

# Deep Learning 182, HW #1

Roy Uziel  
203398854

Irit Chelly  
021565510

April 30, 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). After this step we will have 14\*14\*32 neurons.:

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

### 3. Convolutional Layer 2:

Here we used 64 filters, each filter is a kernel of size 5\*5:

```
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:

After this step we will have 7\*7\*64 neurons:

```
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:

This layer is fully connected: we picked 1024 neurons in this layer that are fully connected to the neurons from the previous layer ("Pooling Layer 2"):

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

## 7. Dropout:

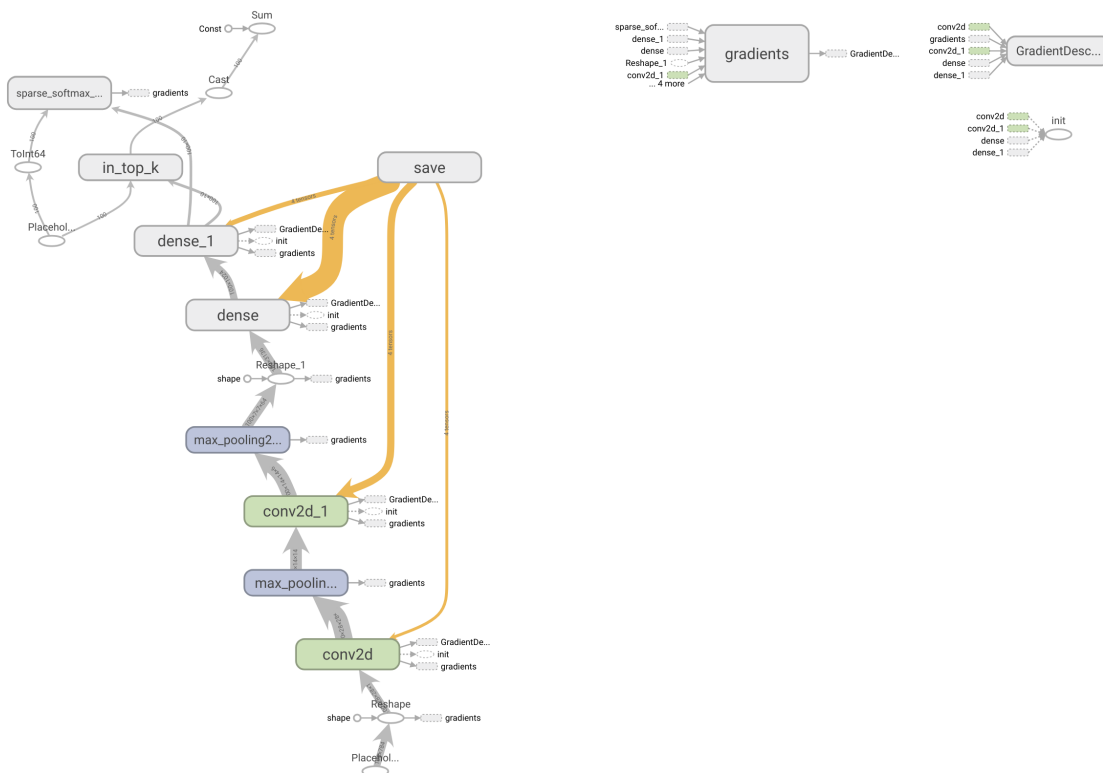
```
1 dropout = tf.layers.dropout(inputs=dense, rate=0.2)
```

## 8. Output:

The output layer is 10 neurons, each neuron is a classifier for one digit. This layer is full connected to the previous layer:

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

Figure 1: Architecture Description Graph



## 2 Results

Below are the script results after running with 100 batches and 4000 steps to run the trainer, reaching precision of 0.97:

Training Data Eval:

Num examples: 55000 Num correct: 53798 Precision @ 1: 0.9781

Validation Data Eval:

Num examples: 5000 Num correct: 4896 Precision @ 1: 0.9792

Test Data Eval:

Num examples: 10000 Num correct: 9778 Precision @ 1: 0.9778

An exception has occurred, use

Figure 2: Loss Function Graph

