DD2424 - Assignment 1 Report

Axel Berglund
KTH Royal Institute of Technology

April 8, 2025

1 Introduction

This assignment focuses on implementing and training a simple image classifier using the CIFAR-10 dataset. The model is a one-layer neural network trained using mini-batch gradient descent with softmax activation and cross-entropy loss. The goal was to understand and implement core components of supervised learning pipelines from scratch, including data preprocessing, forward and backward propagation, gradient checking, and performance evaluation.

2 Implementation

As part of the model implementation, I manually derived and implemented the gradients for the one-layer neural network with respect to the weights and biases. To verify the correctness of the analytical gradient computations, I compared them to gradients obtained using PyTorch's automatic differentiation engine.

The gradients computed by both methods were then compared using an element-wise absolute difference check. The CompareGradients function confirmed that the gradients from my implementation closely matched those from PyTorch, with all differences falling well within a small tolerance (1e-6). This served as a strong indication that the analytical gradients were implemented correctly.

3 Results

3.1 Loss and Cost Curves

The training and validation loss and cost were recorded after each epoch during training for four different configurations of hyperparameters. When $\lambda = 0$, the loss and cost are equivalent since no regularization is applied.

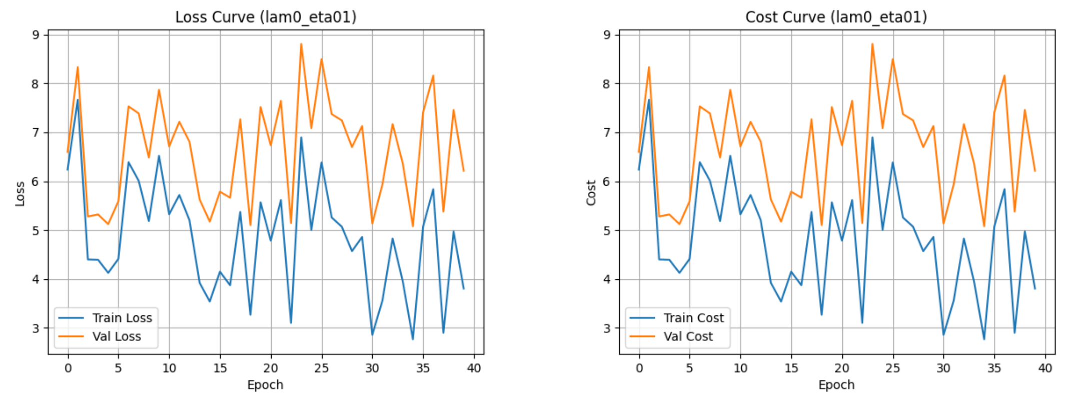

Figure 1: Loss (left) and cost (right) curves for $\lambda = 0$, $\eta = 0.1$.

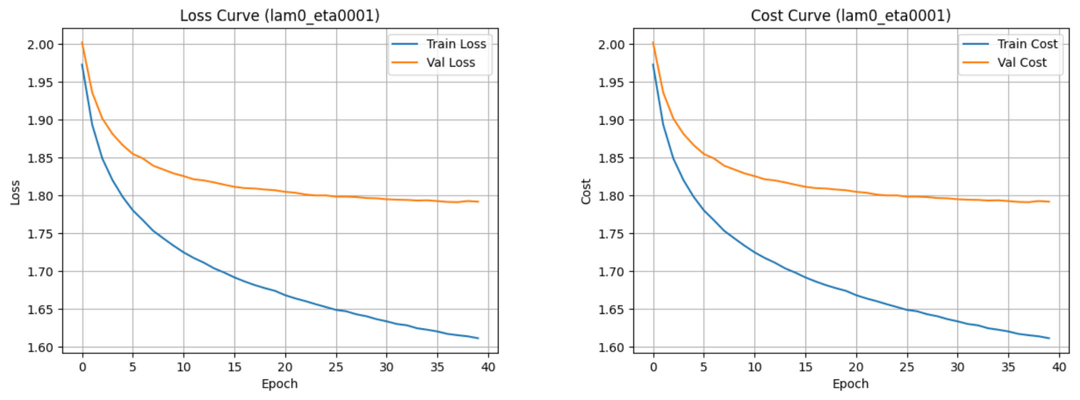

Figure 2: Loss (left) and cost (right) curves for $\lambda = 0$, $\eta = 0.001$.

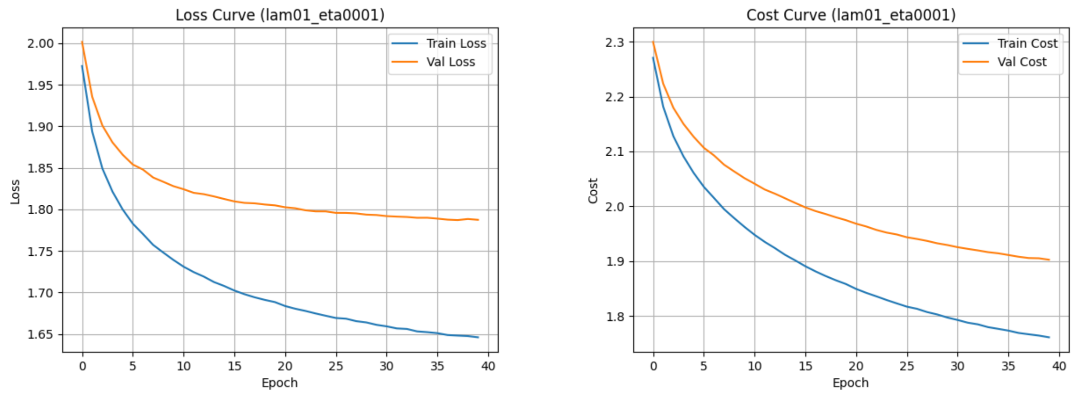

Figure 3: Loss (left) and cost (right) curves for $\lambda = 0.1$, $\eta = 0.001$.

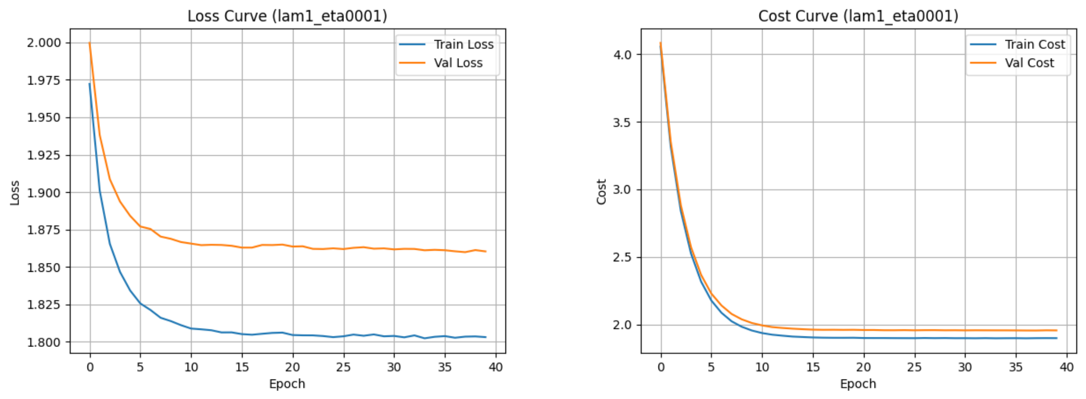

Figure 4: Loss (left) and cost (right) curves for $\lambda = 1$, $\eta = 0.001$.

3.2 Visualization of Learned Weights

The weight matrix of the trained network was reshaped and visualized as images to illustrate what patterns the model has learned for each class in CIFAR-10. Below are the visualizations for each training configuration.

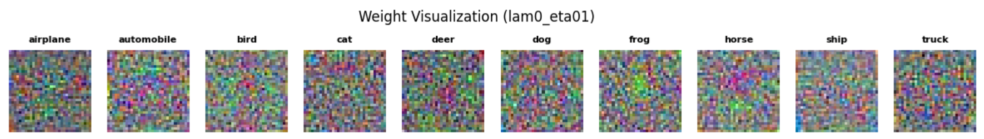

Figure 5: Learned weights for $\lambda = 0$, $\eta = 0.1$.

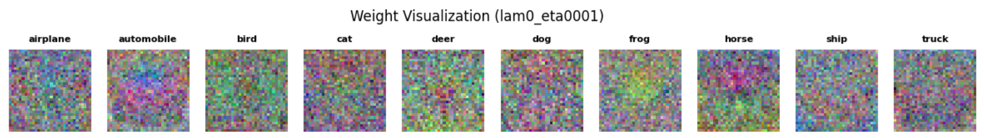

Figure 6: Learned weights for $\lambda = 0$, $\eta = 0.001$.

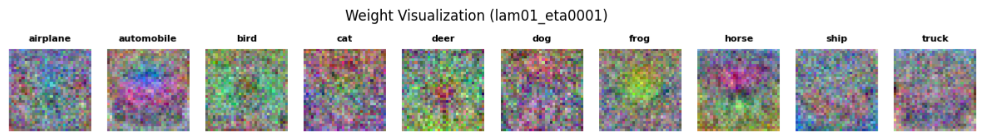

Figure 7: Learned weights for $\lambda = 0.1$, $\eta = 0.001$.

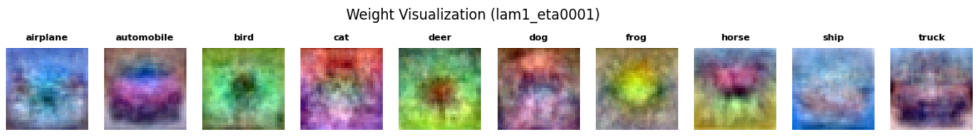

Figure 8: Learned weights for $\lambda = 1$, $\eta = 0.001$.

4 Conclusion

The network was evaluated across four different hyperparameter settings. The final test accuracies were as follows:

- $\lambda = 0$, $\eta = 0.1$: **28.77%**
- $\lambda = 0$, $\eta = 0.001$: **39.32%**
- $\lambda = 0.1$, $\eta = 0.001$: **39.40%**
- $\lambda = 1$, $\eta = 0.001$: **37.38%**

These results highlight the importance of choosing appropriate hyperparameters for training. A high learning rate ($\eta = 0.1$) led to significantly worse performance, due to instability during optimization as can be seen in Figure 1. Lowering the learning rate to $\eta = 0.001$ resulted in a substantial improvement in accuracy.

Introducing a small amount of regularization ($\lambda = 0.1$) slightly improved generalization compared to no regularization, which was expected. However, increasing the regularization strength further ($\lambda = 1$) reduced performance, suggesting that excessive regularization constrained the model too much and limited its ability to fit the data.

Overall, careful tuning of the learning rate and regularization parameter is crucial for achieving good performance in training even simple neural networks.