Deep Learning (DD2424) Assignment 1

Badi Mirzai

April 2021

1 Intro

In this assignment, we train a single layer perception on the CIFAR-10 data set, which contains 10000 images of labeled objects. The input layer is 3072 pixeles values from each image with softmax as activation function for the output layer. The classification of the image is done by choosing the output with the highest predicted probability. The network is trained with gradient descent through back propagation of the cross entropy loss. The trained network achieved an accuracy of 39.13% on the test dataset with no regularization and 37.64% with regularization ($\lambda = 1$).

2 Gradient checks

In this assignment, the gradients of the neural network was checked with numerical approximations. The gradients was first checked with the first 50 parts of the input dimensions of the 100th image, which yielded an maximum absolute error in the magnitude of 10^{-8} (for the weights and bias derivatives) and an maximum relative error in the order of 10^{-7}. These were the comparisons with the fast numerical gradients. For the slow numerical gradients that was much more precise, the absolute error descried to the order 10^{-10}.

The error for the entire 100th image was also computed. The absolute error was in the order or 10^{-7} and 10^{-8} for the weights and bias respectively (with the fast approximator). The maximum relative error was in the magnitude of 10^{-10} for both weights and biases. Similarly for the slow-grad check, all the maximum errors was in the order of $2.5 * 10^{-6}$ or lower.

I am quite certain than the gradients were correctly calculated, based on the tests and results obtained.

3 Trained Network results

The models was trained on the CIFAR-10 data-set, where mini-batches of size 100 was used. The training data was shuffled for each epoch, where the loss and accuracy was calculated each epoch for the training and validation data

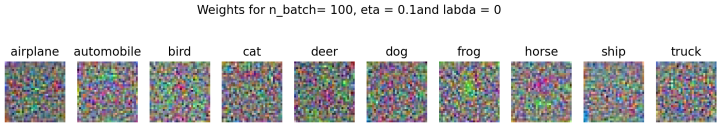

Figure 1: Visualization of the weights of the trained network with $\lambda = 0$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.1$.

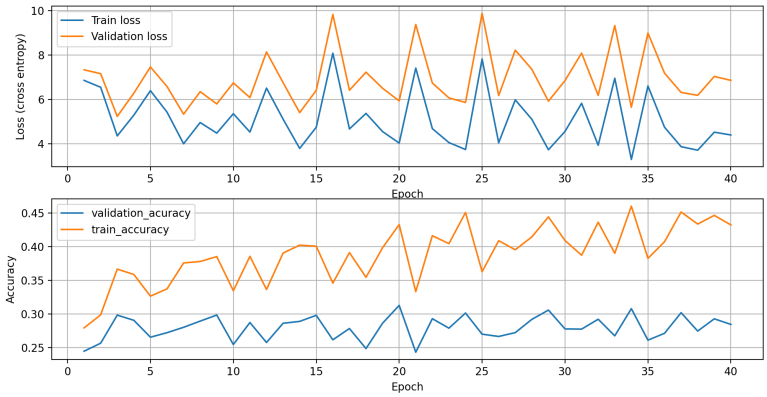

Figure 2: Learning curves with $\lambda = 0$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.1$.

sets. The input data was pre-processed by first normalized to one (division by 255). The mean and variance of the training data was calculated and used as a way to center the training, test and validation input data for increasing the performance. For the different runs, we have

1. Hyperparameters: $\lambda = 0$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.1$. Accuracy on test set is : 0.2925 Cost on test set is : 6.8402

2. Hyperparameters: $\lambda = 0$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$ Accuracy on test set is : 0.3913 Cost on test set is : 1.7571

3. Hyperparameters: $\lambda = 0.1$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$ Accuracy on test set is : 0.397 Cost on test set is : 1.8671

4. Hyperparameters: $\lambda = 1$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$ Accuracy on test set is : 0.3764 Cost on test set is : 1.9291

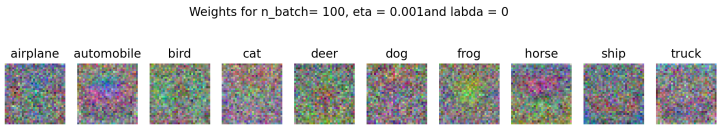

Figure 3: Visualization of the weights of the trained network with $\lambda = 0$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$.

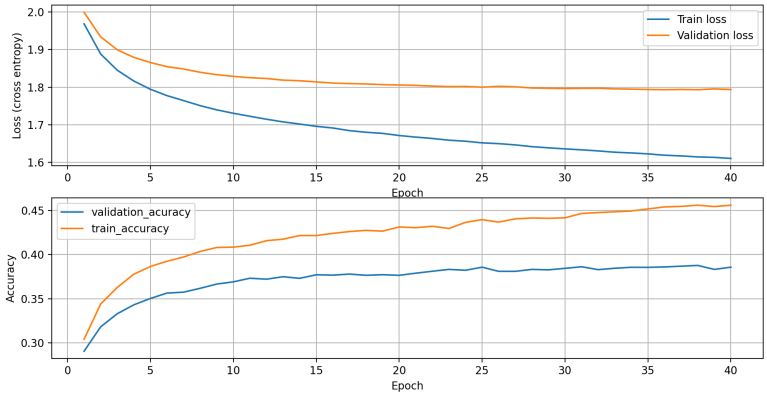

Figure 4: Learning curves with $\lambda = 0$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$.

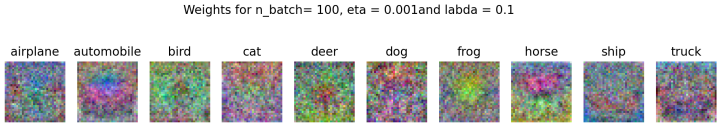

Figure 5: Visualization of the weights of the trained network with $\lambda = 0.1$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$.

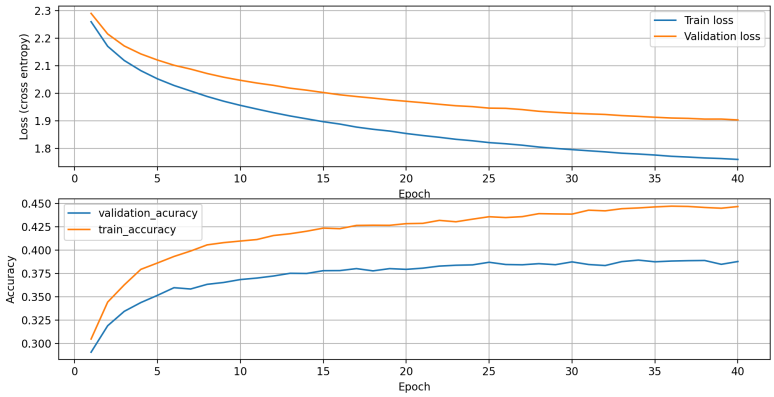

Figure 6: Learning curves with $\lambda = 0.1$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$.

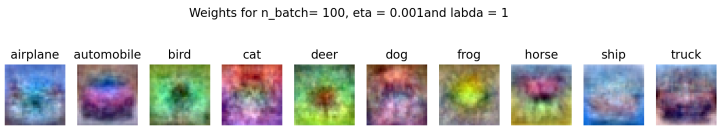

Figure 7: Visualization of the weights of the trained network with $\lambda = 1$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$.

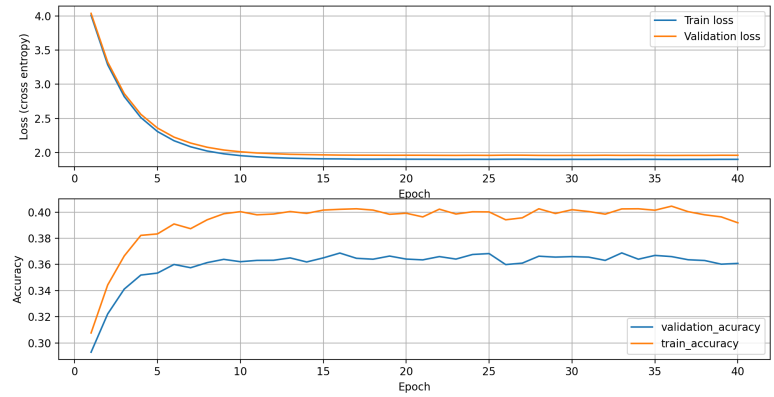

Figure 8: Learning curves with $\lambda = 1$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$.

4 Discussion and conclusion

After training a single layer perception, the results show the importance of tuning and finding the correct hyperparameters settings. First of all having a too small learning rate will result in a slower convergence of the trained Neural Network with can be unpractical. However, Figure 2 show how too large of an learning rate results to the loss never converging due to the the step size being too large. We can clearly see the difference in Figure 4, where the loss is decreasing and accuracy increasing. Even though the accuracy is high in the second hyperparameter setting, Figure 4, there is a high bias between the training and validation set. Using a larger λ will increase the loss overall, but decrease the the bias (difference) between the loss and accuracy between the training and validation curves. In other words will the model be less over-fitted since it will be trained less for the specific images and more in general. The vizualization of the weights in figure 3-7 shows how the regularization term λ creates less noise in the learned model is will be more robust to noise in the trained image and thus more generalized. This can be seen where for example Figure 5 has higher accuracy in the test data set than Figure 7, but the weights for $\lambda = 1$ shows a better visualization of the trained labels. It is also worth mentioning that the pre-processing of the data also increases the performance of the model, since it will crease a more smooth variation in the input data. Shuffling the data is important whn using mini batches, (especially when using batch normalization later) since the model won't be dependent of the order of input data. This is because of the average loss computed for each mini batch can vary depending on the permutation of the data.