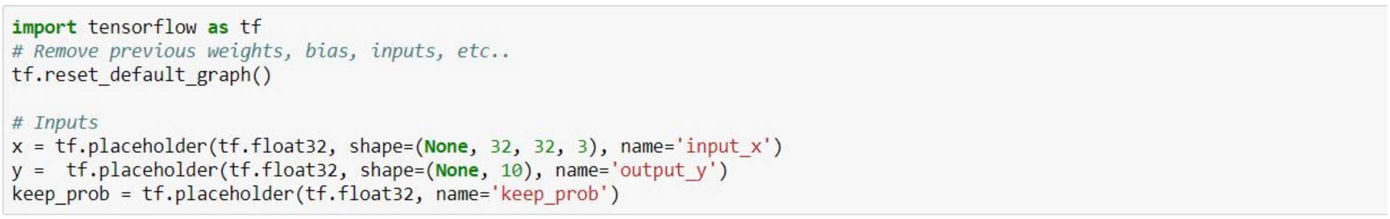
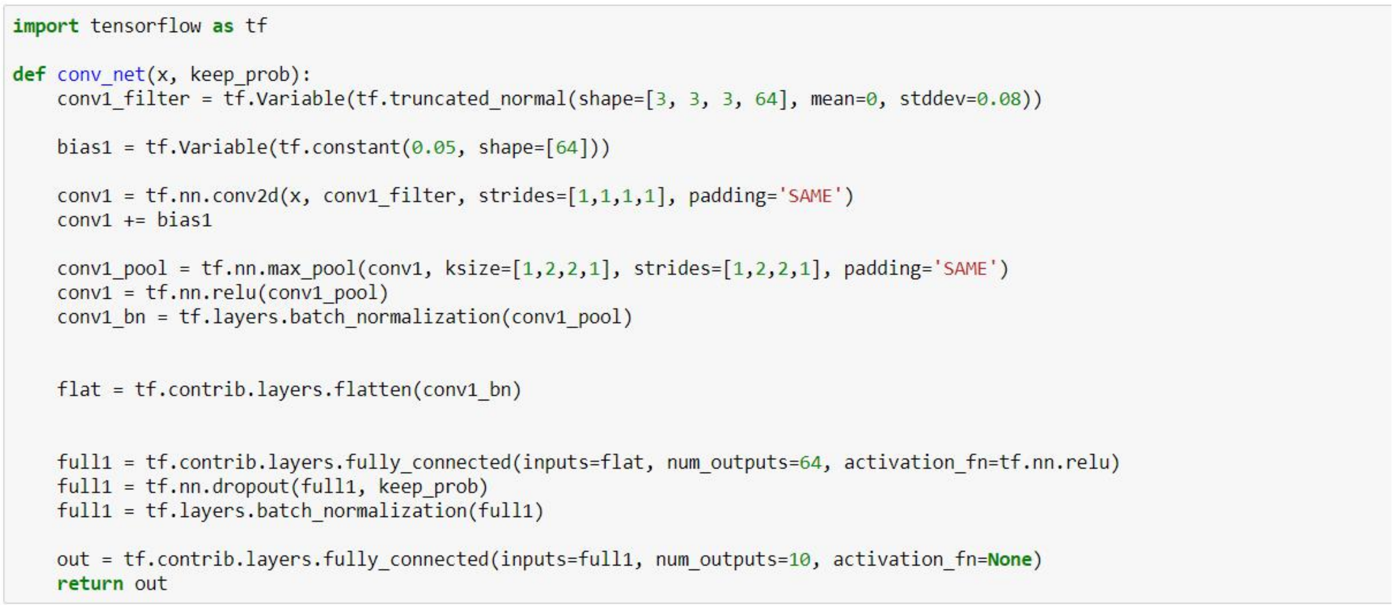
Distributed Data Analytics
Exercise Sheet # 6
Arooba Jamil Khokhar
278077

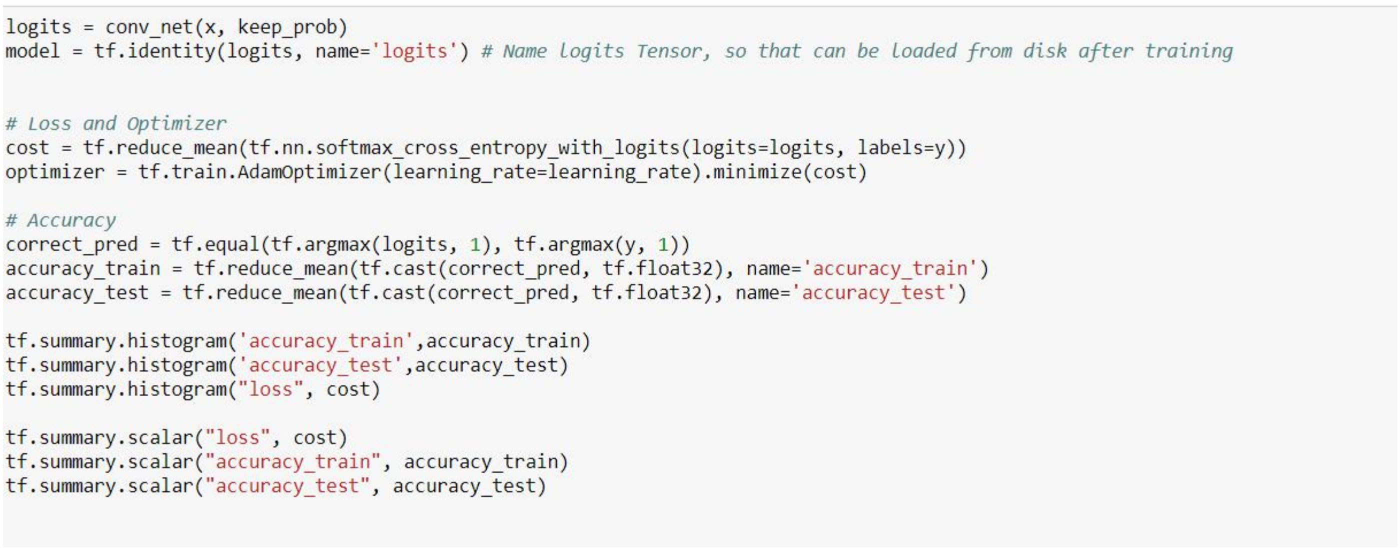
Exercise 1 Convolutional Neutral Network For the Image Classification

In the following cells we will setup our structure of CNN

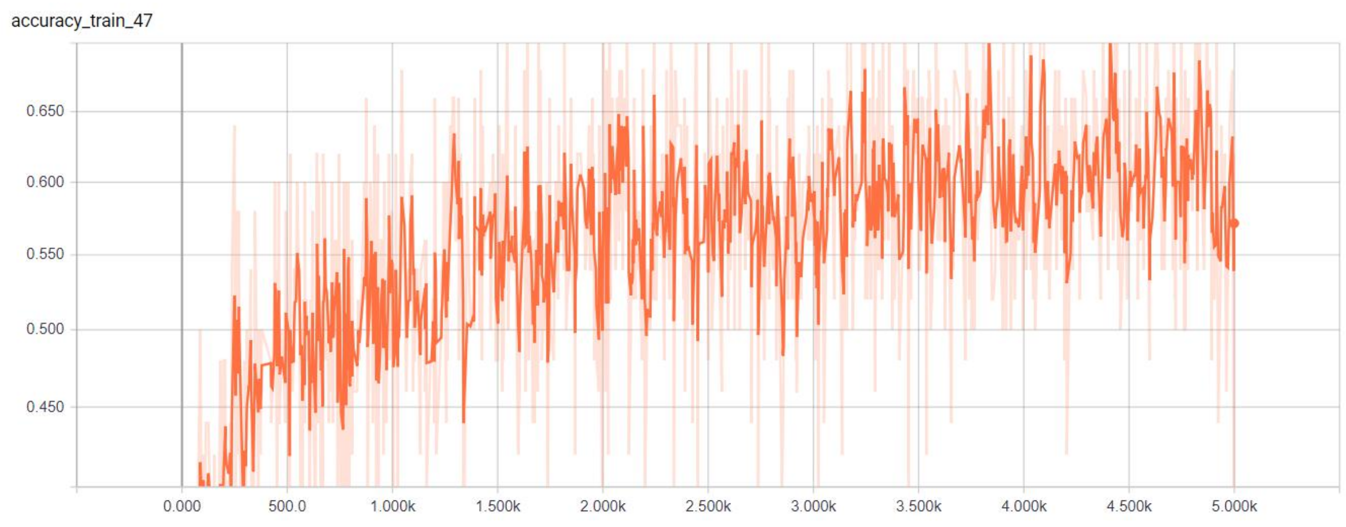

```
import tensorflow as tf
# Remove previous weights, bias, inputs, etc..
tf.reset_default_graph()

# Inputs
x = tf.placeholder(tf.float32, shape=(None, 32, 32, 3), name='input_x')
y =  tf.placeholder(tf.float32, shape=(None, 10), name='output_y')
keep_prob = tf.placeholder(tf.float32, name='keep_prob')
```


```
import tensorflow as tf

def conv_net(x, keep_prob):
    conv1_filter = tf.Variable(tf.truncated_normal(shape=[3, 3, 3, 64], mean=0, stddev=0.08))

    bias1 = tf.Variable(tf.constant(0.05, shape=[64]))

    conv1 = tf.nn.conv2d(x, conv1_filter, strides=[1,1,1,1], padding='SAME')
    conv1 += bias1

    conv1_pool = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')
    conv1 = tf.nn.relu(conv1_pool)
    conv1_bn = tf.layers.batch_normalization(conv1_pool)


    flat = tf.contrib.layers.flatten(conv1_bn)


    full1 = tf.contrib.layers.fully_connected(inputs=flat, num_outputs=64, activation_fn=tf.nn.relu)
    full1 = tf.nn.dropout(full1, keep_prob)
    full1 = tf.layers.batch_normalization(full1)

    out = tf.contrib.layers.fully_connected(inputs=full1, num_outputs=10, activation_fn=None)
    return out
```

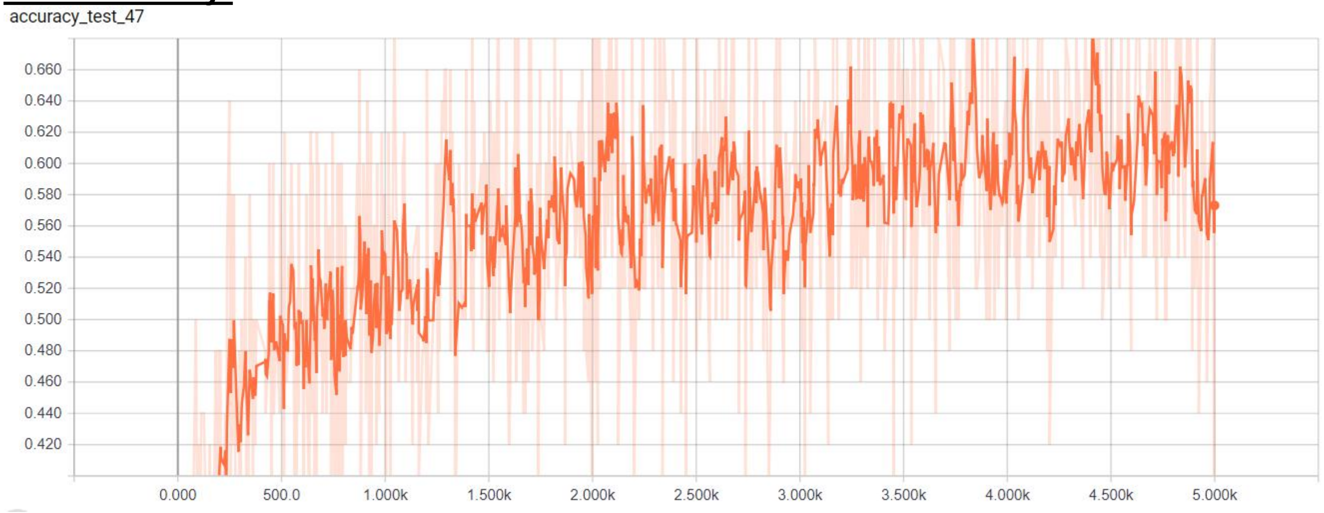
We initialize our CNN, define the optimizer and loss function and also accuracy


```
logits = conv_net(x, keep_prob)
model = tf.identity(logits, name='logits') # Name logits Tensor, so that can be loaded from disk after training


# Loss and Optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

# Accuracy
correct_pred = tf.equal(tf.argmax(logits, 1), tf.argmax(y, 1))
accuracy_train = tf.reduce_mean(tf.cast(correct_pred, tf.float32), name='accuracy_train')
accuracy_test = tf.reduce_mean(tf.cast(correct_pred, tf.float32), name='accuracy_test')

tf.summary.histogram('accuracy_train',accuracy_train)
tf.summary.histogram('accuracy_test',accuracy_test)
tf.summary.histogram("loss", cost)

tf.summary.scalar("loss", cost)
tf.summary.scalar("accuracy_train", accuracy_train)
tf.summary.scalar("accuracy_test", accuracy_test)
```

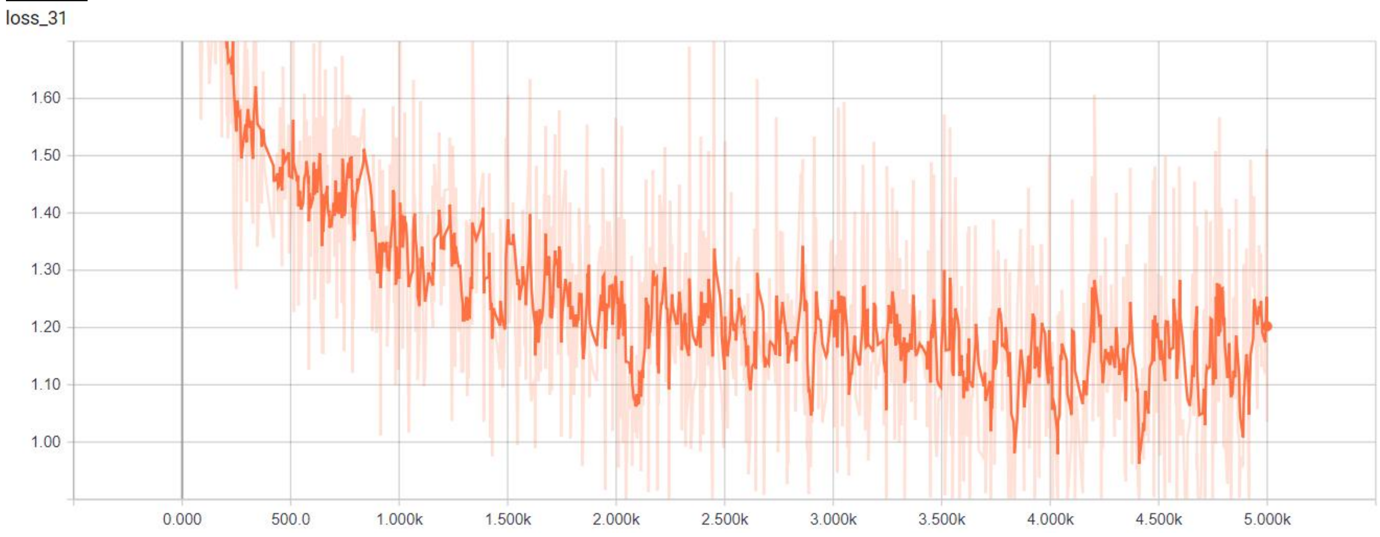
Training Accuracy:



Test Accuracy:

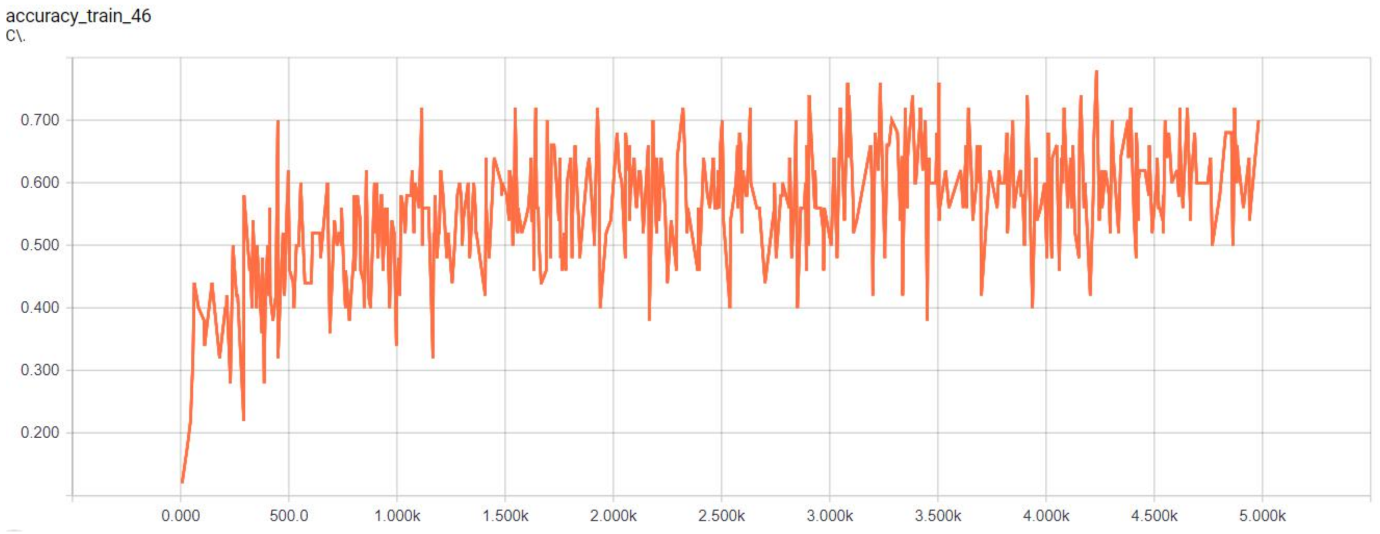


Cost:

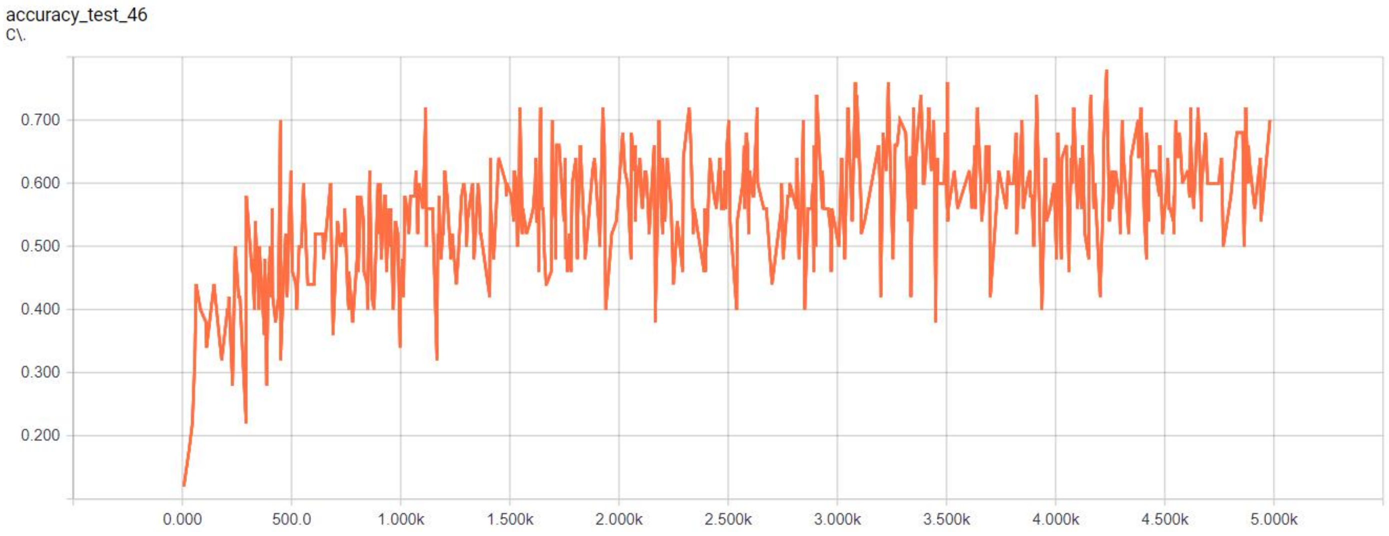


Distributions:

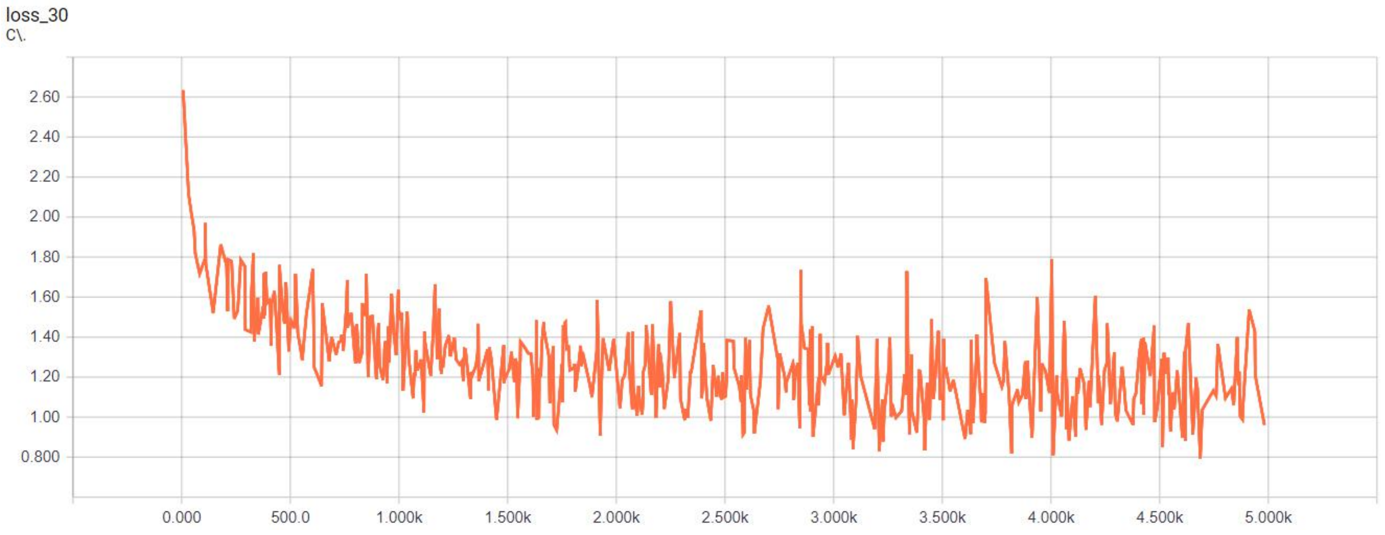
Training Accuracy:



Test Accuracy

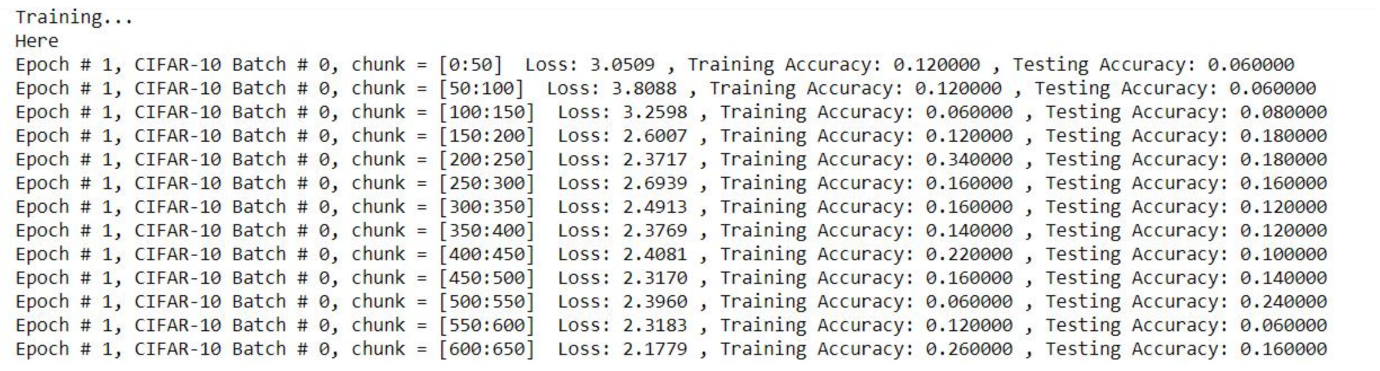


Cost

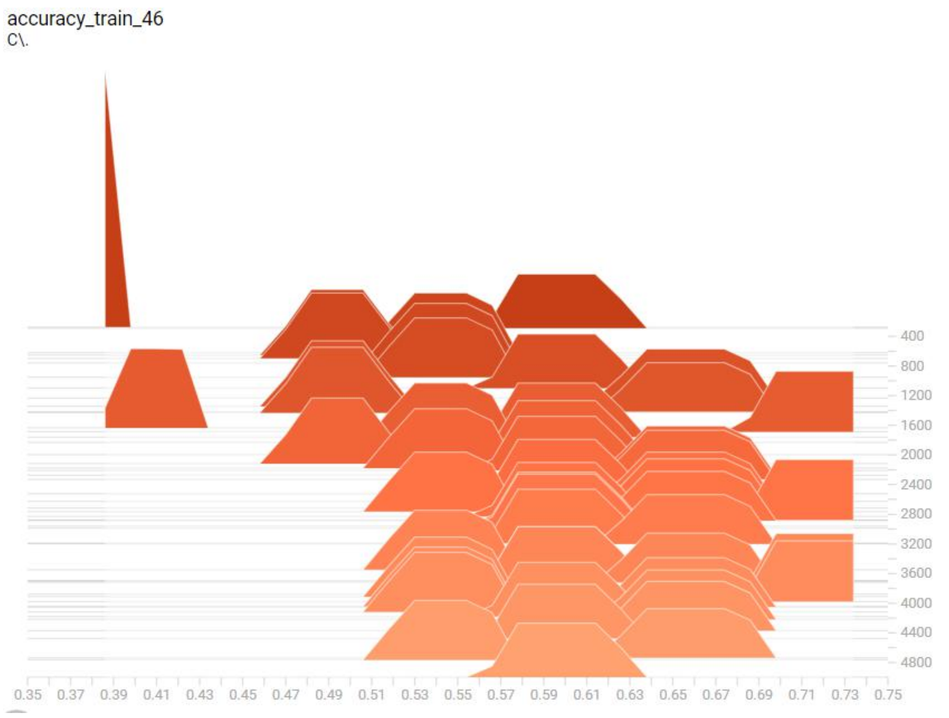


Histograms:

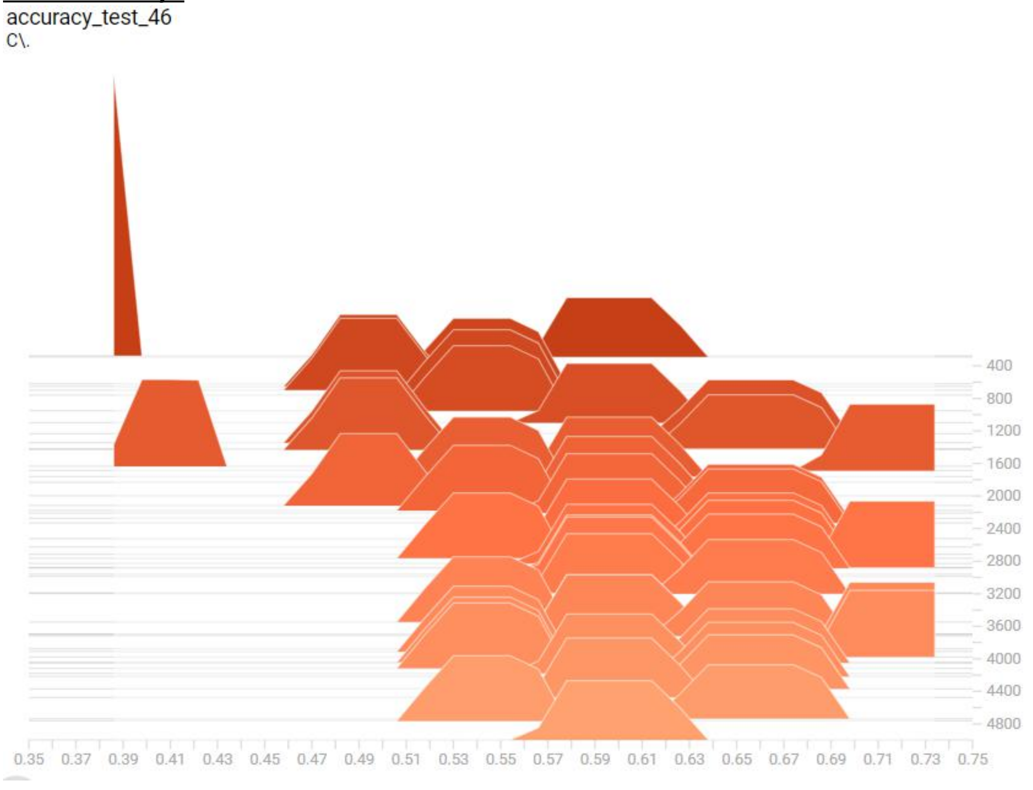
Training Accuracy



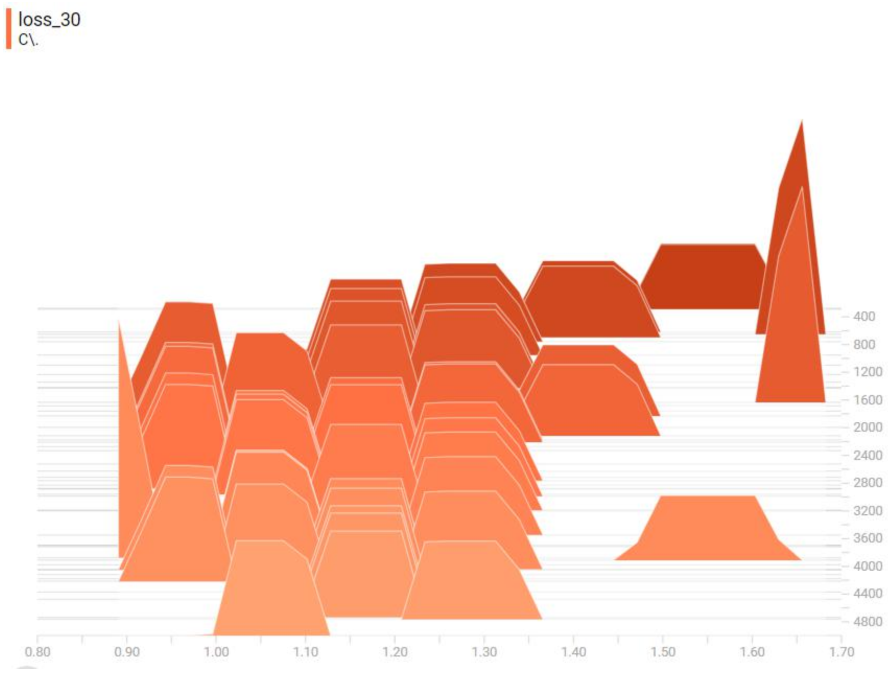
```
Training...
Here
Epoch # 1, CIFAR-10 Batch # 0, chunk = [0:50]  Loss: 3.0509 , Training Accuracy: 0.120000 , Testing Accuracy: 0.060000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [50:100]  Loss: 3.8088 , Training Accuracy: 0.120000 , Testing Accuracy: 0.060000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [100:150]  Loss: 3.2598 , Training Accuracy: 0.060000 , Testing Accuracy: 0.080000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [150:200]  Loss: 2.6007 , Training Accuracy: 0.120000 , Testing Accuracy: 0.180000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [200:250]  Loss: 2.3717 , Training Accuracy: 0.340000 , Testing Accuracy: 0.180000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [250:300]  Loss: 2.6939 , Training Accuracy: 0.160000 , Testing Accuracy: 0.160000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [300:350]  Loss: 2.4913 , Training Accuracy: 0.160000 , Testing Accuracy: 0.120000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [350:400]  Loss: 2.3769 , Training Accuracy: 0.140000 , Testing Accuracy: 0.120000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [400:450]  Loss: 2.4081 , Training Accuracy: 0.220000 , Testing Accuracy: 0.100000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [450:500]  Loss: 2.3170 , Training Accuracy: 0.160000 , Testing Accuracy: 0.140000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [500:550]  Loss: 2.3960 , Training Accuracy: 0.060000 , Testing Accuracy: 0.240000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [550:600]  Loss: 2.3183 , Training Accuracy: 0.120000 , Testing Accuracy: 0.060000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [600:650]  Loss: 2.1779 , Training Accuracy: 0.260000 , Testing Accuracy: 0.160000
```



Test Accuracy



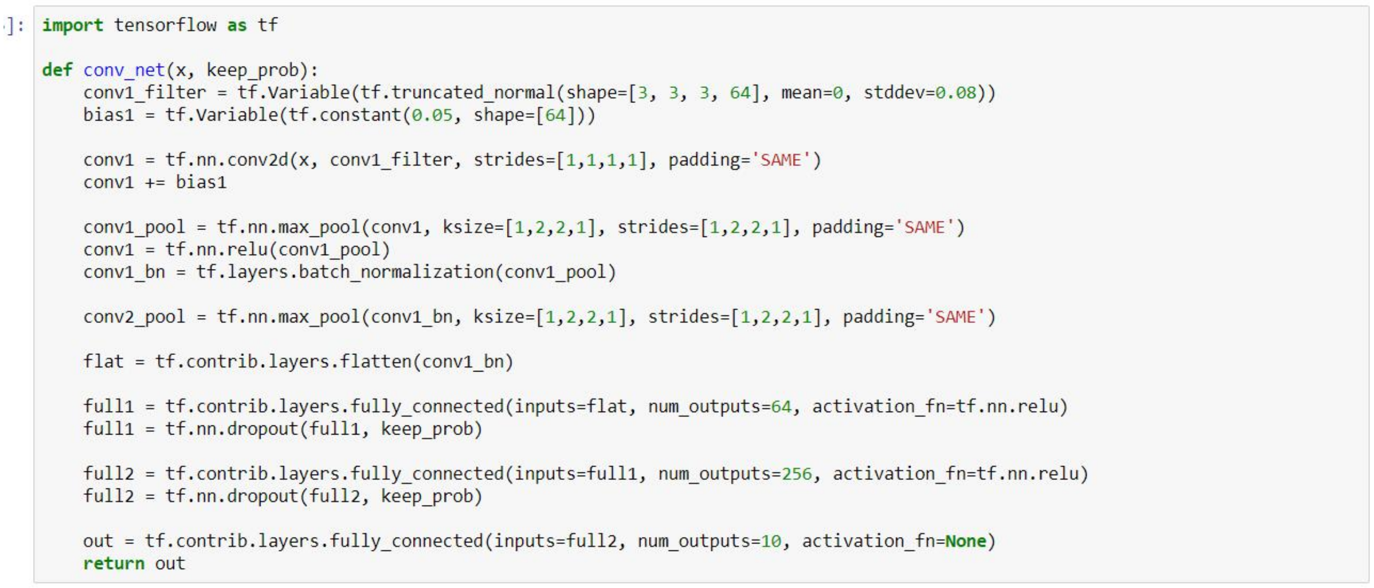
Cost



EXERCISE 2 CNN for Image Classification

In the following cells we will setup our structure of CNN

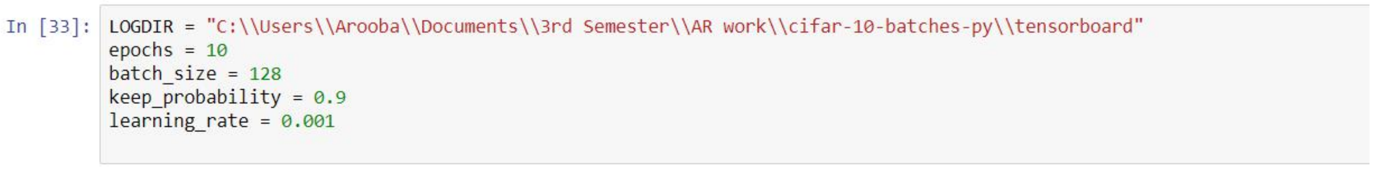
```python
import tensorflow as tf
# Remove previous weights, bias, inputs, etc..
tf.reset_default_graph()

# Inputs
x = tf.placeholder(tf.float32, shape=(None, 32, 32, 3), name='input_x')
y =  tf.placeholder(tf.float32, shape=(None, 10), name='output_y')
keep_prob = tf.placeholder(tf.float32, name='keep_prob')
```



```
]: import tensorflow as tf

   def conv_net(x, keep_prob):
       conv1_filter = tf.Variable(tf.truncated_normal(shape=[3, 3, 3, 64], mean=0, stddev=0.08))
       bias1 = tf.Variable(tf.constant(0.05, shape=[64]))

       conv1 = tf.nn.conv2d(x, conv1_filter, strides=[1,1,1,1], padding='SAME')
       conv1 += bias1

       conv1_pool = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')
       conv1 = tf.nn.relu(conv1_pool)
       conv1_bn = tf.layers.batch_normalization(conv1_pool)

       conv2_pool = tf.nn.max_pool(conv1_bn, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

       flat = tf.contrib.layers.flatten(conv1_bn)

       full1 = tf.contrib.layers.fully_connected(inputs=flat, num_outputs=64, activation_fn=tf.nn.relu)
       full1 = tf.nn.dropout(full1, keep_prob)

       full2 = tf.contrib.layers.fully_connected(inputs=full1, num_outputs=256, activation_fn=tf.nn.relu)
       full2 = tf.nn.dropout(full2, keep_prob)

       out = tf.contrib.layers.fully_connected(inputs=full2, num_outputs=10, activation_fn=None)
       return out
```

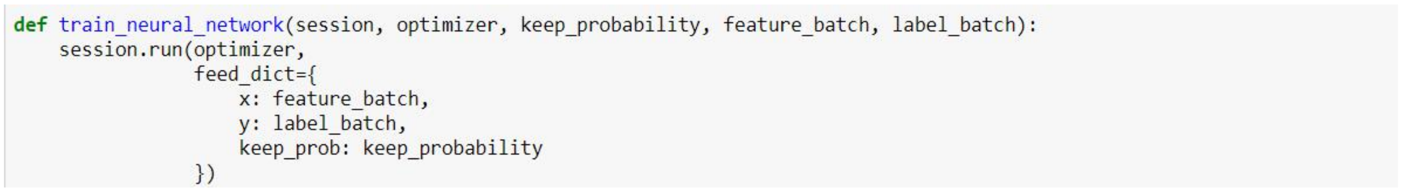
Hyperparameters

- epochs: number of iterations until the network stops learning or start overfitting
- batch_size: highest number that your machine has memory for. Most people set them to common sizes of memory:
- keep_probability: probability of keeping a node using dropout
- learning_rate: number how fast the model learns



```
In [33]: LOGDIR = "C:\\Users\\Arooba\\Documents\\3rd Semester\\AR work\\cifar-10-batches-py\\tensorboard"
         epochs = 10
         batch_size = 128
         keep_probability = 0.9
         learning_rate = 0.001
```

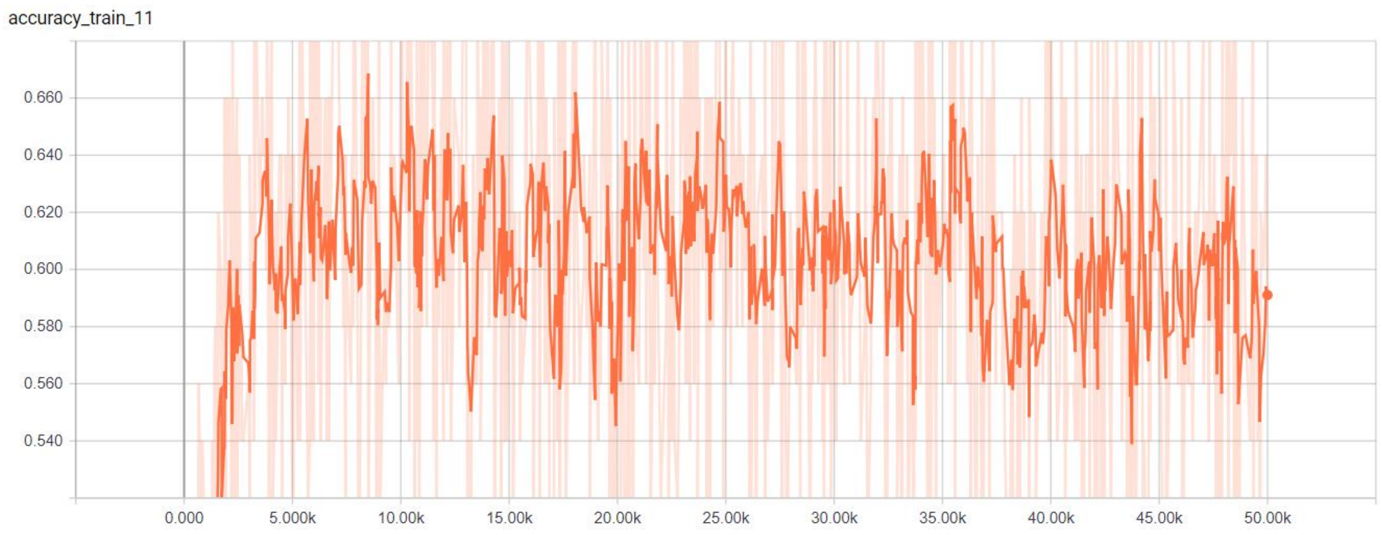
This function performs the learning/optimization using the training data



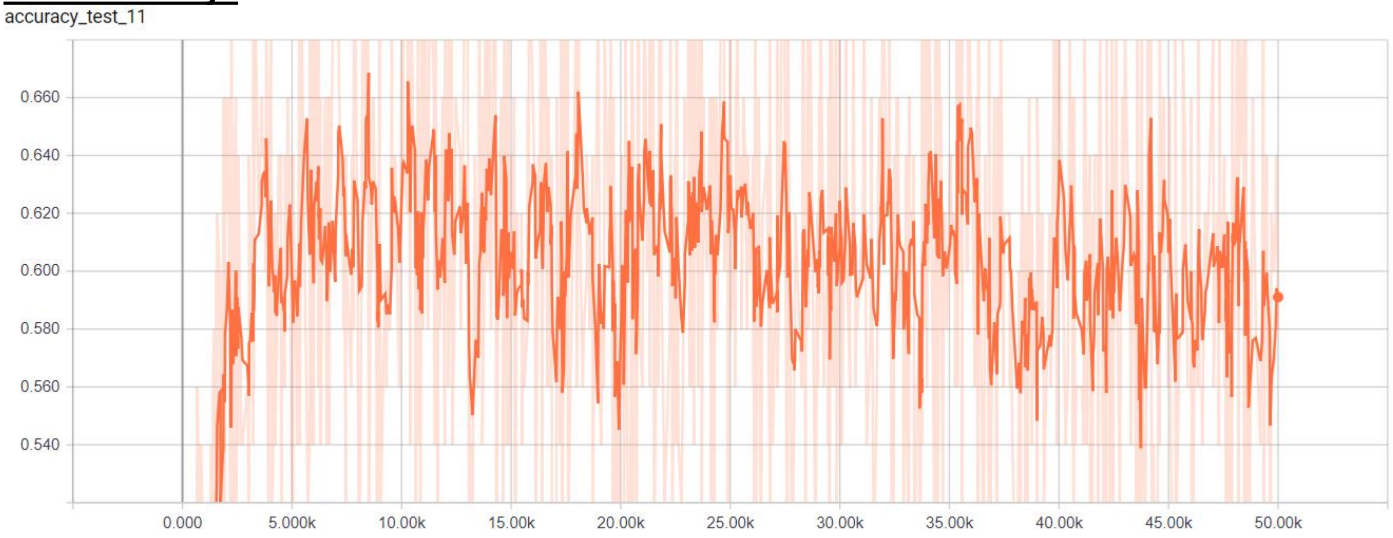
```
def train_neural_network(session, optimizer, keep_probability, feature_batch, label_batch):
    session.run(optimizer,
                feed_dict={
                    x: feature_batch,
                    y: label_batch,
                    keep_prob: keep_probability
                })
```

Scalars:

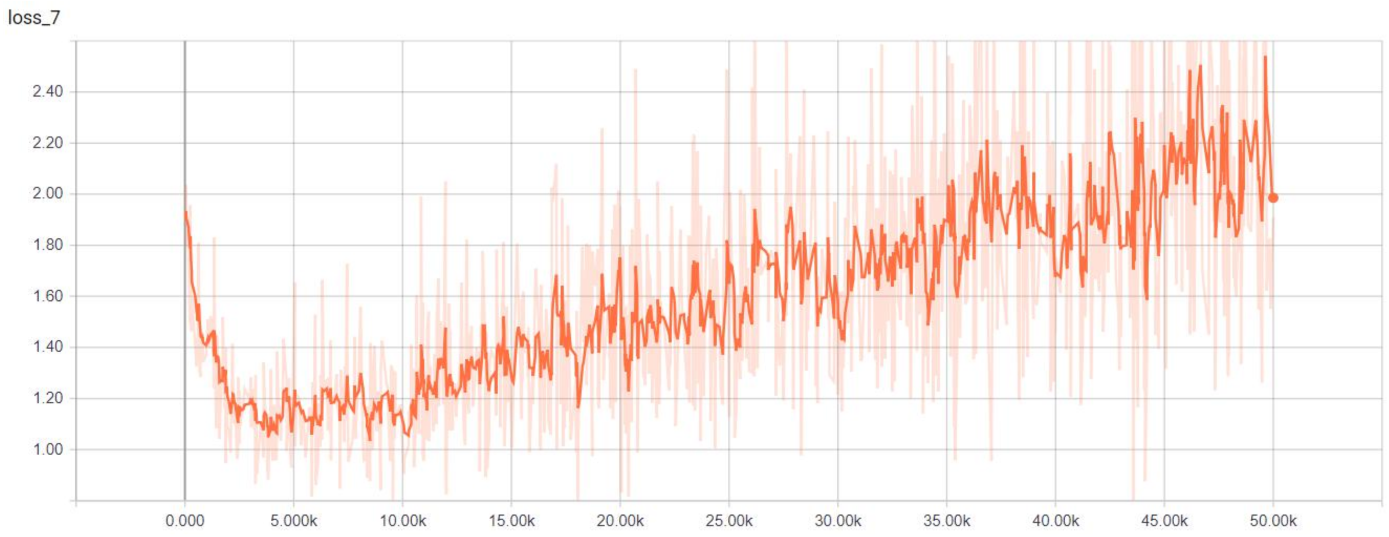
Training Accuracy



Test Accuracy



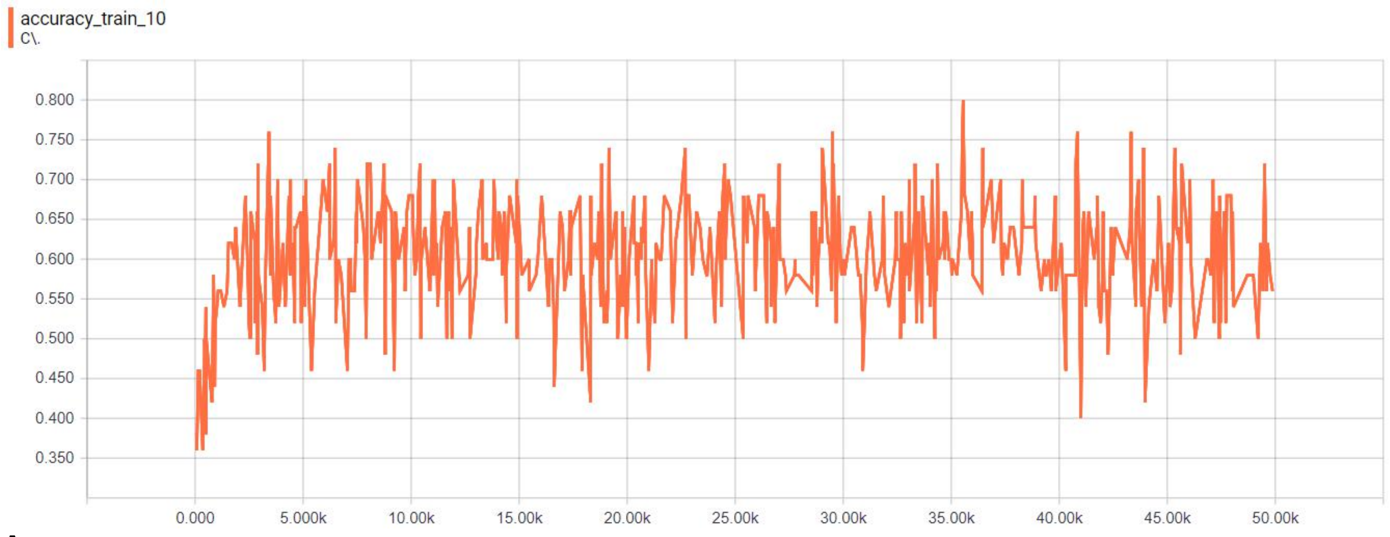
Cost



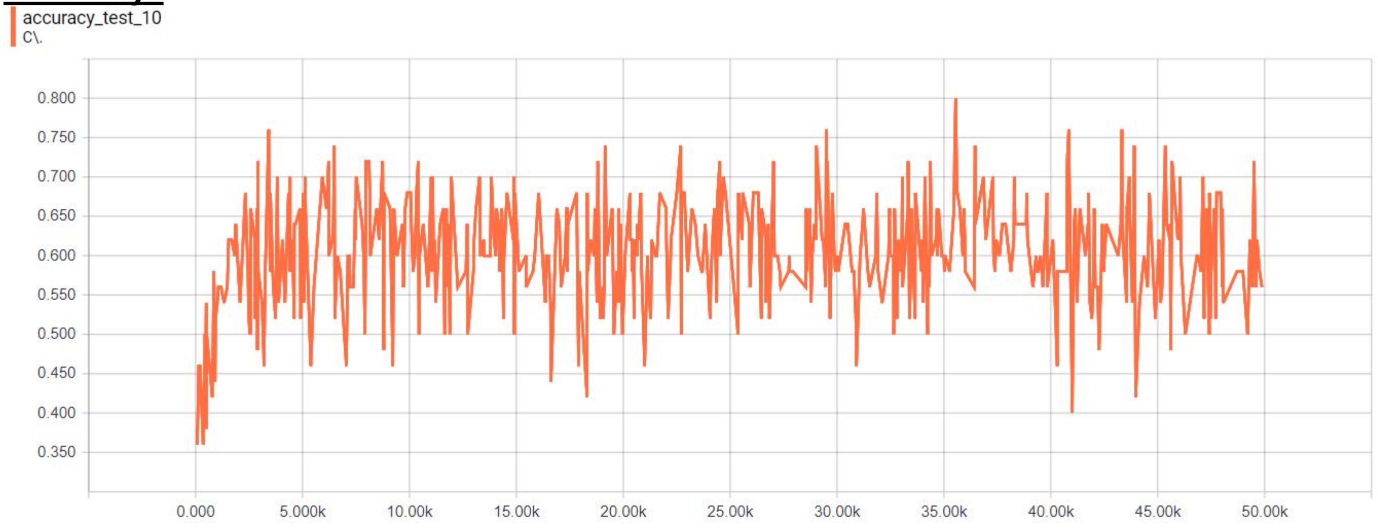
Distributions:

Training Accuracy

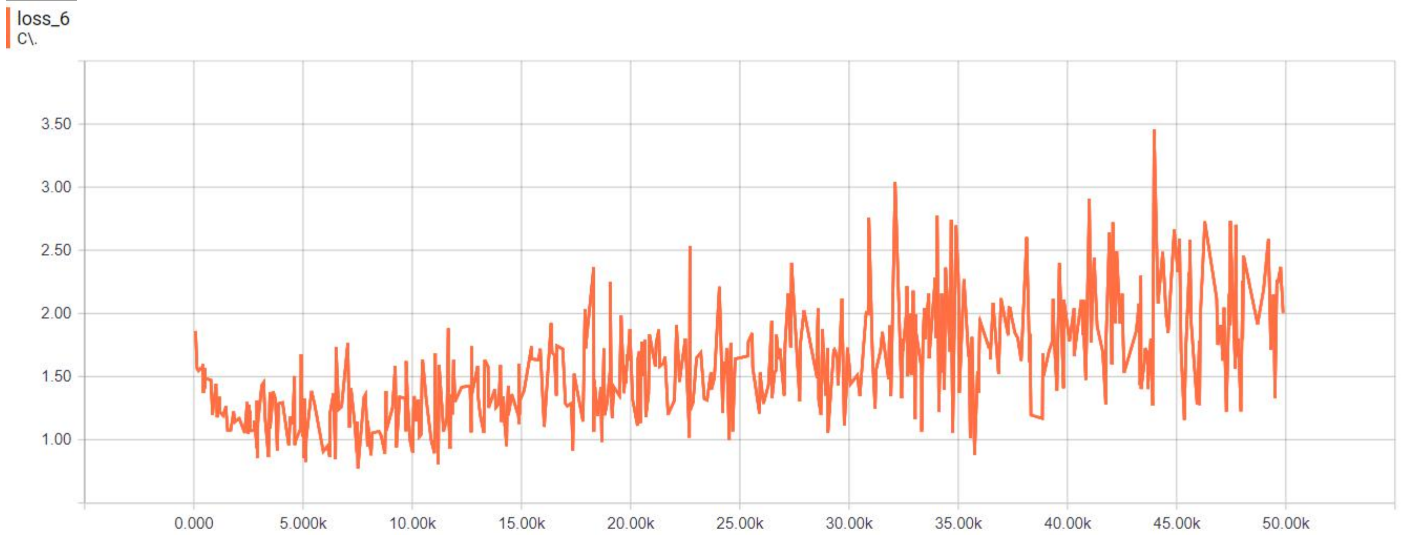
Test



Accuracy

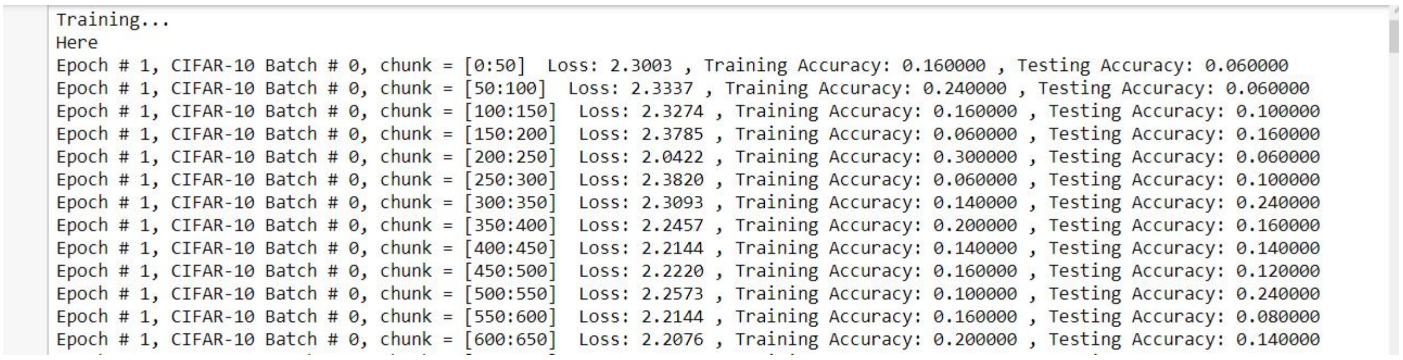


Cost

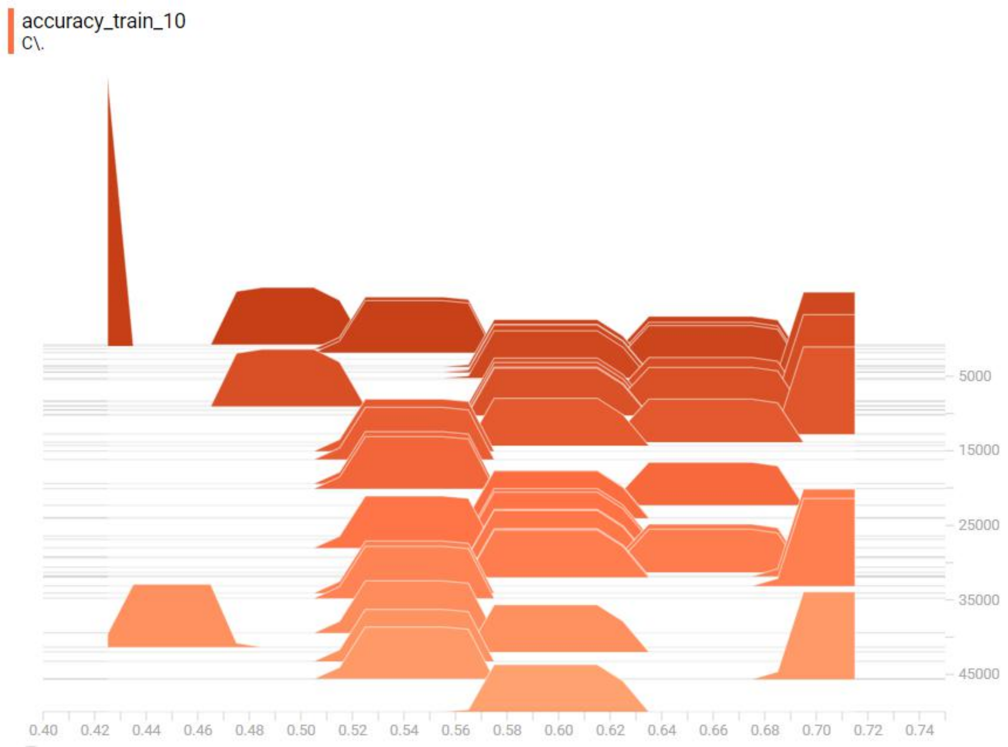


Histogram:

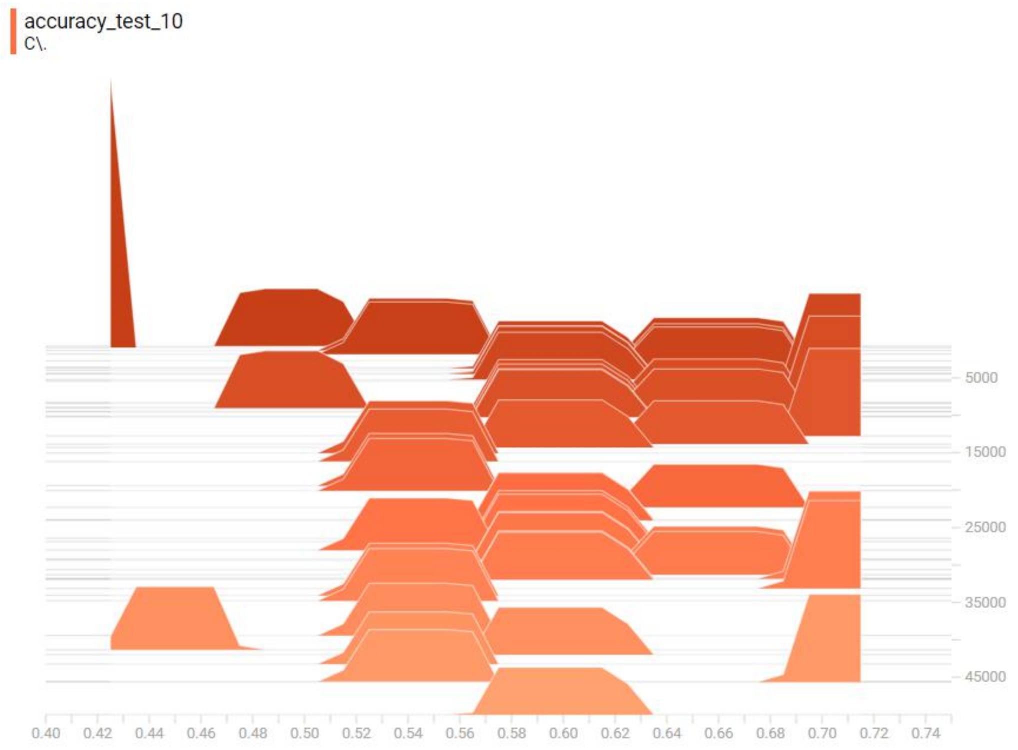
Training Accuracy



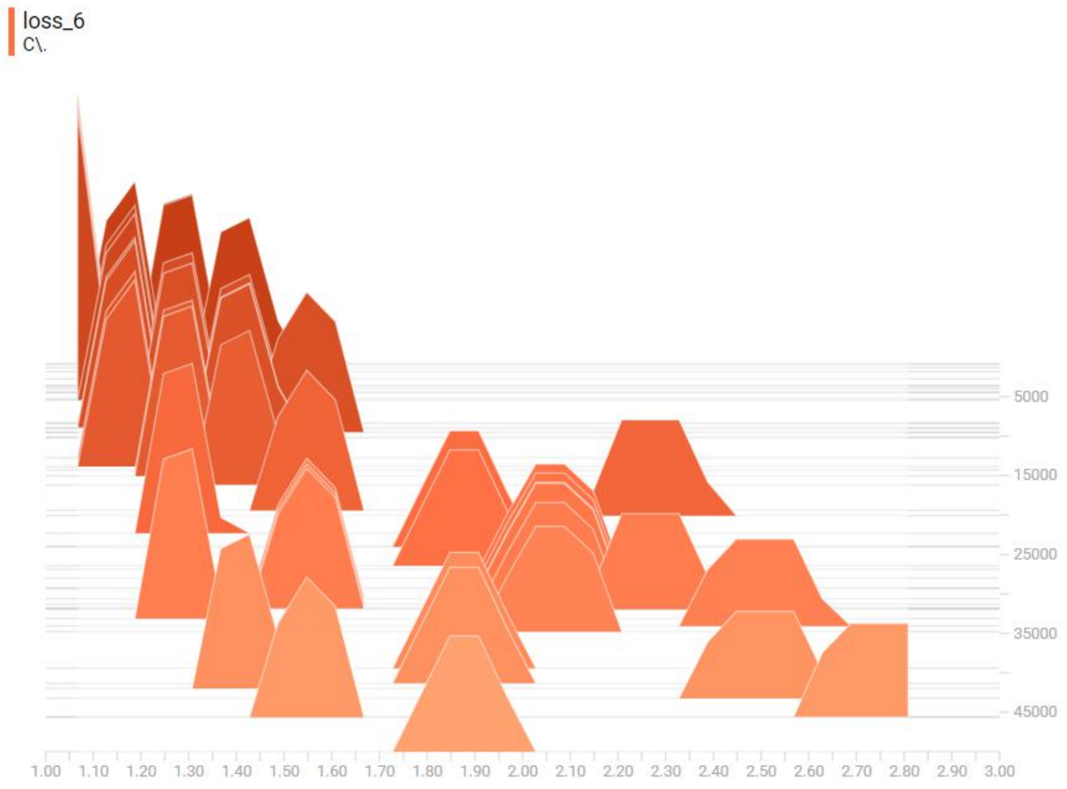
```
Training...
Here
Epoch # 1, CIFAR-10 Batch # 0, chunk = [0:50]  Loss: 2.3003 , Training Accuracy: 0.160000 , Testing Accuracy: 0.060000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [50:100]  Loss: 2.3337 , Training Accuracy: 0.240000 , Testing Accuracy: 0.060000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [100:150]  Loss: 2.3274 , Training Accuracy: 0.160000 , Testing Accuracy: 0.100000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [150:200]  Loss: 2.3785 , Training Accuracy: 0.060000 , Testing Accuracy: 0.160000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [200:250]  Loss: 2.0422 , Training Accuracy: 0.300000 , Testing Accuracy: 0.060000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [250:300]  Loss: 2.3820 , Training Accuracy: 0.060000 , Testing Accuracy: 0.100000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [300:350]  Loss: 2.3093 , Training Accuracy: 0.140000 , Testing Accuracy: 0.240000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [350:400]  Loss: 2.2457 , Training Accuracy: 0.200000 , Testing Accuracy: 0.160000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [400:450]  Loss: 2.2144 , Training Accuracy: 0.140000 , Testing Accuracy: 0.140000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [450:500]  Loss: 2.2220 , Training Accuracy: 0.160000 , Testing Accuracy: 0.120000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [500:550]  Loss: 2.2573 , Training Accuracy: 0.100000 , Testing Accuracy: 0.240000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [550:600]  Loss: 2.2144 , Training Accuracy: 0.160000 , Testing Accuracy: 0.080000
Epoch # 1, CIFAR-10 Batch # 0, chunk = [600:650]  Loss: 2.2076 , Training Accuracy: 0.200000 , Testing Accuracy: 0.140000
```



Test Accuracy


accuracy_test_10
C\.
5000
15000
25000
35000
45000
0.40
0.42
0.44
0.46
0.48
0.50
0.52
0.54
0.56
0.58
0.60
0.62
0.64
0.66
0.68
0.70
0.72
0.74

Cost


loss_6
C\.
5000
15000
25000
35000
45000
1.00
1.10
1.20
1.30
1.40
1.50
1.60
1.70
1.80
1.90
2.00
2.10
2.20
2.30
2.40
2.50
2.60
2.70
2.80
2.90
3.00