# Report for Assignment 3

## DD2424 Deep Learning in Data Science

Jiaying Yang

950826-9124

## 1. Gradient check

In this assignment, I deal with a problem to train, validate and test a k layer network with multiple outputs to classify images from the CIFAR-10 dataset. The whole assignment is realized in these steps: loading data, initialization, computing cost and gradient, realizing mini-batch, adding momentum and training the network.

This exercise 1 is to check whether the gradients calculated by myself has the same value as that calculated from the functions given by the teacher. To check the gradients, for both cases, I use 100 out of 3072 dimensions and 100 out of 10000 training images. And λ is set to 0 to avoid wasting time doing numerically computation. The gradients are checked for the cases of a 2-layer network and a 3-layer network, and the relative errors between the results of the two cases are shown in the table below, which are small enough to indicate the correctness of my functions for calculating gradients.

| network | $e_{W1}$ | $e_{b1}$ | $e_{W2}$ | $e_{b2}$ | $e_{W3}$ | $e_{b3}$ |
|---|---|---|---|---|---|---|
| two-layer | 5.235e-10 | 5.063e-11 | 3.275e-09 | 8.643e-11 | - | - |
| three-layer | 1.693-09 | 1.969e-10 | 3.867e-09 | 1.332e-10 | 1.367e-09 | 1.409e-10 |

**Table 1. The gradient check for two-layer and three-layer network**

## 2. Three-layer fully-connected network

In this part, I implemented the three-layer fully connected neural network with and without batch normalization, respectively. I use the data in the file data_batch_1.mat for training, the file data_batch_2.mat for validation and the file test_batch.mat for testing. The experiments (with and without batch normalization) are set with the same hyperparameters as follows:

| | |
|---|---|
| regularization term λ | 0.001 |
| learning rate η | 0.2 |
| batch size | 100 |
| the number of epochs | 20 |
| momentum parameter ρ | 0.9 |

**Table 2. Hyperparameters setting**

The accuracies for the three-layer fully connected neural network with and without normalization are shown in the table below:

| | with normalization | without normalization |
|---|---|---|
| training accuracy | 55.67% | 56.62% |
| validation accuracy | 40.35% | 44.57% |
| test accuracy | 40.38% | 45.21% |

**Table 3. Accuracies of the three-layer network**

The visualization of loss function with and without batch normalization is shown below, where the figure on the right is the one without batch normalization, and that on the left is the one with normalization.

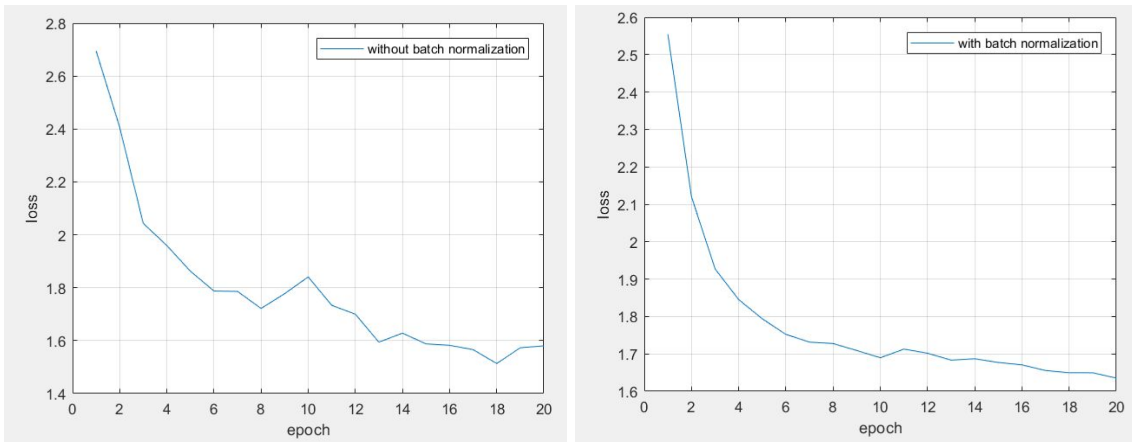


**Figure 1. Loss functions with and without batch normalization**

## 3. Coarse and fine search

During the training process,  coarse to fine random search are performed to discover good/effective values for the hyperparameters. Below are the results for coarse search and fine search.

### 3.1 Coarse search

The range of lambda I set is from 10^-6 to 10^-1, and the range of eta I set is from 10^-4 to 10^1. The number of epochs used is 20. The number of iterations for generating random parameters is set to 100. The best three performing networks and their hyper-parameter settings are:

| | **lambda** | **eta** | **Validation Accuracy** |
|---|---|---|---|
| **1** | 0.00195(10^-2.711) | 0.0186(10^-1.731) | 44.23% |
| **2** | 0.00920(10^-2.036) | 0.0123(10^-1.911) | 43.34% |
| **3** | 0.00543(10^-2.265) | 0.00849(10^-2.071) | 43.14% |

**Table 4. The three best validation accuracy and parameter settings during coarse search**

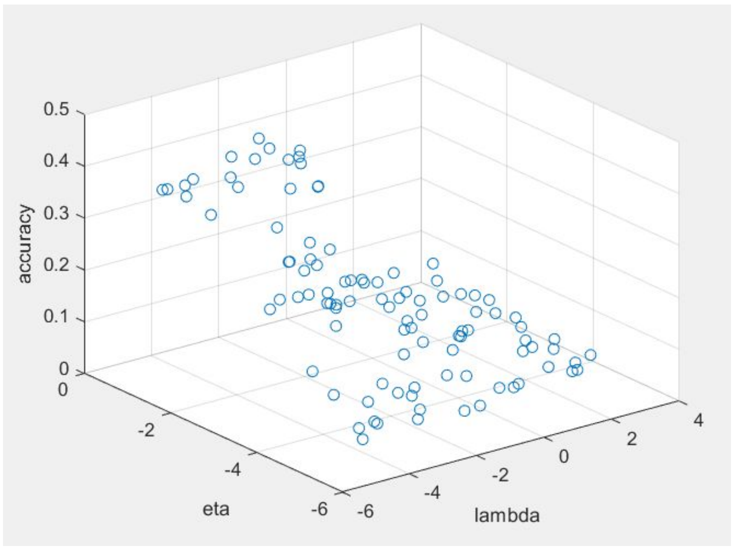


**Figure 2. Validation accuracy with respective to lambda and eta after coarse search**

## 3.2 Fine search

The range of lambda I set is from 10^-6 to 10^-1, and the range of eta I set is from 10^-2.5 to 10^0.5. The number of epochs used is 50. The number of iterations for generating random parameters is set to 100. The best three performing networks and their hyper-parameter settings are:

| | lambda | eta | Validation Accuracy |
|---|---|---|---|
| **1** | 0.00644(10^-2.191) | 0.0302(10^-1.520) | 46.45% |
| **2** | 0.00532(10^-2.274) | 0.0307(10^-1.513) | 46.07% |
| **3** | 0.00622(10^-2.206) | 0.129(10^-0.890) | 45.72% |

**Table 5. The three best validation accuracy and parameter settings during fine search**

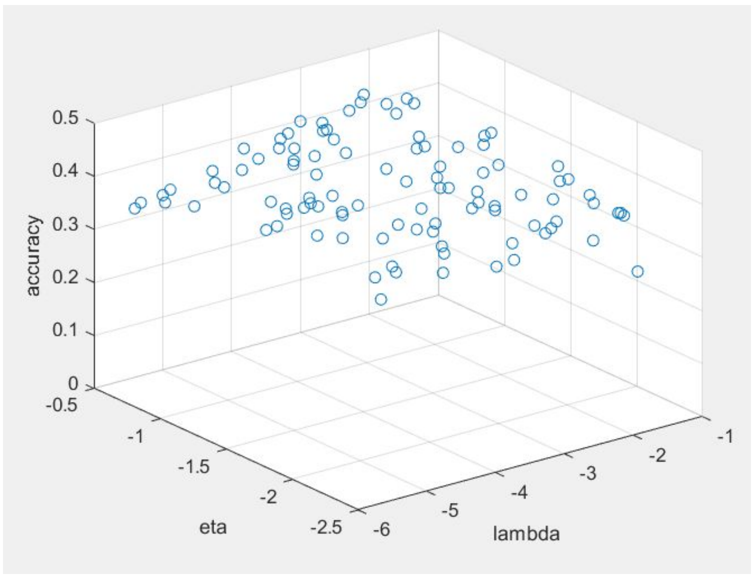


**Figure 3. Validation accuracy with respective to lambda and eta after fine search**

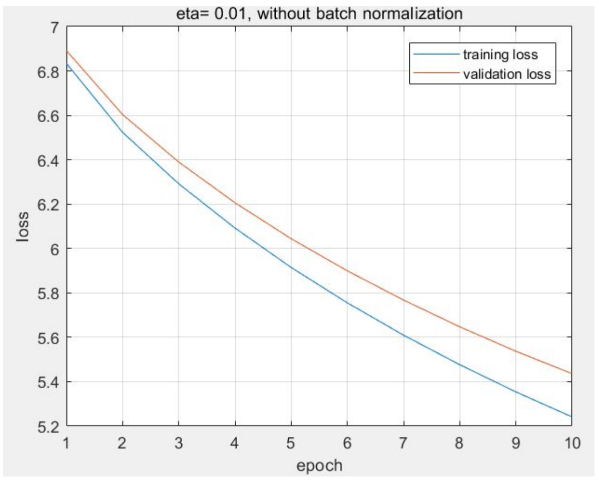
## 4. two-layer network with and without batch normalization

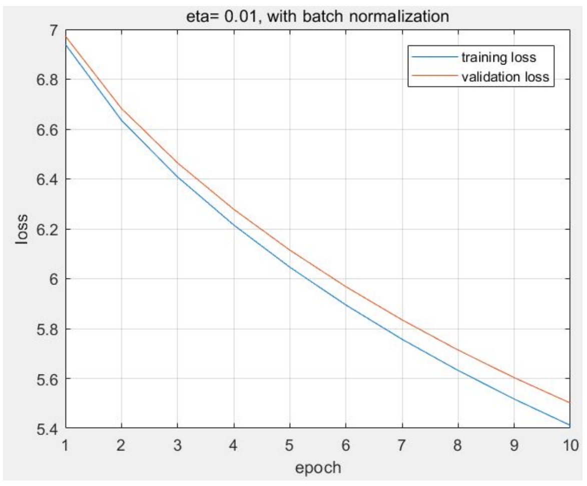
In this section, I run a two-layer network with 3 different learning rates (small=0.01, medium=0.1, high=0.3) for 10 epochs, to compare its results with and without batch normalization. The accuracies in these different cases are:

| | | training accuracy | validation accuracy | test accuracy |
|---|---|---|---|---|
| η =0.01 | without batch norm | 47.03% | 39.54% | 40.61% |
| | with batch norm | 42.21% | 38.26% | 39.86% |
| η =0.1 | without batch norm | 54.88% | 39.33% | 40.03% |
| | with batch norm | 57.03% | 43.93% | 44.49% |
| η =0.3 | without batch norm | 27.36% | 24.68% | 25.79% |
| | with batch norm | 47.44% | 40.13% | 40.87% |

**Table 6. Accuracies without and with batch normalization with 3 different learning rates**

As shown in the table above and figure below, when learning rate is small, the network with normalization has worse accuracies compared with the case without normalization; but as the learning rate increases to a medium value, we can find out the benefit of using batch normalization; when learning rate is high, without batch normalization, the loss will become very large at the beginning, then drop dramatically, and the accuracies are very bad, while with batch normalization, the loss will not start from a large value and will decrease more smoothly to a relative low value.





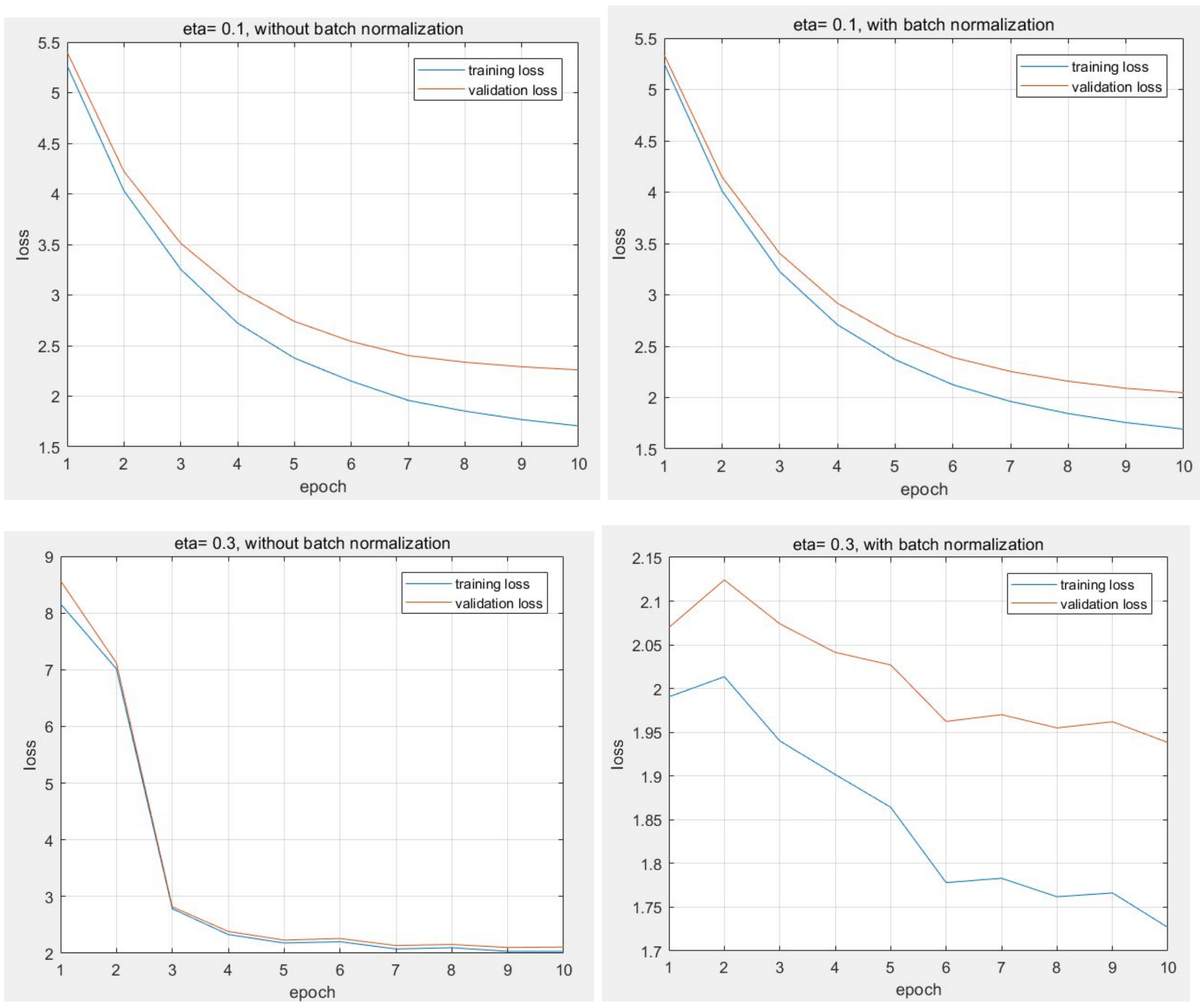


**Figure 4. Training and validation loss without and with batch normalization with 3 different learning rates**