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.

```
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:

   ```
       conv1 = tf.layers.conv2d(
               inputs=new_input,
               filters=32,
               kernel_size=[5, 5],
               padding="same",
               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.:

   ```
   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:

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

4. Pooling Layer 2:
   After this step we will have 7*7*64 neurons:

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

5. Reshaping:

```
        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”):

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

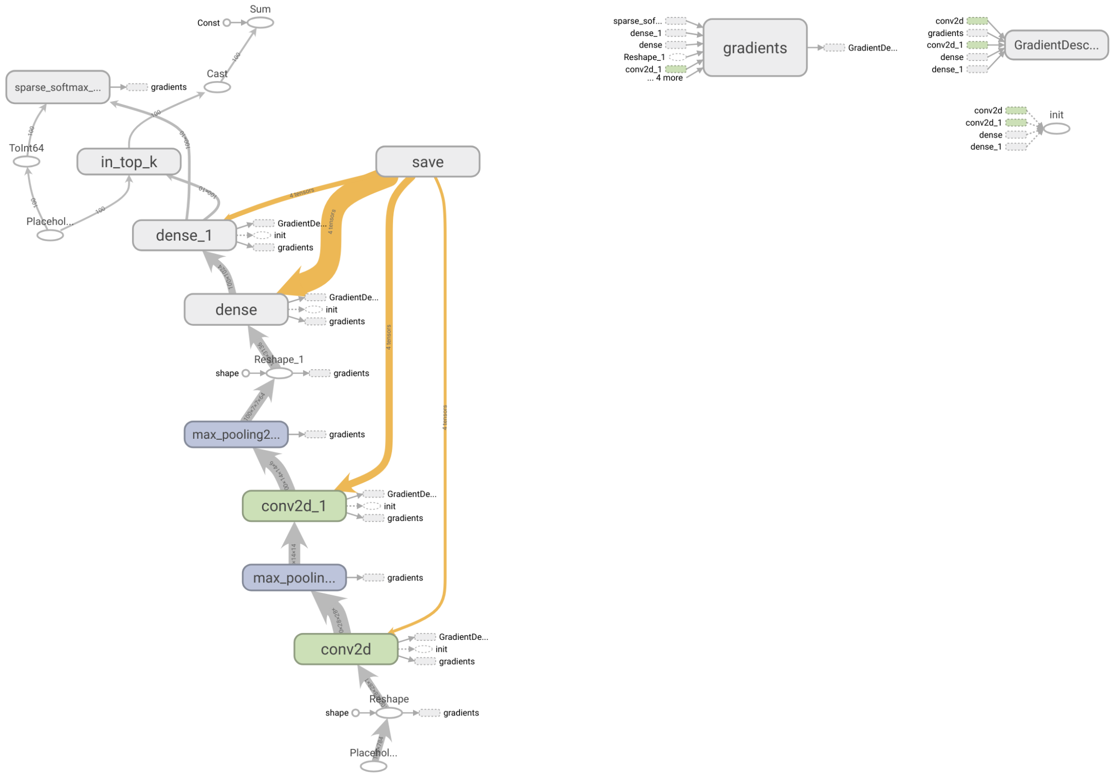
7. Dropout:

```
    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:

```
    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

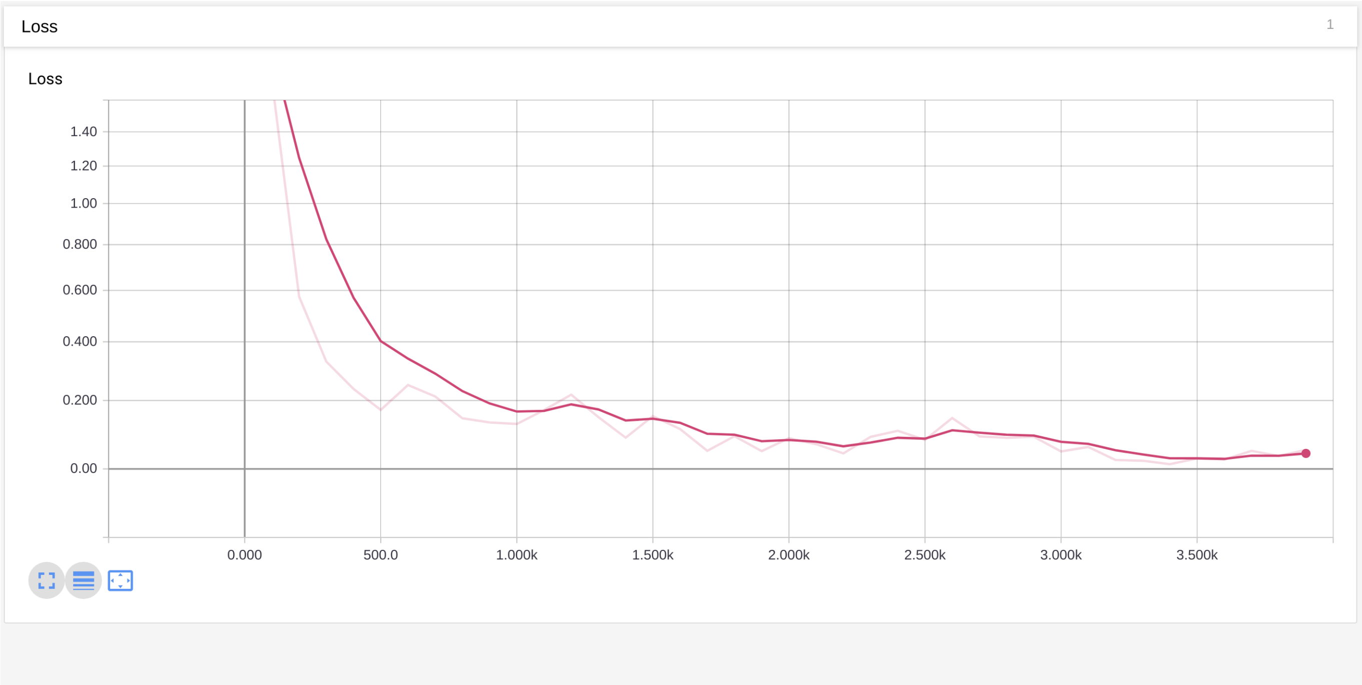
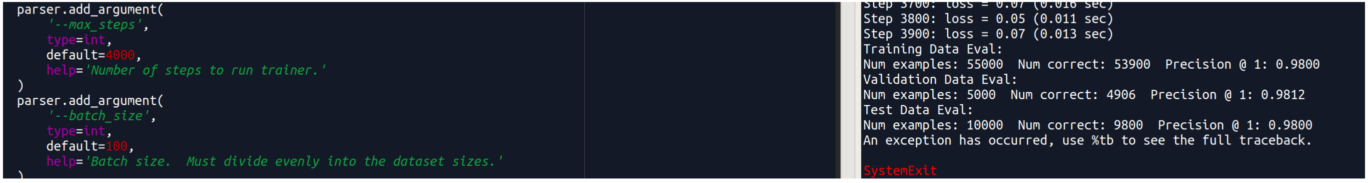


Figure 3: Console results and parameters:



```
parser.add_argument(
    '--max_steps',
    type=int,
    default=4000,
    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 (0.016 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
```