EE6337: Deep Learning, Spring 2019

Indian Institute of Technology Hyderabad

HW 1, 100 points. Assigned: Sunday 24.02.2019. **Due: Sunday 03.03.2019 at 11:59 pm.**

In this assignment you will build on the functions you wrote in the previous assignment to implement an image classifier. Use the MNIST database for this assignment. *Use Tensorflow only to load data from the database. Do NOT use Tensorflow for any other part of this assignment.* The steps for implementing your classifier are outlined below.

1. Code to load MNIST data

```
"""
Load the MNIST dataset into numpy arrays
Author:  Alexandre Drouin
License:  BSD
"""
import numpy as np
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
X_train = np.vstack([img.reshape((28, 28)) for img in mnist.train.images])
y_train = mnist.train.labels
X_test = np.vstack([img.reshape(28, 28) for img in mnist.test.images])
y_test = mnist.test.labels
del mnist
```

2. *Reshape* the numpy training and test input arrays to 28×28 to form the input images to your network.
3. Refer to *slide 21* in the DL slide deck (uploaded on classroom) for the architecture of your CNN. Construct your network using the functions implemented in the previous assignment.
4. The code to load training data does a *one-hot encoding* of the labels. So, use *cross-entropy loss* to find the error at the softmax output layer.
5. Implement the *back propagation* algorithm for computing gradients.
6. Update the weights of the network using two approaches:
   (a) *Batch gradient descent*: divide the training data into *batches*. Experiment with different batch sizes. The gradient is computed and the weights are updated after passing each batch through the network.
   (b) *Stochastic gradient descent*: the gradient is computed and the weights are updated after each sample is passed through the network.

   Ensure that your update step handles pooling layers.
7. *Shuffle* the training data after one *epoch* i.e., after all the training data points have been passed through the network once. Go back to step 5. Do this for 15 epochs. You can experiment with the number of epochs as well.
8. After each epoch, compute the *error* on the training and test data sets. *Plot* the training and test errors as a function of epochs.
9. *Visualize* the activation maps after each epoch. You can pick a couple of representative slices from the activation volumes at each of the convolution layers.
10. Also report the *accuracy* of your classifier at the end of the each epoch.
11. Note: If you have trouble with training this network on your computer reduce the depth of convolution layers suitably.