Deep Learning and Its Applications to Signal and Image Processing and Analysis

Exercise 1:

The main aim of the following exercise is to run TensorFlow and creates new layers in a given code.

1. Download the code from our web-site https://wwwee.ee.bgu.ac.il/~rrtammy/DNN/DNN.html

Tensor Flow (by Ohad Shitrit)

__This file contains all the functions shown in the Lab.

** The code was taken from TensorFlow homepage, so you can find the original repository there.

2. Go over the Class **mnist** (see mnist/mnist.py) and try to understand each member function

3. The main script, which used the mnist class, is **fully_connected_feed.py**. Read it and try to run it on your computer (*TensorFlow should be installed)

** In order to run the code, from the "tensorflowPresentation" directory run:

```
python mnist/fully_connected_feed.py
```

It is very important to run it from the main directory. The reason is that a "log" directory is assumed to be exists in your working directory.

4. Change the "inference" function in mnist.py class to be as follows:

a. Instead of two hidden layers, create 3 convolutional layers with 3x3xN filters. N could be any number (Possible combination is 16, 32, 64). You can find examples in **mnist/convolutional.py** script.

b. After the first two convolutional layers, add **max polling layers** with kernel size of [1,2,2,1] and strides of [1,2,2,1], padding='SAME'.

c. The activation function after the max polling will be ReLU.

d. The last layer is fully-connected with 10 outputs (It is already in the code)

The network should look like:

Conv -> ReLU -> Pool -> Conv -> ReLU -> Pool -> Conv -> ReLU - > FC

e. Run the main script again and examine the results.

Notes:

- the "inference" function includes the variables declaration and the layers. For example, for Conv layer:

```
Weights = tf.Variable…
      Bias = tf.Variable…
conv = tf.nn.conv2d(data,
      conv_weights,
         strides=[1, 1, 1, 1],
         padding='SAME')
relu = tf.nn.relu(tf.nn.bias_add(conv, biases))
```

- *In order to feed the fully connected layer, you need to serialize your data, see example in the convlutional.py code (make it 1 dim vector)*

5. Run the script again using 3 differents learning rates (by multiples of 10) and watch the optimization process (The loss is already dumped out to TensorBoard).

6. You need to submit a print screen from TensorBoard of the following:
a. Loss value, for different learning rates
b. The network graph

** Note - You don’t have to get better results than the “regular” network.

Good luck