ECE194N HW2

Deren Lei
derenlei@umail.ucsb.edu

Question 1 (XOR)

(a)

(b)

Structure:

Input: 32*32*3

Convolution layer 1:

Kernel size: 5*5*3

Activation function: relu

Normalization and Pooling Layer 1: [2*2]

Convolution layer 2:

Kernel size: 5*5*64

Activation function: relu

Normalization and Pooling Layer 1: [2*2]

Fully connected layer:

Neurons: 1024

Output layer:

Neurons: 10

Activation function: softmax

Loss function: cross-entropy loss

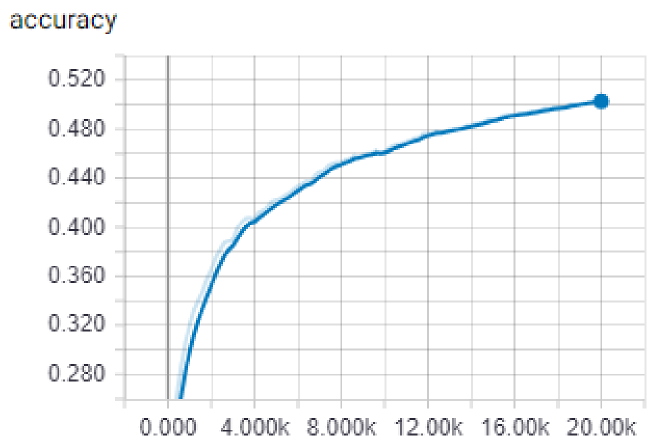
I tried to process image into gray scale and compare the result of RGB images and gray scale images has -5% worse accuracy on predictin the testing data. The kernel size in both convolution layer and neurons in the fully connected layer are chosen by following the instruction tutorial in tensorflow cnn (MNIST dataset). I choose activation function as relu after several experiments which shows relu has better performance. The output layer must have 10 neurons because the output is one-hot representation. The activation function for outputlayer is softmax because it canconvert to probability value between 0 and 1. I choose cross-entropy loss beacause it measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. The normalization is added because it improved the accuracy slightly by around 2% percent. Note: The structure on tensorflow cifar10's official website, it uses two fully connected layer and image is preprocessed by:

1. Random flip
2. Corp to 28*28
3. Approximately whitened
4. Randomly distort the image brightness.
5. Randomly distort the image contrast.

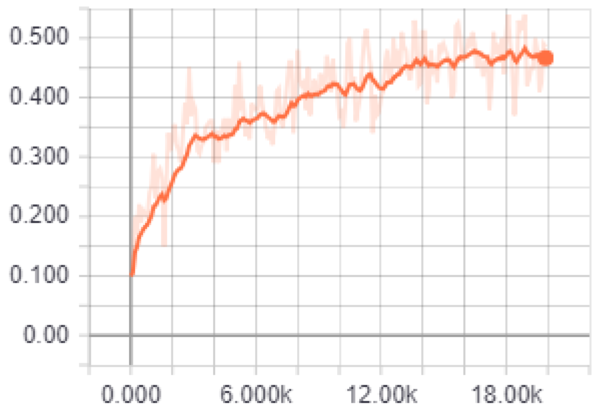
The accuracy improved to 20%. One important reson is the images are processed to create more training data when training. The extra fully connected layer also brings a great help. However, due to the training time, the extra fully connected layer is not implemented when training the model. I believe the accuracy will increase for around 10%. In conclusion, the model gets the correct accuracy as it suppose to be and it is implemented correctly.

20000 Iteration:

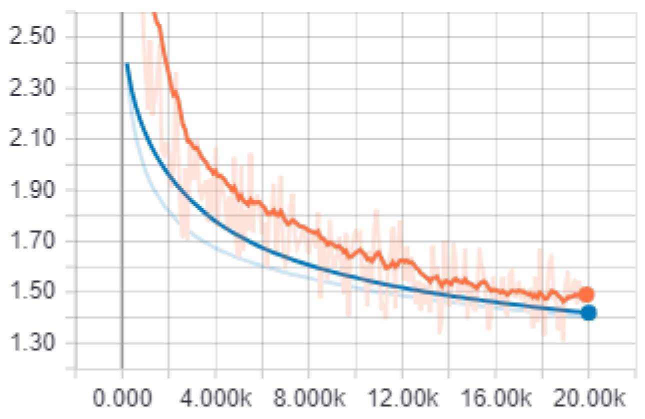
Testing Data Accuracy during each iteration:



Training data accuracy during each iteration:

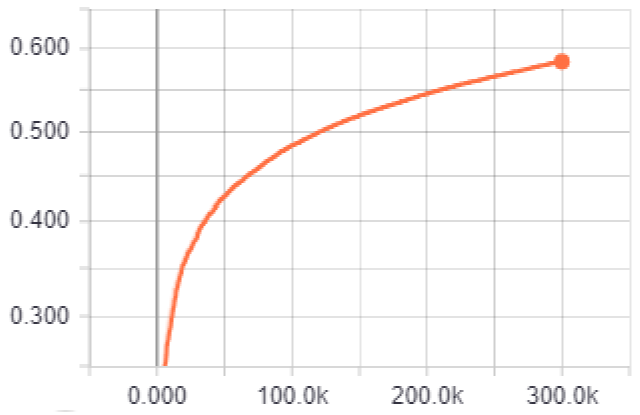


Loss during each iteration (blue for training data and orange for training data):

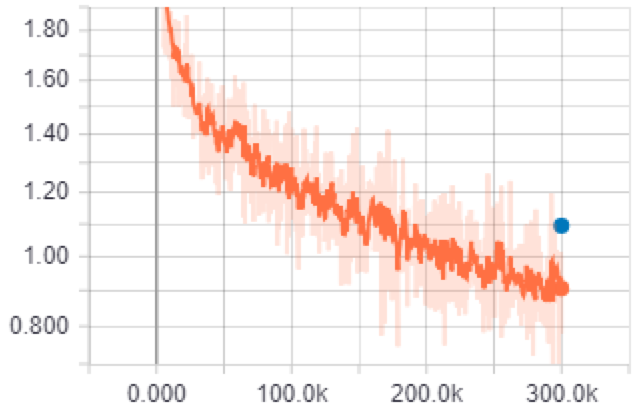


300000 Iteration (without calculating testing data's loss and accuracy during each iteration):

Training accuracy:



Training loss:



Note: evaluating testing data's accuracy costs double time so the graph with evaluating it only contains relatively small number of iterations.

(c)

Convolution layer 1:

Weight: 3*5*5*64=4800

Bias: 1*64=64

Convolution layer 2:

Weight: 64*5*5*64=102400

Bias: 1*64=64

Fully connected layer 1:

Weight: 64*8*8*1024=4194304

Bias: 1*1024=1024

Output layer:

Weight: 1024*10=10240

Bias:1*10=10

Total:

Weight: 4800+102400+4194304+10240=4311744

Bias: 64+64+1024+10=1162

Total number of free, trainable parameters:

8495808+1162=4312906

(d)

The wrongly-classified sample is in the following order:
['airplane','automobile','bird','cat','deer','dog','frog','horse', 'ship','truck']