hidden nodes		400 hidden nodes	
	6 cycles	8 cycles	6 cycles	8 cycles
$\lambda = 1e-4$	51.10%	52.90%	55.20%	56.00%
$\lambda = 1e-3$	53.70%	51.90%	56.50%	58.00%

Table 2: Grid search for regularization strength λ, number of hidden nodes and number of cycles

Learning rate annealing

To prevent the cyclic learning rate from using large values for η towards the end of the training I implemented a linear decay, resulting in a learning rate as seen in Figure 6.

3.2 Best performing network

I decided to run another grid search that can be seen in Table 2 to find good values for regularization strength λ, number of hidden nodes and number of cycles. The best validation accuracy I achieved was 58.00% with a hyperparameter configuration of $\lambda = 1e-3$, 400 hidden nodes and 8 cycles with 1000 training steps each. The networks training process is illustrated in Figure 7.

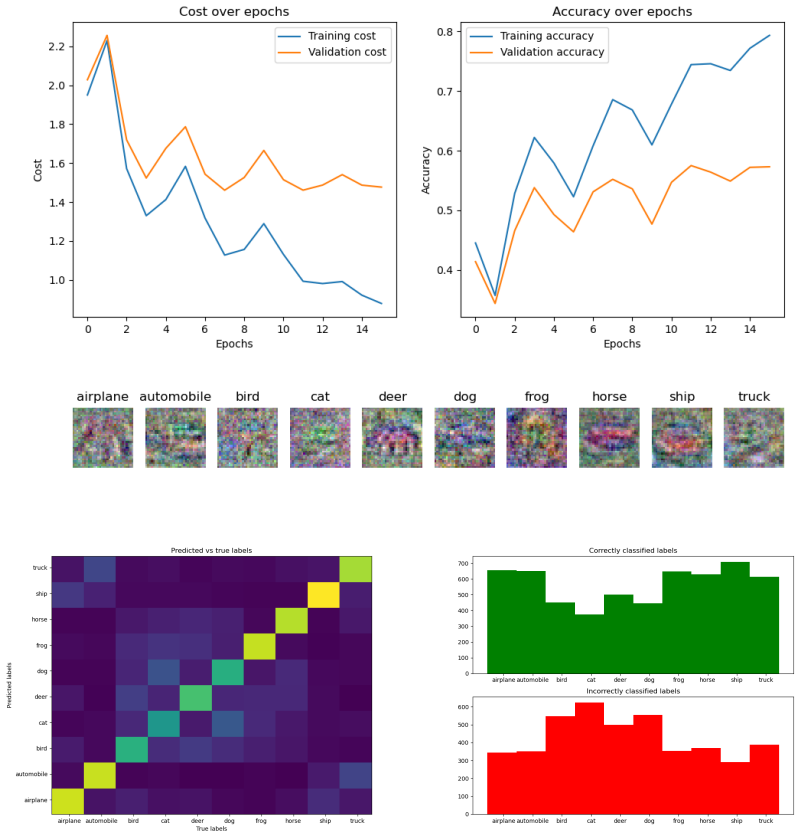

Figure 7: λ=0.001, η=[0.1, 0.00001], number of training steps = 8000, augmentation probability = 0.05, $\mu = 0.5$;
Final Validation accuracy: 58.00%
Final Test accuracy: 56.80%