Programming 3 for ECSE 6965

Lingyu Zhang, RIN:661950104

April 5, 2018

Theoretical part

The objective of this problem is that we utilize CNN network to construct a classifier for CIFAR10 dataset. For this problem, we have the training dataset which contains 50000 samples, each sample is a $32 * 32 * 3$ matrix, For the label y, we construct it as one-of-k coding, if label is 2, then y is denoted as below

$$y = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

For the input layer, the dimension is $32 * 32 * 3$
For the first convolutional layer, the dimension is $28 * 28 * 32$. The number of weight parameters for the convolutional layer 1 is 800.

$$dim(Conv1) = \frac{32 - 5}{1} + 1 = 28$$

For the first max-pooling layer, the dimension is $14 * 14 * 32$

$$dim(pool1) = \frac{28}{2} = 14$$

For the second convolutional layer, the dimension is $10 * 10 * 32$. The number of weight parameters for the convolutional layer 2 is 800.

$$dim(Conv2) = \frac{14 - 5}{1} + 1 = 10$$

For the second max-pooling layer, the dimension is $5 * 5 * 32$

$$dim(pool2) = \frac{10}{2} = 5$$

For the third convolutional layer, the dimension is $3 * 3 * 64$. The number of weight parameters for the convolutional layer 2 is 576.

$$dim(Conv2) = \frac{5-3}{1} + 1 = 3$$

The dimension of output layer is $10 * 1$
For the output layer, the number of weight parameters for the output layer is 5760
We use cross entropy loss for the training task, the cross entropy loss can be represented as below:

$$\triangledown L[m] = -\sum_{k=1}^{K} y[m][k] log\hat{y}[m][k]$$

Model architecture

The whole model consists of one input layer, three convolutional layers and one output layer. For each convolutional layer, we add ReLu activation layer, and we also utilize normalization to prevent overfitting. The model architecture is shown as below:

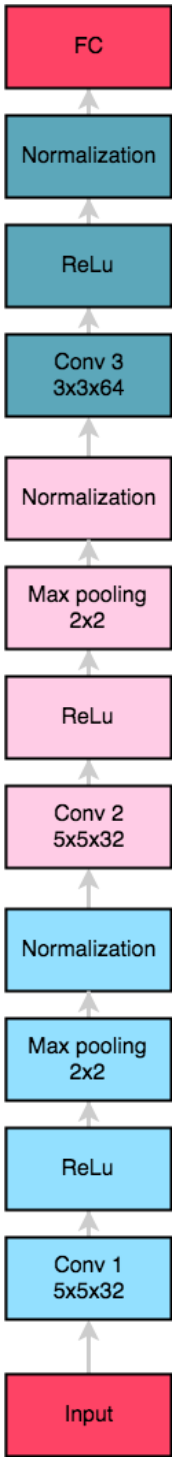


Figure 1: Model architecture by Lingyu

Experimental part

Firstly we initialize the parameters, the weight matrix is initialized by xavial initializer provided by tensorflow, for the bias vector, we initialized them as zero vectors. We tune the hyper-parameters and the optimal ones are as below:
For the learning rate, we give a initial learning rate value and perform an exponential decay for every 3500 global steps with a decay rate 0.1.

$$\eta = \eta_{inital} * 0.1^{\frac{gs}{3500}}$$

where gs represent the current global step. We set the initial learning rate as below

$$\eta_{inital} = 0.0001$$

For the optimizer, we use RMSprop optimizer with

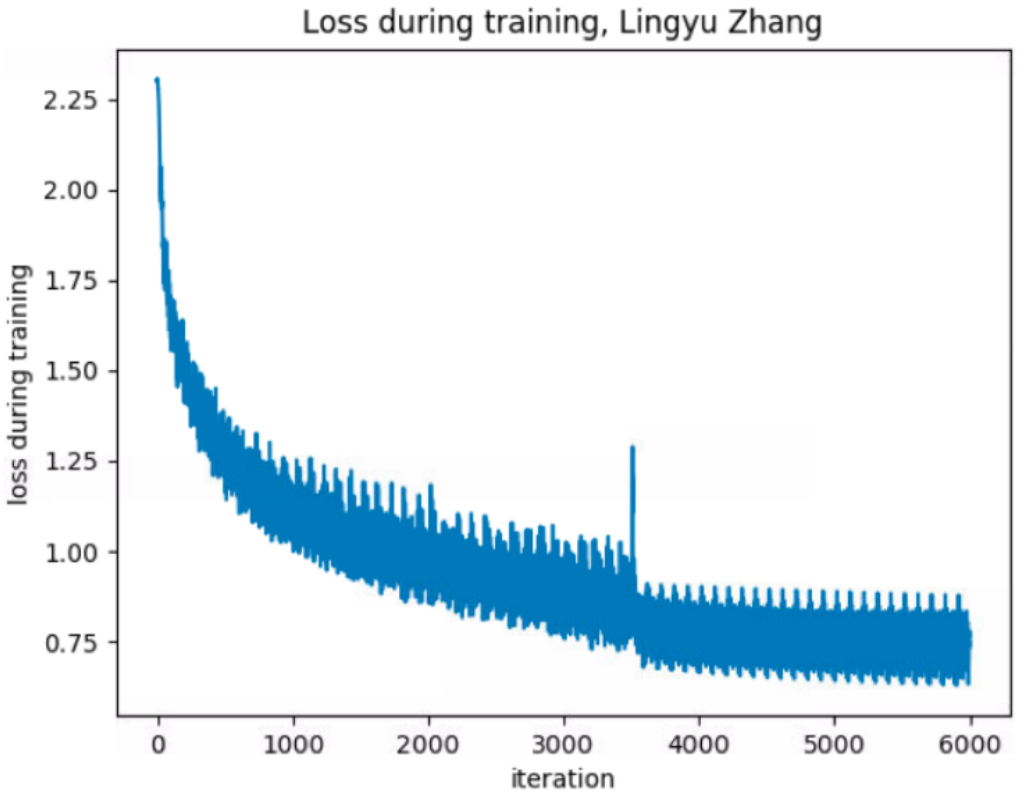
$$decay = 0.0005, momentum = 0.9, epsilon = 1e - 10$$

For the batch size and training epochs, we set the values as below:

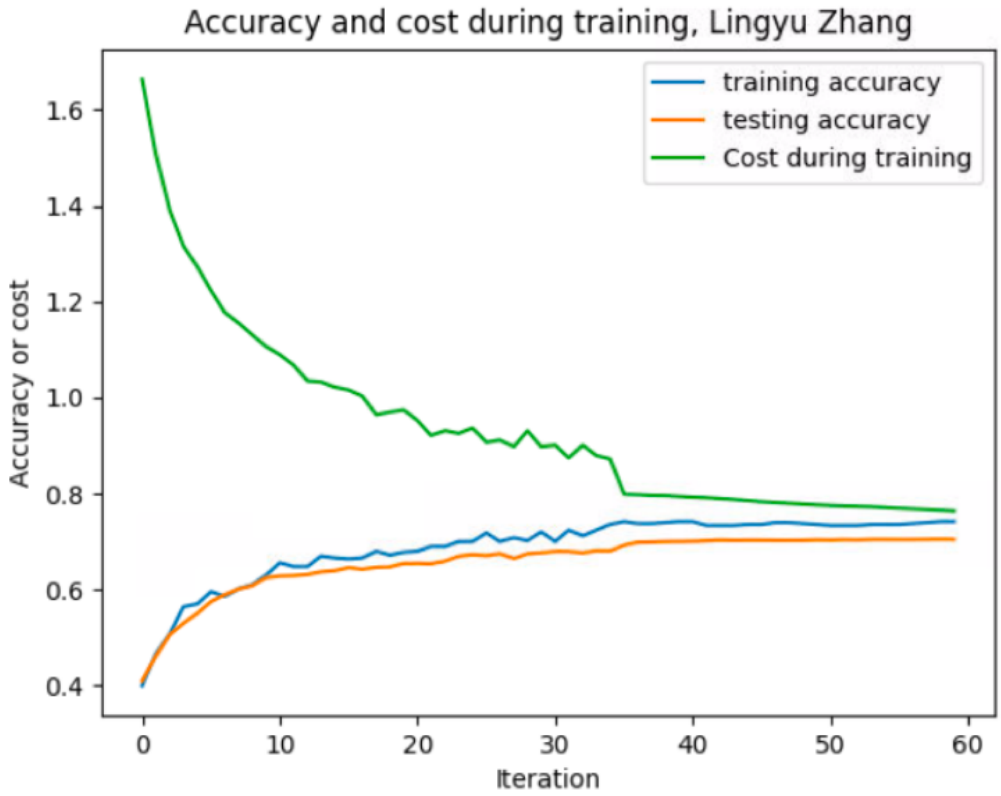
$$\text{batch size} = 500$$

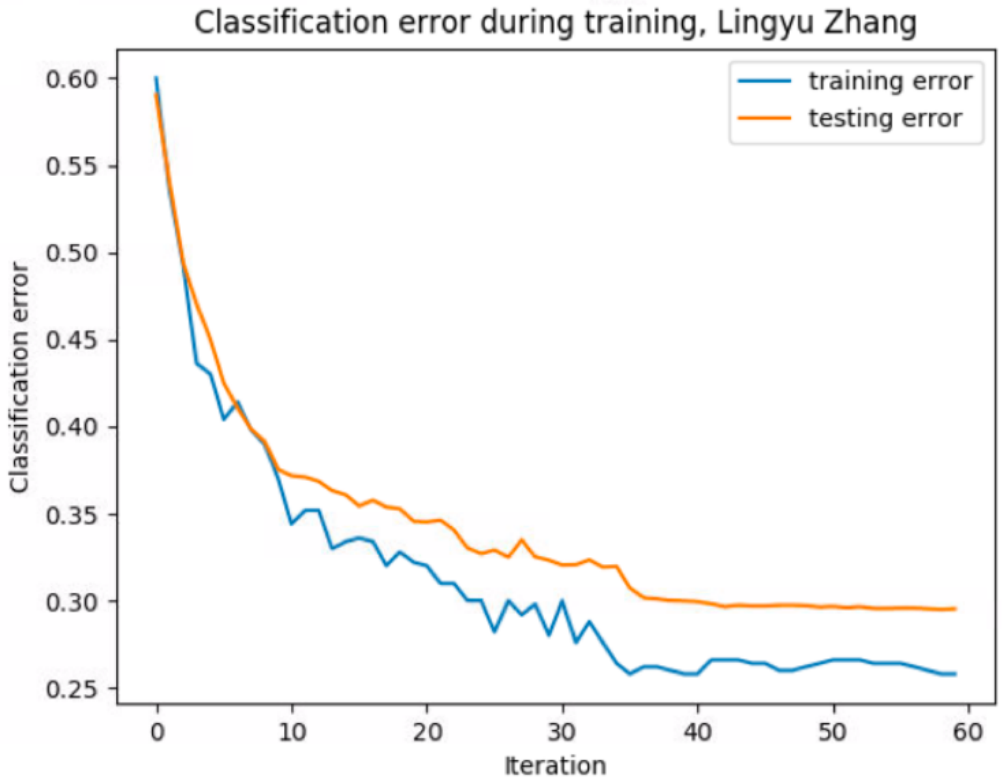
$$\text{training epochs} = 60$$

Below is the training loss over iterations



We can find that the loss is going down with the training iteration increases and finally converged, and at the 3500-th global step, there is a decay for learning rate.
For the experimental part, we use GPU to train, and the result is as below: Below is the prediction accuracy and prediction error plot for training set and testing set over iterations





We can find that the prediction accuracy for both training set and testing set is increasing with the training process going on, and finally converged. And the training accuracy is slightly higher than the testing accuracy, which is reasonable.

Classification result

Below is the training and testing accuracy.

```
epoch e= 59 training accuracy [ 0.74199998]
epoch e= 59 testing accuracy [ 0.70459998]
```

Belos is the confusion matrix for the testing result

```
[[344   7  33  16  11   4   5   9  41  18]
 [ 20 368   3   7   6   4  11   7  24  55]
 [ 30   4 313  34  36  34  34  15   5   7]
 [ 13   3  38 268  27  89  29  13  10   7]
 [  8   0  29  33 332  25  33  32  10   5]
 [ 11   2  22  92  18 301  13  22   3   4]
 [  2   0  25  34  18   8 397   5   2   0]
 [  9   1  15  27  30  34   3 367   2   7]
 [ 34   7   8   2   5   5   2   0 422  19]
 [ 11  26   5  15   4   4  11   6  20 411]]
```

Therefore,

$$error(class = 0) = 1 - \frac{344}{344 + 7 + 33 + 16 + 11 + 4 + 5 + 9 + 41 + 18} = 29.508196721311\%$$

$$error(class = 1) = 1 - \frac{368}{20 + 368 + 3 + 7 + 6 + 4 + 11 + 7 + 24 + 55} = 27.128712871287\%$$

$$error(class = 2) = 1 - \frac{313}{30 + 4 + 313 + 34 + 36 + 34 + 34 + 15 + 5 + 7} = 38.8671875\%$$

$$error(class = 3) = 1 - \frac{268}{13 + 3 + 38 + 268 + 27 + 89 + 29 + 13 + 10 + 7} = 46.076458752515\%$$

$$error(class = 4) = 1 - \frac{332}{8 + 0 + 29 + 33 + 332 + 25 + 33 + 32 + 10 + 5} = 34.516765285996\%$$

$$error(class = 5) = 1 - \frac{301}{11 + 2 + 22 + 92 + 18 + 301 + 13 + 22 + 3 + 4} = 38.319672131148\%$$

$$error(class = 6) = 1 - \frac{397}{2 + 0 + 25 + 34 + 18 + 8 + 397 + 5 + 2 + 0} = 19.144602851324\%$$

$$error(class = 7) = 1 - \frac{367}{9 + 1 + 15 + 27 + 30 + 34 + 3 + 367 + 2 + 7} = 25.858585858586\%$$

$$error(class = 8) = 1 - \frac{422}{34 + 7 + 8 + 2 + 5 + 5 + 2 + 0 + 422 + 19} = 16.269841269841\%$$

$$error(class = 9) = 1 - \frac{411}{11 + 26 + 5 + 15 + 4 + 4 + 11 + 6 + 20 + 411} = 19.883040935673\%$$

$$avg(error) = 1 - \frac{344 + 368 + 313 + 268 + 332 + 301 + 397 + 367 + 422 + 411}{5000} = 29.54\%$$

We can find that the average classification error is 29.54%
Below is the visualized convolutional filters for convolutional layer 1.

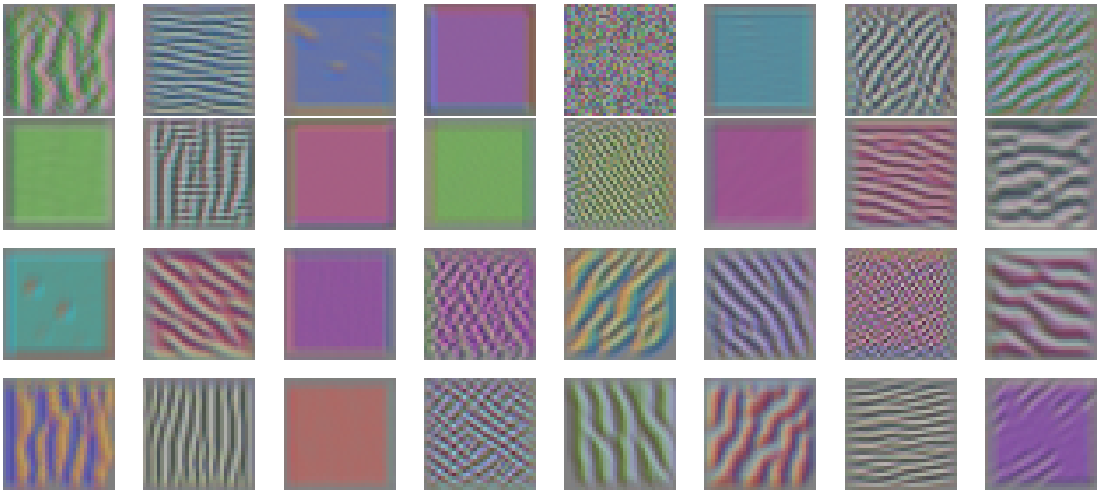


Figure 2: Visualization of all the filters in the convolutional layer 1

Submission notes from Lingyu

I use a newer version of tensorflow so my model is saved as the following 4 files

$$checkpoint$$

$$model - 6000.data - 00000 - of - 00001$$

$$model - 6000.index$$

$$model - 6000.meta$$

In my submission, I've included all the checkpoint I saved during the training, the final optimal model is the index 6000 version. We should always use model-6000 for testing and evaluation.