Assignment 1 Report

DD2424, Kristian Fatohi

1. Introduction

The objective of Lab Assignment 1 is to develop and evaluate a single-layer neural network designed for the classification of the diverse categories present in the CIFAR-10 dataset. This network will utilize a cross-entropy loss function combined with L_2 regularization, and the optimization of its parameters will be achieved through the implementation of mini-batch gradient descent.

2. Results

I chose to use CIFAR-10 *data batch 1* for training and *data batch 2* for validation. The testing was done by using CIFAR-10 *test batch*. There are 10 possible targets (labels) and each batch contains 10000 images. The training was done using different values of hyperparameters (Total epoch, Batch size, Lambda, and Eta). The following plots and tables all represent different models with different values of hyperparameters.

2.1 Model 1

First I tested the accuracy for this model for each step (Training, Validation, and Testing) and got the following:
Training: 40.83%**, Validation:** 29.12%, **Testing:** 27.21%

Then using these values for the hyperparameters, I got these plots for model 1.
<u>Hyperparameters</u>:
Total Epoch: 40, **Batch Size:** 100, **λ:** 0, **η:** 0.1

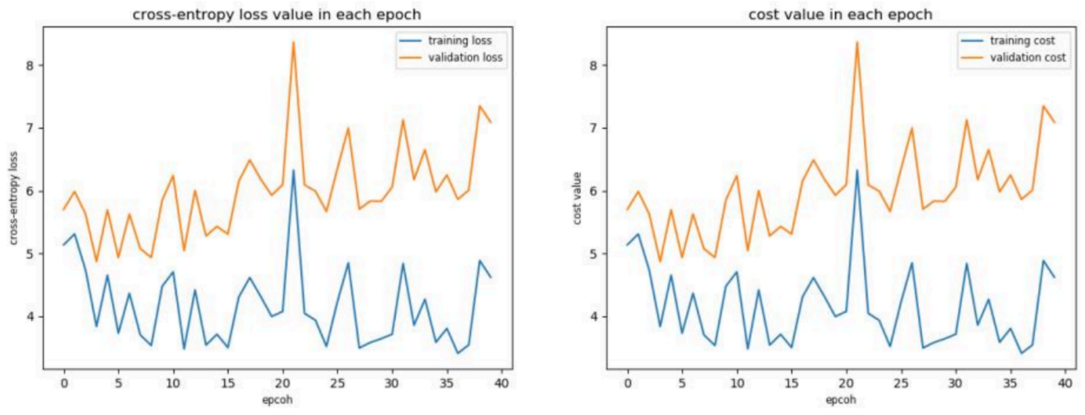

Figure 1: Figure representation of cross-entropy and cost value in each epoch

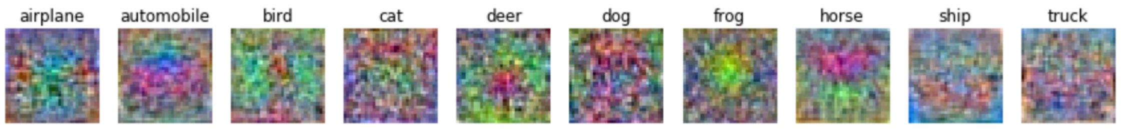

Figure 2: Visualization of learned weight from Model 1

2.2 Model 2

First I tested the accuracy for this model for each step (Training, Validation, and Testing) and got the following:
Training: 44.94%, **Validation:** 37.59%, **Testing:** 38.11%

Then using these values for the hyperparameters, I got these plots for model 2.
<u>Hyperparameters:</u>
Total Epoch: 40, **Batch Size:** 100, **λ:** 0, **η**: 0.001

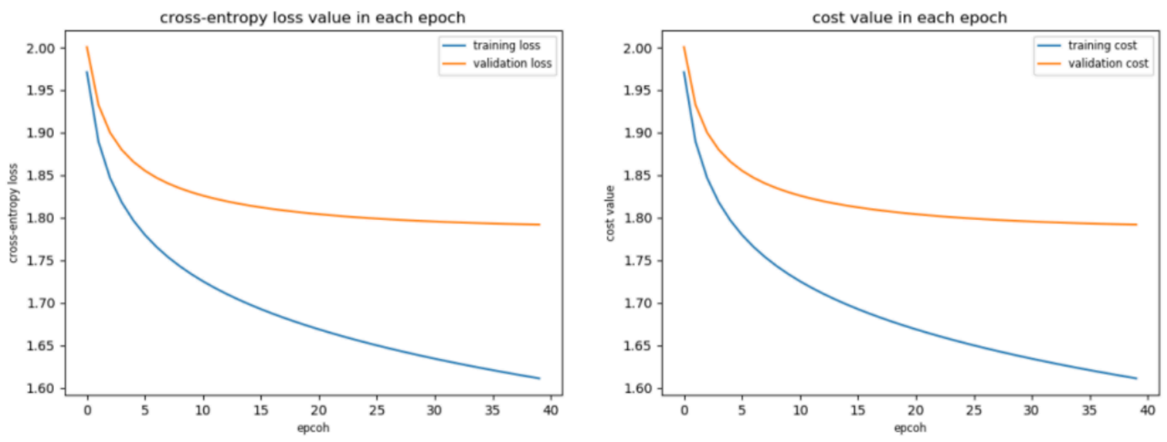

Figure 3: Figure representation of cross-entropy and cost value in each epoch

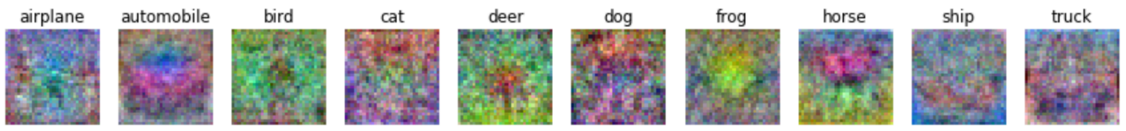

Figure 4: Visualization of learned weight from Model 2

2.3 Model 3

First I tested the accuracy for this model for each step (Training, Validation, and Testing) and got the following:
Training: 43.08%, **Validation:** 37.11%, **Testing:** 38.96%

Then using these values for the hyperparameters, I got these plots for model 2.
<u>**Hyperparameters:**</u>
Total Epoch: 40, **Batch Size:** 100, **λ:** 0.1, **η:** 0.001

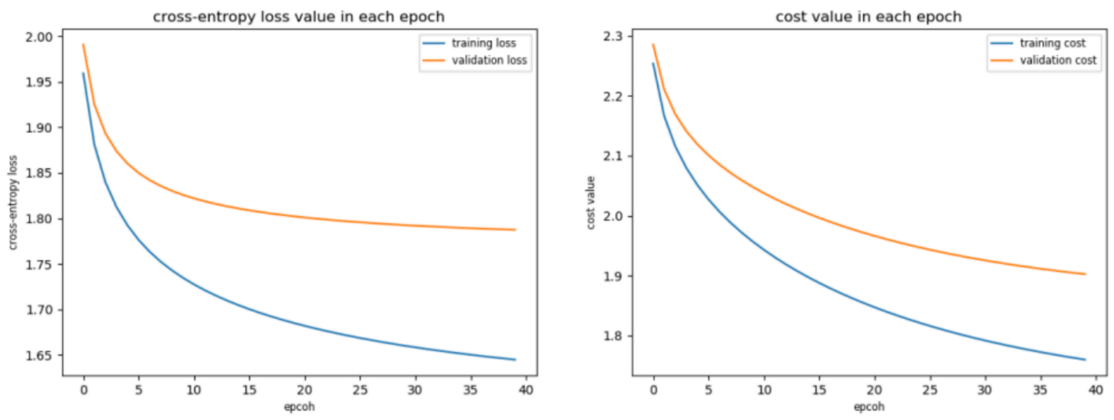

Figure 5: Figure representation of cross-entropy and cost value in each epoch

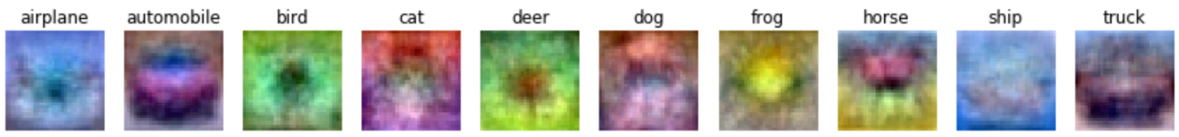

Figure 6: Visualization of learned weight from Model 3

2.4 Model 4

First I tested the accuracy for this model for each step (Training, Validation, and Testing) and got the following:
Training: 39.95%, **Validation:** 36.74%, **Testing:** 37.82%

Then using these values for the hyperparameters, I got these plots for model 2.
<u>Hyperparameters:</u>
Total Epoch: 40, **Batch Size:** 100, **λ:** 1, **η:** 0.001

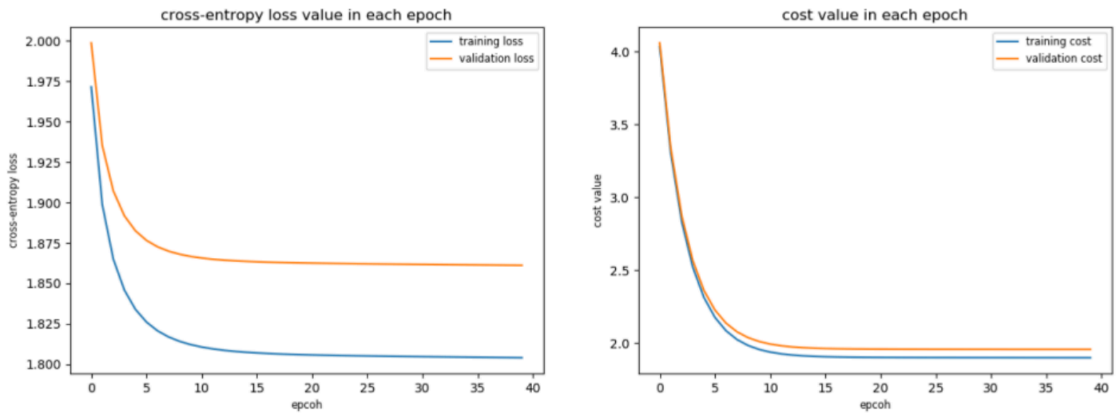

Figure 7: Figure representation of cross-entropy and cost value in each epoch

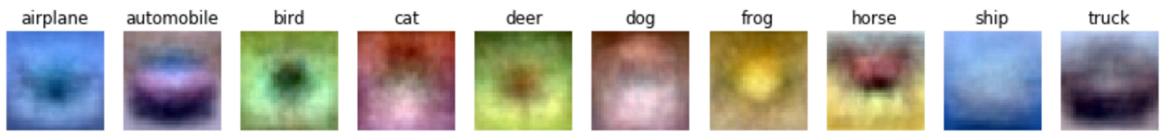

Figure 8: Visualization of learned weight from Model 4

3. Discussion

Exploring the impact of the learning rate (η) can be done by a comparative analysis of Model 1 which is using a learning rate of 0.1 and Model 2, with a much smallaer learning rate of 0.001. Interestingly, Model 2, with its lower learning rate, did a lot better, reaching an accuracy of 38.11% on the tests, which beats Model 1's score of 27.21%. You can see the difference yourself in Figure 4 for Model 2 and Figure 2 for Model 1, showing us how important the right learning rate is.

Both models didn't use any form of regularization, so when we look at the graphs showing their loss over time, the patterns are pretty much the same. However, Model 1, with its higher learning rate, doesn't really get better at lowering its error as it trains. On the other hand, Model 2 does a much better job as it trains, thanks to its smaller learning rate. This can best be explained by an example. If the learning rate is too high, it's like trying to find a hidden treasure by taking huge leaps – you might jump right over it. But with a smaller learning rate, it's more like taking careful steps towards the treasure, making it much easier to find.

I looked at how changing λ affects training in models 2, 3, and 4, with λ set to 0, 0.1, and 1, respectively. λ is a form of setting that kind of controls how complicated a model is. Turning this setting down (a low λ) lets the model do its thing without much restriction which can lead to the model overfitting to the data, but cranking it up (a high λ) tightens the reins, preventing the model from getting too carried away, which is something one can do to stop it from overfitting.

Looking at how accurate each model was, I noticed that as I increased λ from 0 in Model 2 to 1 in Model 4, the model's accuracy on the training data dropped from 44.94% to 39.95%. Also, as we turned up λ, the difference in performance on the training and validation data started to get closer together, which is shown in Figures 3, 5, and 7. This suggests that a higher λ can make the model learn faster, reaching a good level of performance more quickly.

But. If we go overboard with λ, the model starts to oversimplify, practically ignoring some of the data it's given. This problem is hinted at in Figure 8, where the model with a high λ (Model 4) produces a lot of black points, showing it's not paying much attention to those parts of the data.