

Deep Learning 182, HW #1

Roy Uziel	Irit Chelly
203398854	021565510

April 28, 2018

1 Network architecture

In our graph we used convolutional layers. The input image is of size 28*28.

```
1 new_input = tf.reshape(input_images, [-1, 28, 28, 1])
```

We defined the following hidden layers:

1. Convolutional Layer 1:

We used 32 filters, each filter is a kernel of size 5*5. Each neuron in this layer is a result (scalar) of a convolution of each kernel centered on one neuron in the input layer. Thus, this layer consists of 28*28*32 neurons. We then compute the activation function relu on the result of each neuron:

```
1 conv1 = tf.layers.conv2d(  
2     inputs=new_input,  
3     filters=32,  
4     kernel_size=[5, 5],  
5     padding="same",  
6     activation=tf.nn.relu)
```

2. Pooling Layer 1:

Here we reduce the spatial size of the conv. layer by using a Max Pooling filter of size 2*2 and apply the maximum value of each 2*2 sized part of the image (Convolutional Layer 1):

```
1 pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[2,  
2], strides=2)
```

3. Convolutional Layer 2:

```
1 conv2 = tf.layers.conv2d(  
2     inputs=pool1,  
3     filters=64,  
4     kernel_size=[5, 5],  
5     padding="same",  
6     activation=tf.nn.relu)
```

4. Pooling Layer 2:

```
1 pool2 = tf.layers.max_pooling2d(inputs=conv2,  
    pool_size=[2, 2], strides=2)
```

5. Reshaping

```
1 pool2_flat = tf.reshape(pool2, [-1, 7 * 7 * 64])
```

6. Dense

```
1 dense = tf.layers.dense(inputs=pool2_flat, units  
    =1024, activation=tf.nn.relu)
```

7. Dropout

```
1 dropout = tf.layers.dropout(inputs=dense, rate=0.2)  
2 )
```

8. Output

```
1 logits = tf.layers.dense(inputs=dropout, units=10)
```

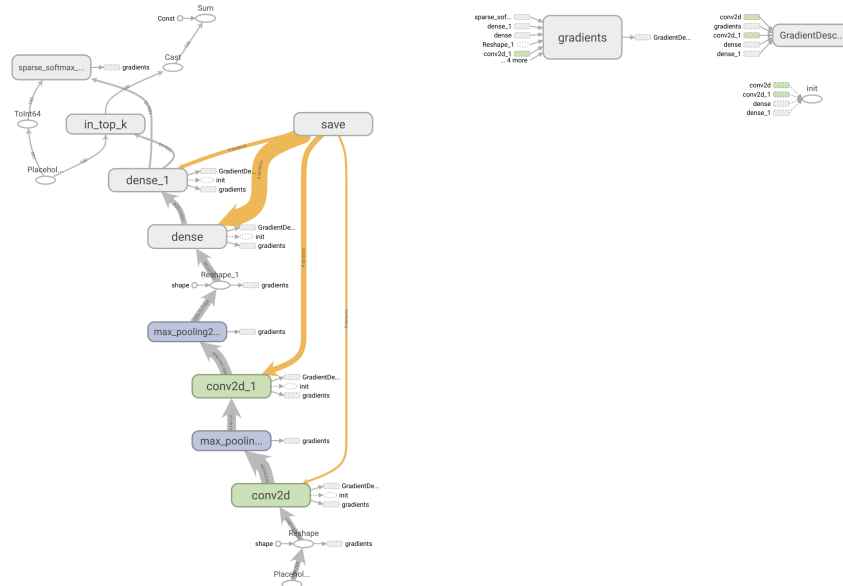


Figure 1: Tensorboard : Structure Graphs

5.2 attach a short document describing the network architecture and any other architectures that you tested

5.3 Make sure to have your full name and ID on the top of the document

5.4 It is recommended to add tensorboard screenshots that describe the results (Optional)

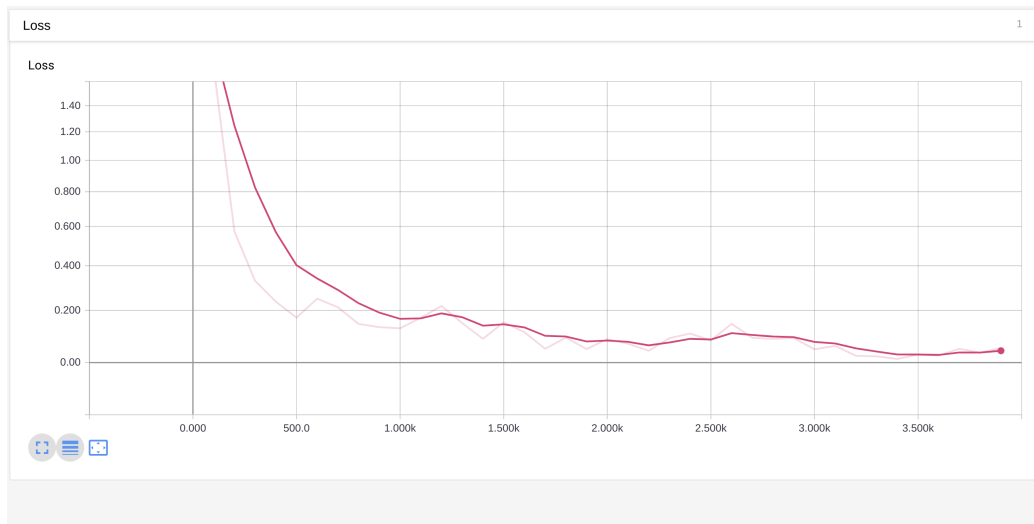


Figure 2: Tensorboard : Loss Over Time

```

parser.add_argument(
    '--max_steps',
    type=int,
    default=100,
    help='number of steps to run trainer.'
)
parser.add_argument(
    '--batch_size',
    type=int,
    default=100,
    help='batch size. Must divide evenly into the dataset sizes.'
)

```

```

Step 3700: loss = 0.07 (0.010 sec)
Step 3800: loss = 0.05 (0.011 sec)
Step 3900: loss = 0.07 (0.013 sec)
Training Data Eval:
Num examples: 55000 Num correct: 53900 Precision @ 1: 0.9800
Validation Data Eval:
Num examples: 5000 Num correct: 4906 Precision @ 1: 0.9812
Test Data Eval:
Num examples: 10000 Num correct: 9800 Precision @ 1: 0.9800
An exception has occurred, use %tb to see the full traceback.

SystemExit

```

Figure 3: Console Results