DD2424 - Deep Learning in Data Science
Assignment 2 Report

Axel Berlgund

April 16, 2025

1 Gradient Computation Verification

I checked my analytical gradient computations against those calculated using PyTorch. For the verification, I compared the PyTorch-based gradient computation to the results with my own backpropagation implementation, by checking the maximum absolute difference in weights (W) and biases (b) for each layer.
The verification was performed on a small subset of the training data with reduced dimensionality, for simplicity, since the mathematical correctness of gradient computation is not dependent on the size of the network. This also to avoid numerical issues. I used a network with the following configuration:

- Input dimension (d) = 5
- Hidden layer size (m) = 6
- Output classes (K) = 10
- Batch size = 3
- λ (regularization) = 0

The maximum absolute differences between my gradients and the PyTorch-computed gradients were:

Parameter	Max Absolute Difference
Layer 1 - Weights (W_1)	2.78×10^{-17}
Layer 1 - Biases (b_1)	1.56×10^{-17}
Layer 2 - Weights (W_2)	2.22×10^{-16}
Layer 2 - Biases (b_2)	2.78×10^{-17}

Table 1: Maximum absolute differences between analytical and PyTorch gradients.

These differences are extremely small (on the order of 10^{-16} to 10^{-17}), which are within floating-point precision error bounds. This indicates that my gradient computations are essentially identical to the PyTorch implementation and therefore can be considered bug-free.
As a sanity check, I also verified that the network can overfit a small dataset (100 examples) with $\lambda = 0$, as, can be seen in Fig 1, which further confirms that the gradient computations are correct.

In the overfitted example I used parameters

- Input dimension $(d) = 3072$
- Hidden layer size $(m) = 50$
- Output classes $(K) = 10$
- Batch size $= 1$
- N Samples $= 100$
- $\eta = 10^{-5}$
- $\lambda = 0$

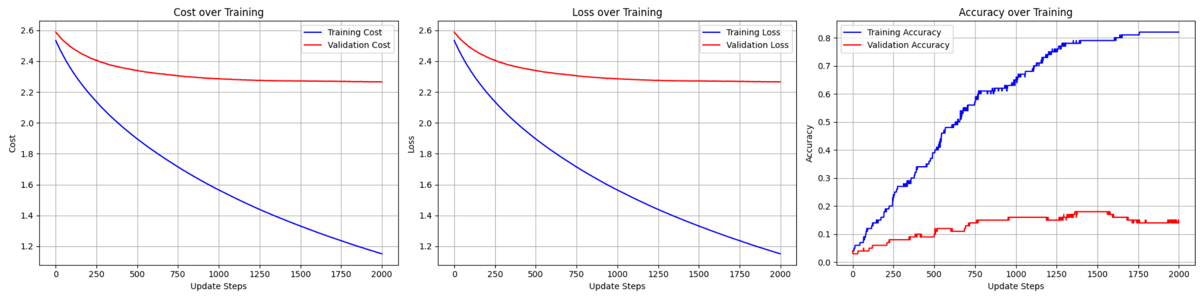

Figure 1: Training and validation curves after 200 epochs

We observe an increasing gap between the training and validation curves across all plots. While the validation loss and cost have plateaued, the accuracy plot shows a slight decline in validation performance. This suggests early signs of overfitting, where the model continues to improve on the training data but no longer generalizes better to the validation set.

2 Cyclical Learning Rates Training

2.1 One Cycle Training

I implemented cyclical learning rates as specified in the assignment with the following parameters:

- Minimum learning rate $(\eta_{min}) = 10^{-5}$
- Maximum learning rate $(\eta_{max}) = 10^{-1}$
- Step size $(n_s) = 500$
- Batch size $= 100$
- Regularization parameter $(\lambda) = 0.01$

Fig 2 shows the training and validation cost, loss, and accuracy over one cycle of training.

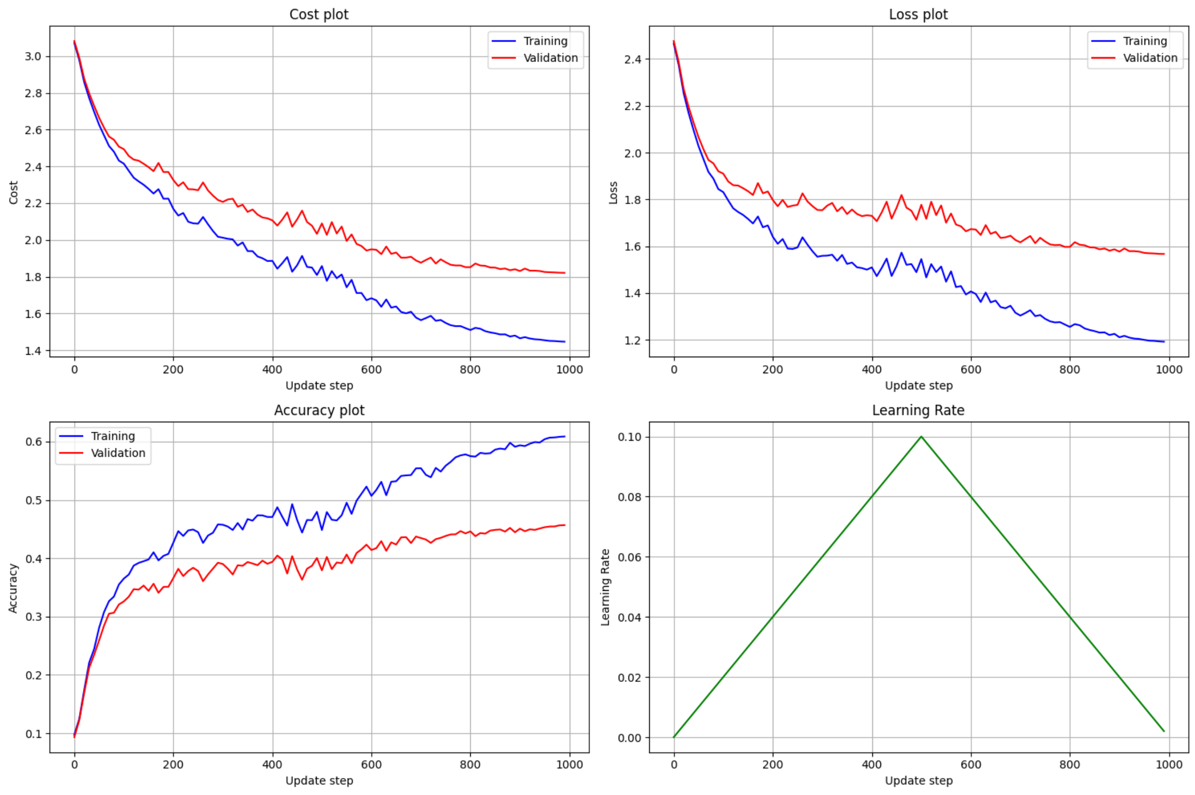

Figure 2: Training curves for one cycle of training with cyclical learning rates.

2.2 Three Cycles Training

I extended the training to three cycles with a larger step size:

- Step size $(n_s) = 800$
- Other parameters remained the same

Fig 3 shows the training and validation cost, loss, and accuracy over three cycles of training.

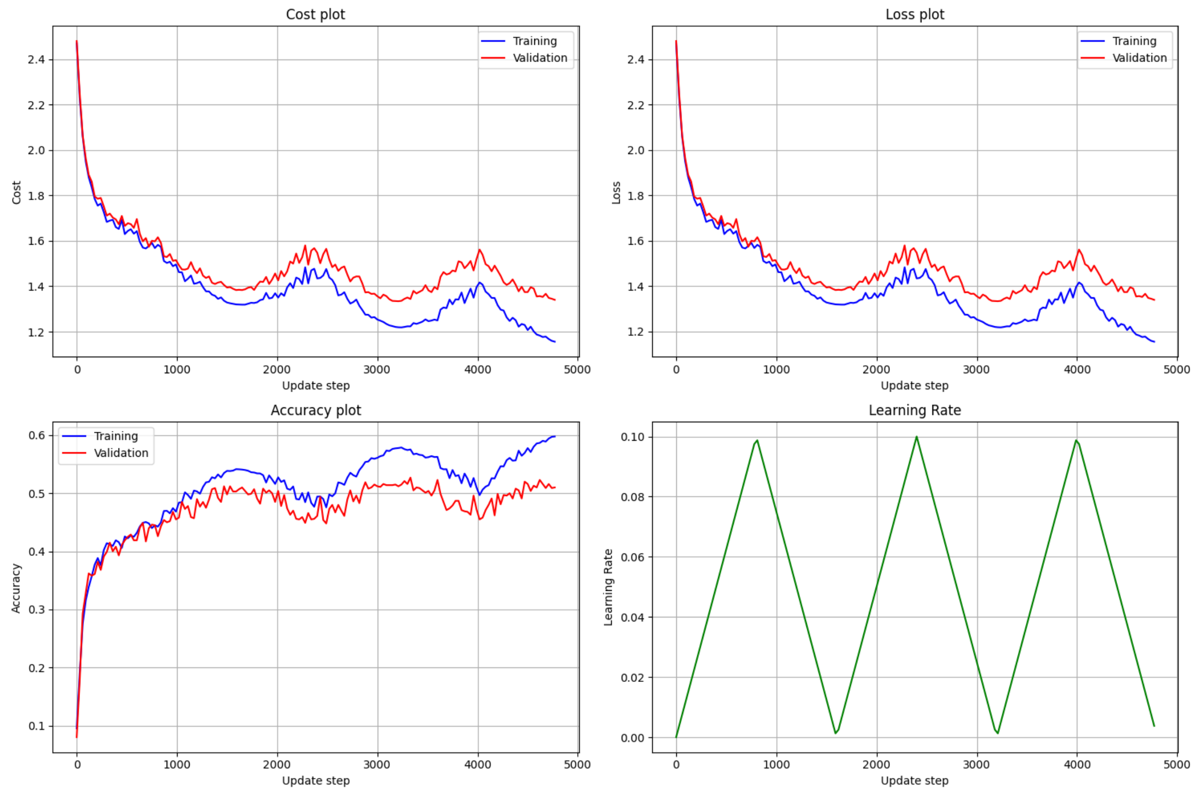

Figure 3: Training curves for three cycles of training with cyclical learning rates.

2.3 Comments on the Training Curves

From the training curves, we observe the following patterns:

- **Learning Rate Effect**: The cyclical pattern of the learning rate clearly impacts the loss and accuracy. When the learning rate is at its highest, the loss tends to increase temporarily, and the accuracy drops. When the learning rate decreases, the network converges again, achieving better performance.
- **Convergence Pattern**: With each cycle, the overall validation accuracy improved, indicating that the cyclical learning rate helps the network escape local minima and find better parameters.
- **Overfitting**: The training accuracy consistently exceeds the validation accuracy, suggesting some degree of overfitting despite the regularization. However, the gap remains relatively stable over time and this pattern is normal since the training performance will almost always be a little better than new samples.
- **Stability**: The training is stable throughout the cycles, with no divergence observed even at the highest learning rates, demonstrating the robustness of the cyclical learning rate approach. We could smooth the graph, as shown in the assignment template, by plotting scores less frequently instead of at every update step. However, in this example, the current level of detail was considered sufficient.

These observations align with the expectations set so far in the assignment.

3 Lambda Search and Final Training

3.1 Coarse Lambda Search

For the coarse search of the regularization parameter λ, I used the following setup:

- Search range: $\lambda \in [10^{-5}, 10^{-1}]$ (log scale)
- Number of values tested: 8 (uniform grid in log space)
- Number of cycles for each training: 2
- Step size: $n_s = 2 \times \lfloor \frac{n}{n_{batch}} \rfloor$
- Training data: All 5 batches except for 5000 validation samples

The three best performing networks from the coarse search had the following parameters:

Rank	λ	Validation Accuracy	Training Accuracy
1	1.39×10^{-4}	0.5252	0.5912
2	1.00×10^{-5}	0.5224	0.5887
3	3.70×10^{-5}	0.5212	0.5898

Table 2: Top 3 performing networks from the coarse lambda search.

Figure 4 shows the results of the coarse lambda search.

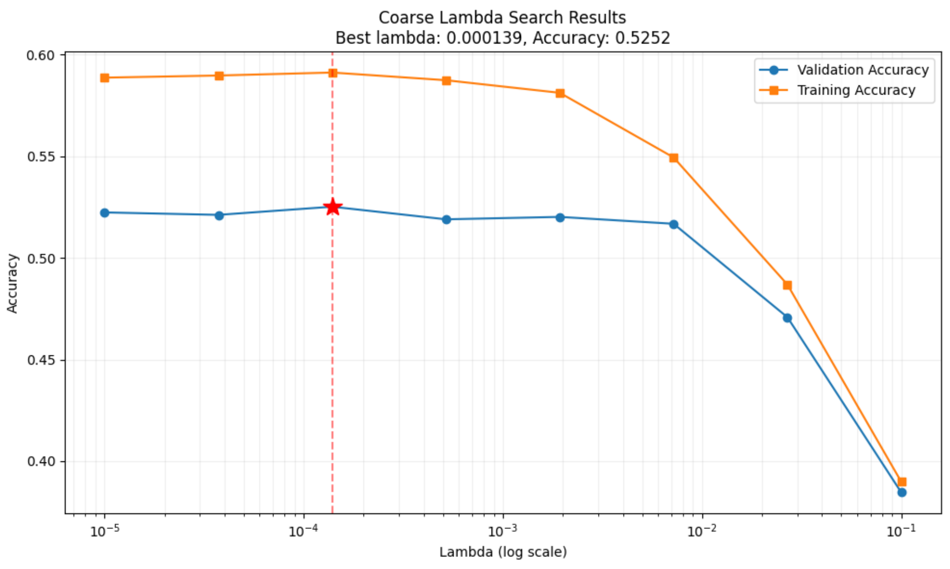

Figure 4: Results of the coarse lambda search showing validation and training accuracy.

3.2 Fine Lambda Search

Based on the coarse search results, I conducted a fine search with the following setup:

- Search range: $\lambda \in [1.05 \times 10^{-4}, 9.48 \times 10^{-3}]$
- Number of values tested: 8 (random sampling in the range)
- Number of cycles for each training: 2

- Same step size calculation as the coarse search
- Same training/validation split as the coarse search

The three best performing networks from the fine search had the following parameters:

Rank	λ	Validation Accuracy	Training Accuracy
1	3.91×10^{-6}	0.5262	0.5897
2	1.25×10^{-4}	0.5246	0.5877
3	7.00×10^{-6}	0.5238	0.5887

Table 3: Top 3 performing networks from the fine lambda search.

Figure 5 shows the results of the fine lambda search.

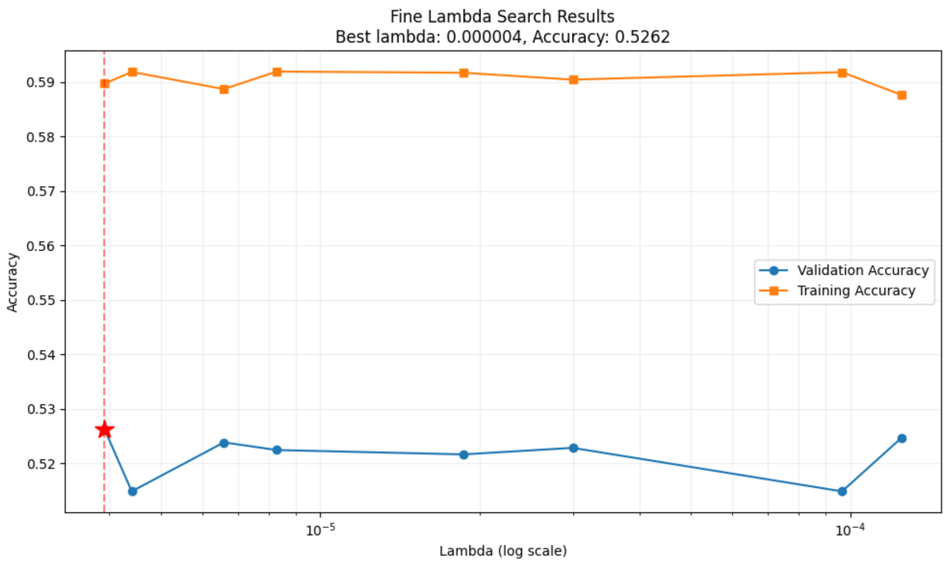

Figure 5: Results of the fine lambda search showing validation and training accuracy.

3.3 Final Model Training and Evaluation

Using the best lambda value from the fine search ($\lambda = 3.91 \times 10^{-6}$), I trained the final model with the following setup:

- Training data: All 5 batches except for 1000 validation samples (49000)
- Number of cycles: 3
- Step size: $n_s = 800$
- Other parameters: $\eta_{min} = 10^{-5}$, $\eta_{max} = 10^{-1}$, batch size $= 100$

Figure 6 shows the training and validation loss/cost during the final training.

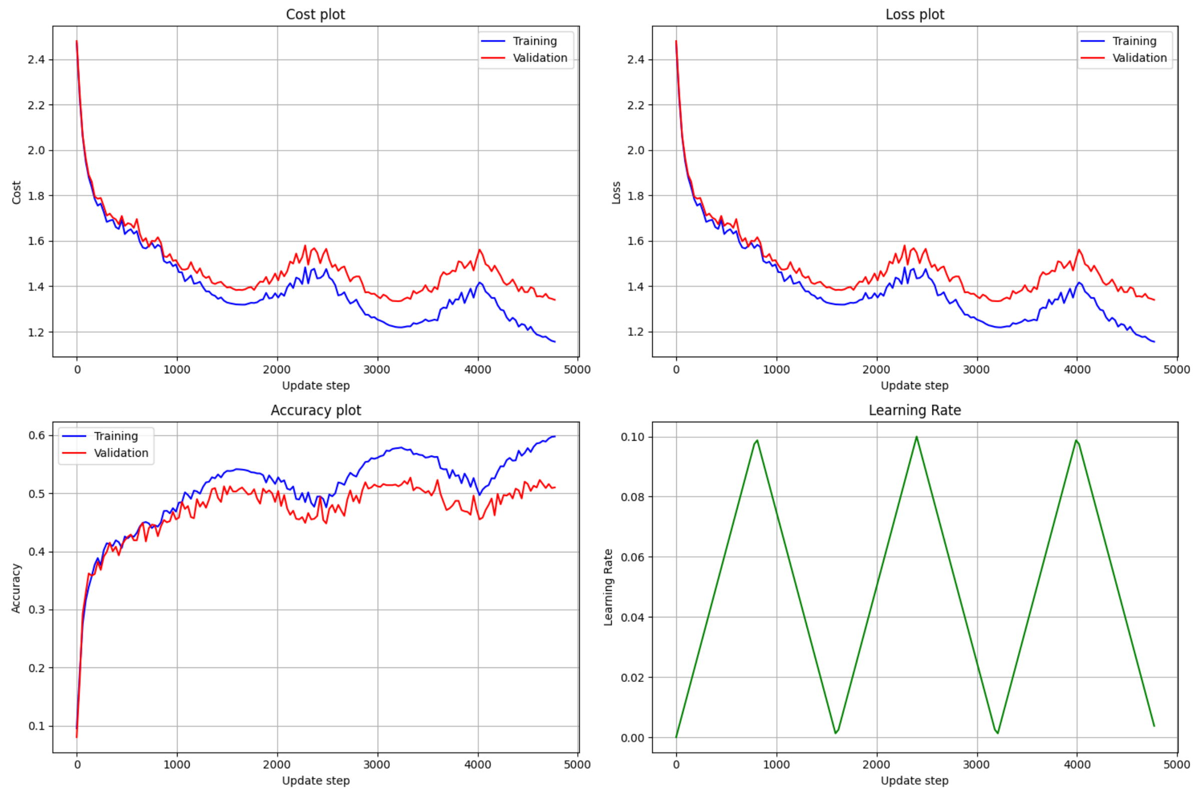

Figure 6: Training and validation curves for the final model.

The final model achieved the following performance metrics:

- Test accuracy: 0.5114
- Final validation accuracy: 0.5100
- Final training accuracy: 0.5978

This test accuracy exceeds the 50% threshold mentioned in the assignment, confirming the effectiveness of the implementation and the parameter search strategy.

4 Conclusion

In this assignment, I implemented a two-layer neural network with ReLU activation and trained it using mini-batch gradient descent with cyclical learning rates. The implementation was verified by comparing gradients with PyTorch, and the network was optimized by searching for the best regularization parameter.
The final model achieved a test accuracy of 51.14% on the CIFAR-10 dataset, demonstrating that even a simple two-layer neural network can achieve reasonable performance with proper tuning. The cyclical learning rate approach proved effective in enabling the network to escape local minima and find better parameters.