DD2424 - Assignment 1

Silpa Soni Nallacheruvu

April 22, 2025

The goal of this project was to implement a simple image classifier for the CIFAR-10 dataset using softmax regression.

The project includes custom implementations for data loading, preprocessing, and one-hot encoding from the CIFAR-10 dataset. Core functions such as `ApplyNetwork`, `ComputeLoss` and `BackwardPass` handle forward propagation, cross-entropy loss calculation, and gradient computation respectively. Gradient correctness is verified against PyTorch using relative error checks. The `MiniBatchGD` function performs model training with mini-batch gradient descent while tracking loss and cost across epochs. Visualizations of the learned weights and performance metrics help interpret the learning progress of the classifier.

I was able to successfully implement the analytical computation of gradients for the cross-entropy loss with L2 regularization in the `BackwardPass` function. To verify the implementation, I compared my gradients with numerically computed gradients from PyTorch on a small subset of the data (d = 10, n = 3).

The comparison was made using a relative error metric:

$$\text{Relative Error} = \frac{|g_a - g_n|}{\max(\epsilon, |g_a| + |g_n|)}$$

where g_a is the analytical gradient, g_n is the numerical gradient and ϵ is a small constant to avoid division by zero.

I tested the gradients of both the weights and biases, and the max relative error in W was $\approx$ 1.09e-15 and in b it was $\approx$ 9.78e-17. I was able to conclude that the analytical gradients are correct, as the relative errors are very small and within the acceptable range for numerical precision.

Visualization

W is the weight matrix of the softmax classifier, where each column corresponds to a class, and each row corresponds to a pixel in the image. The images are reshaped from the weight matrix and displayed to visualize the features learned by the model.

The images below show the visualization of the learned weight matrix W after training the network for 40 epochs using mini-batch gradient descent. Each square represents the weight filter for one of the 10 CIFAR-10 classes, reshaped into a 32×32×3 RGB image. These templates highlight the patterns the model associates with each class based on the training data, using the following respective parameter settings.

For each setting, the loss plot shows the training and validation loss over 40 epochs of mini-batch

gradient descent, whereas the cost plot shows the training and validation cost, including both loss and L2 regularization.

1. High Learning Rate, No Regularization

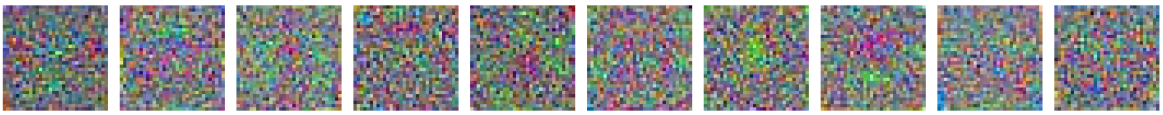


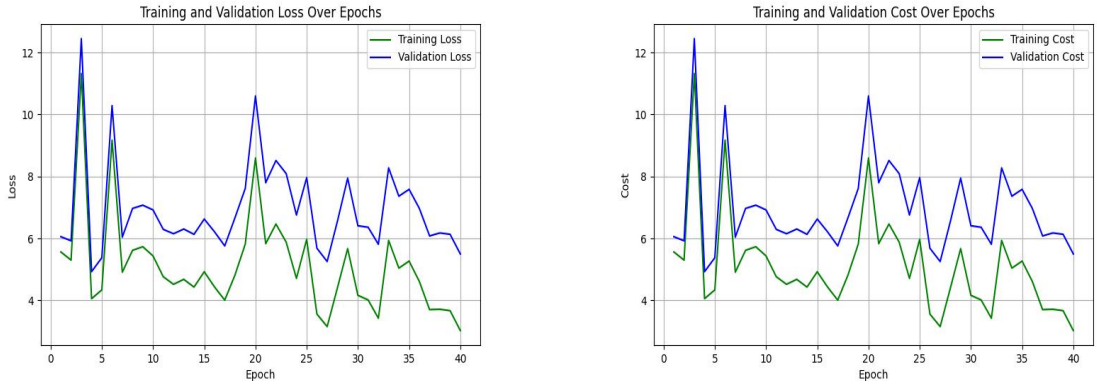
Figure 1: The learnt W matrix visualized with the parameter settings: n_batch=100, eta=.1, n_epochs=40 and lam=0.

Here are the training, validation, and testing accuracies:

```
Accuracy for training : 46.93%
Accuracy for validation : 30.44%
Accuracy for test : 30.46%
```

Listing 1: The final training, validation and test accuracies for Case 1

The filters appear highly noisy and lack structure, likely due to the high learning rate causing unstable updates. Despite this, the model learns to separate classes to a reasonable extent, though without clear feature representations.



(a) Training and Validation Loss

(b) Training and Validation Cost

Figure 2: Loss and cost plots for $\eta = 0.1$, $n\_batch = 100$, $n\_epochs = 40$, $\lambda = 0$

The plots show large fluctuations in both training and validation loss and cost across epochs. This instability suggests that the learning rate is too high, causing the optimization process to diverge rather than converge smoothly.

2: Lower Learning Rate, No Regularization

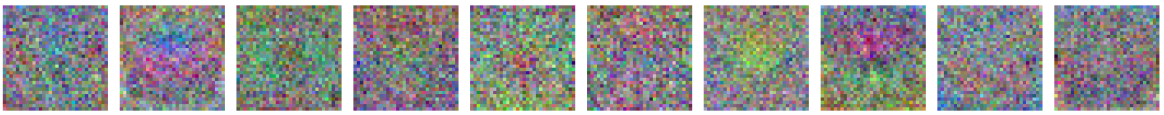
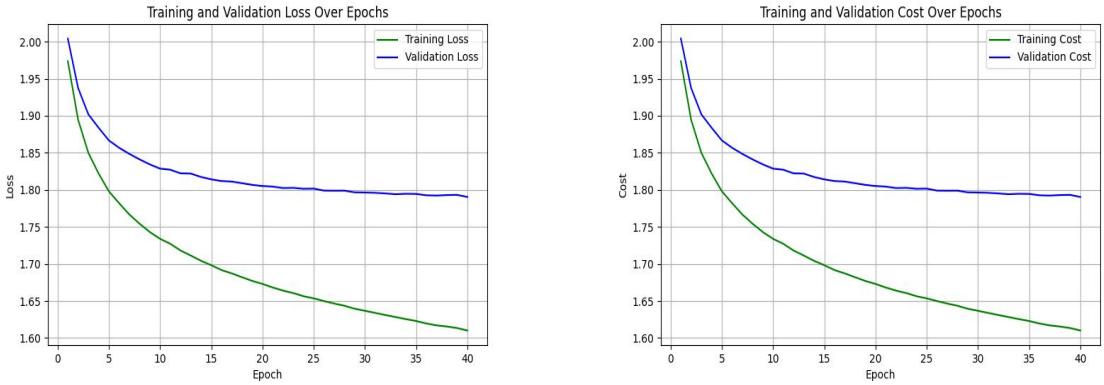


Figure 3: The learnt W matrix visualized with the settings: n_batch=100, eta=.001, n_epochs=40 and lam=0.

```
Accuracy for training : 45.36%
Accuracy for validation : 38.55%
Accuracy for test : 39.34%
```

Listing 2: The final training, validation and test accuracies for Case 2

With a smaller learning rate, the filters begin to form smoother and more defined patterns, suggesting more stable and effective convergence without regularization.



(a) Training and Validation Loss

(b) Training and Validation Cost

Figure 4: Loss and cost plots for $\eta = 0.001$, $n\_batch = 100$, $n\_epochs = 40$, $\lambda = 0$

The plots show smooth and steadily decreasing training and validation loss and cost over 40 epochs, indicating stable convergence. With a small learning rate and no regularization, the model learns effectively while maintaining a consistent gap between training and validation curves, suggesting slight overfitting but good generalization.

3: Lower Learning Rate with Moderate Regularization

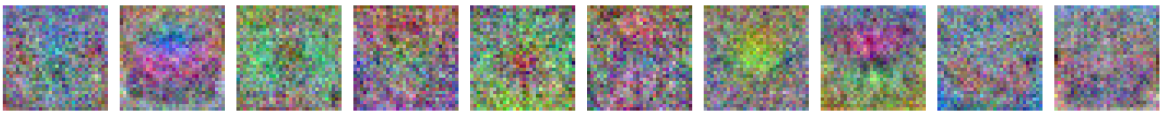
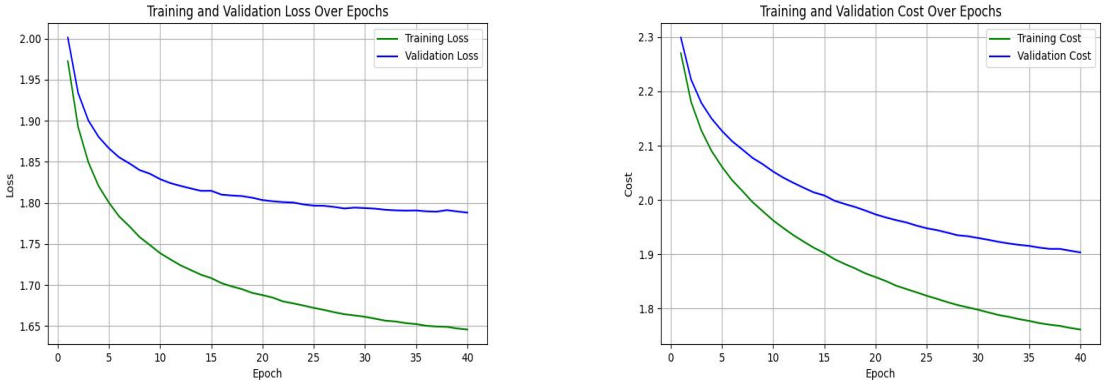


Figure 5: The learnt W matrix visualized with the parameter settings: n_batch=100, eta=.001, n_epochs=40 and lam=0.1

```
Accuracy for training : 44.78%
Accuracy for validation : 39.03%
Accuracy for test : 38.99%
```

Listing 3: The final training, validation and test accuracies for Case 3

The regularization helps suppress noisy weights, leading to slightly cleaner filters compared to no regularization, which helps generalization without significantly compromising accuracy.



(a) Training and Validation Loss

(b) Training and Validation Cost

Figure 6: Loss and cost plots for $\eta = 0.001$, $n\_batch = 100$, $n\_epochs = 40$, $\lambda = 0.1$

Compared to the $\lambda = 0$, the curves for $\lambda = 0.1$ show slightly more regularized behavior, with reduced overfitting as the validation curves stay closer to training ones. However, the final accuracy is marginally lower, reflecting a trade-off between model flexibility and generalization.

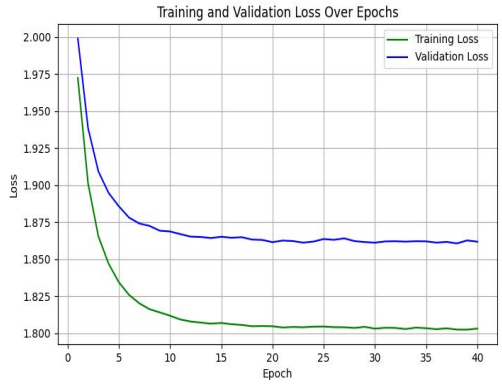
4: Lower Learning Rate with Strong Regularization

Figure 7: The learnt W matrix visualized with the parameter settings: n_batch=100, eta=.001, n_epochs=40 and lam=1

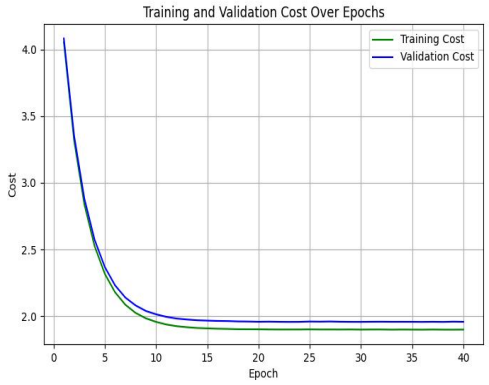
```
Accuracy for training : 39.99%
Accuracy for validation : 36.46%
Accuracy for test : 37.39%
```

Listing 4: The final training, validation and test accuracies for Case 4

The filters become significantly smoother and more structured, suggesting that strong regularization enforces simplicity in the learned weights, though it slightly reduces performance by limiting model capacity.



(a) Training and Validation Loss



(b) Training and Validation Cost

Figure 8: Loss and cost plots for $\eta = 0.001$, $n\_batch = 100$, $n\_epochs = 40$, $\lambda = 1$

The plots show very stable and flat convergence of both training and validation curves, indicating strong regularization. The overall cost remains higher, however the validation cost is marginally close to the training cost, suggesting underfitting due to excessive regularization.

Observations

Across the four configurations, we observe the impact of varying both the learning rate η and regularization strength λ on the learned weight filters, loss/cost trends, and overall model performance.

With a high learning rate and no regularization (1), both loss and cost plots show large fluctuations across epochs, indicating unstable training and divergence. This is reflected in noisy, unstructured filters and limited improvement in accuracy. Reducing the learning rate to $\eta = 0.001$ while keeping $\lambda = 0$ results in a smooth, steadily decreasing trend in the loss and cost plots. This setup yields the best generalization performance, with more stable filters and the highest test accuracy of 39.34%.

Introducing moderate regularization ($\lambda = 0.1$) leads to slightly higher loss and cost values compared to $\lambda = 0$, but the curves remain smooth and stable. This regularization helps suppress noise in the learned weights, resulting in cleaner filters without a notable drop in accuracy. On the other hand, strong regularization ($\lambda = 1$) causes the cost to plateau early and limits the model's ability to learn further. The loss curve also flattens quickly, and although the filters are visibly cleaner, performance drops slightly, indicating underfitting.

From these results, we see that increasing the regularization parameter λ helps reduce overfitting and improves generalization, as it penalizes overly complex models. However, applying too much regularization can limit the model's capacity to learn, leading to underfitting. In this case, the best performance was observed with $\lambda = 0$, but $\lambda = 0.1$ still achieved nearly equivalent results with cleaner filters. This highlights the importance of tuning regularization strength to strike the right balance between model complexity and generalization performance.

The learning rate η also plays a crucial role in the training process. It determines the size of parameter updates and thus affects both the speed and stability of learning. A learning rate that is too large $\eta = 0.1$ may cause the model to diverge or oscillate, while a rate that is too small can lead to excessively slow convergence or getting stuck in suboptimal regions. In this experiment, $\eta = 0.001$ proved to be effective in allowing the model to learn steadily across epochs. These results reinforce the importance of carefully selecting both the learning rate and regularization parameter to achieve optimal training outcomes.