Assignment 12
CSCI E 63 - Big Data Analytics

Problem 1:

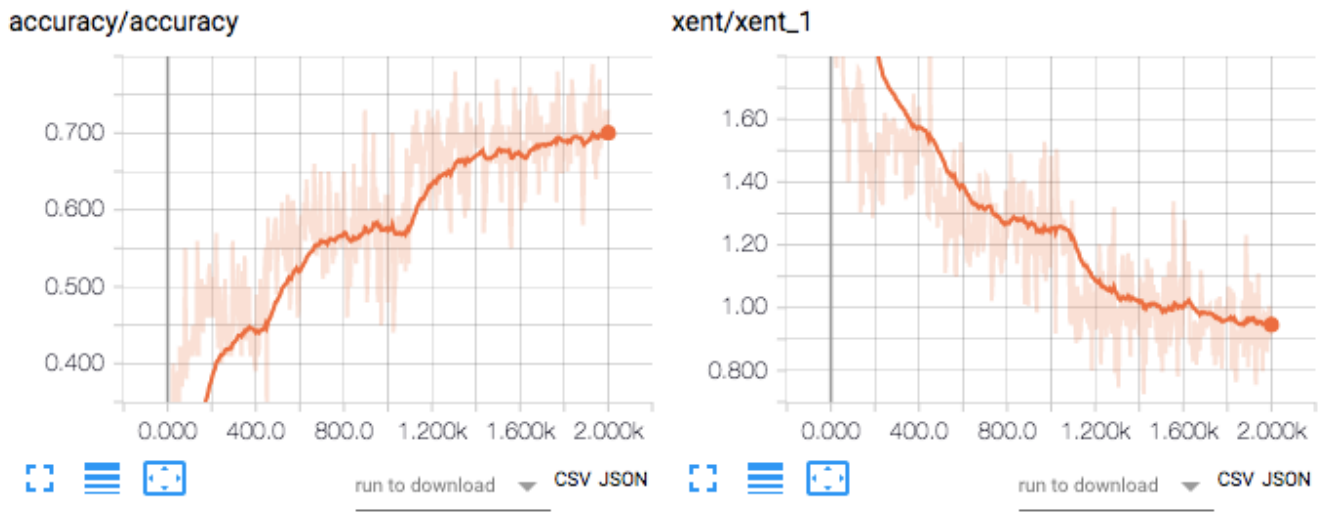
Please find attached 2 files from Google's tutorials sets. I used file `mnist2.py` for preparation of my notes. If you read the file carefully you will see that you can run it in at least two modes. The way it is setup now it selects one learning rate and one particular neural network architecture and generates TensorBoard graph in a particular directory. One problem with this script is that its accuracy is surprisingly low. Such complex architecture and so many lines of code and we get 70% or lower accuracy. We expected more from Convolutional Neural Networks. File `cnn_mnist.py` is practically the same, at least it does all the same things, creates similar architecture, sets the same or similar parameters, but does a much better job. Its accuracy is in high 90%-s. Run two files, compare results and then fix the first file (`mnist.py`) based on what you saw in file `cnn_mnist.py`. Capture the Accuracy and Cross Entropy (summary) graphs from the corrected version of `mnist2.py` and provide working and fixed version of that file. Please describe in detail experiments you undertook and fixes you made.

Answer:

→ Run mnist2.py

```
(tensorflow) karanbhandarkar@Karans-MacBook-Air:~/Projects/PythonProjects/CSCI E 63 Big Data Anlytics/Assignment 12$ python mnist2.py
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting log3/data/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting log3/data/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting log3/data/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting log3/data/t10k-labels-idx1-ubyte.gz
Starting run for lr_1E-04conv2fc2
2017-12-01 23:47:53.144681: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to us
e: SSE4.1 SSE4.2 AVX AVX2 FMA
```

Capture the accuracy and entropy summary graphs(after smoothening)



Assignment 12
CSCI E 63 - Big Data Analytics

→ Run cnn_mnist.py

```
(tensorflow) karanbhandarkar@Karans-MacBook-Air:~/Projects/PythonProjects/CSCI E 63 Big Data Anlytics/Assignment 12$ python cnn_mnist.py
2017-12-01 21:14:14.864359: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not c
ompiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting temp/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting temp/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting temp/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting temp/t10k-labels-idx1-ubyte.gz
Generation # 5. Train Loss: 2.36. Train Acc (Test Acc): 8.00 (9.80)
Generation # 10. Train Loss: 2.21. Train Acc (Test Acc): 23.00 (19.20)
Generation # 15. Train Loss: 2.10. Train Acc (Test Acc): 22.00 (20.40)
Generation # 20. Train Loss: 1.98. Train Acc (Test Acc): 46.00 (42.80)
Generation # 25. Train Loss: 1.90. Train Acc (Test Acc): 58.00 (58.80)
Generation # 30. Train Loss: 1.65. Train Acc (Test Acc): 69.00 (68.20)
Generation # 35. Train Loss: 1.46. Train Acc (Test Acc): 61.00 (72.20)
Generation # 40. Train Loss: 1.16. Train Acc (Test Acc): 72.00 (72.80)
Generation # 45. Train Loss: 0.73. Train Acc (Test Acc): 83.00 (76.20)
Generation # 50. Train Loss: 0.54. Train Acc (Test Acc): 85.00 (82.80)
Generation # 55. Train Loss: 0.62. Train Acc (Test Acc): 78.00 (84.20)
Generation # 60. Train Loss: 0.56. Train Acc (Test Acc): 84.00 (88.20)
Generation # 65. Train Loss: 0.64. Train Acc (Test Acc): 77.00 (87.40)
Generation # 70. Train Loss: 0.60. Train Acc (Test Acc): 89.00 (89.20)
Generation # 75. Train Loss: 0.46. Train Acc (Test Acc): 82.00 (86.80)
Generation # 80. Train Loss: 0.36. Train Acc (Test Acc): 89.00 (88.80)
Generation # 85. Train Loss: 0.32. Train Acc (Test Acc): 89.00 (88.00)
```

```
Generation # 460. Train Loss: 0.08. Train Acc (Test Acc): 97.00 (95.60)
Generation # 465. Train Loss: 0.17. Train Acc (Test Acc): 92.00 (97.20)
Generation # 470. Train Loss: 0.17. Train Acc (Test Acc): 94.00 (96.00)
Generation # 475. Train Loss: 0.19. Train Acc (Test Acc): 92.00 (96.00)
Generation # 480. Train Loss: 0.19. Train Acc (Test Acc): 94.00 (96.80)
Generation # 485. Train Loss: 0.21. Train Acc (Test Acc): 92.00 (96.40)
Generation # 490. Train Loss: 0.13. Train Acc (Test Acc): 93.00 (95.80)
Generation # 495. Train Loss: 0.17. Train Acc (Test Acc): 93.00 (94.80)
Generation # 500. Train Loss: 0.07. Train Acc (Test Acc): 99.00 (95.40)
```

→ Compare the results

mnist2.py shows poor results of 70% accuracy as compared to 99% train and 95.4% test accuracy of cnn_mnist.py

NOTE:
To print results like in cnn_mnist.py while running mnist2.py, add following lines:

```
if(i+1) % 100 == 0:
      print('Generation # {}. Train Acc: {:.2f}'.format(i, train_accuracy * 100))
```

Karan A. Bhandarkar

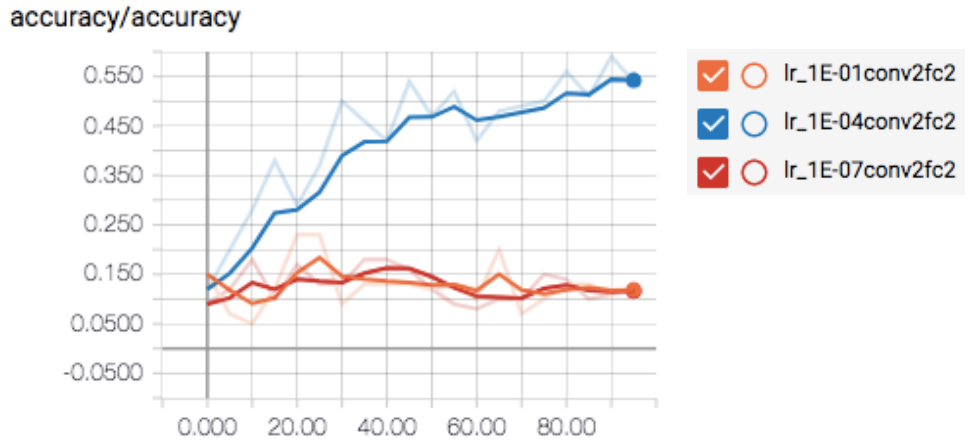
Assignment 12
CSCI E 63 - Big Data Analytics

→ Please describe in detail experiments you undertook and fixes you made.

NOTE: To run the experiments faster, I changed the epoch to 100.

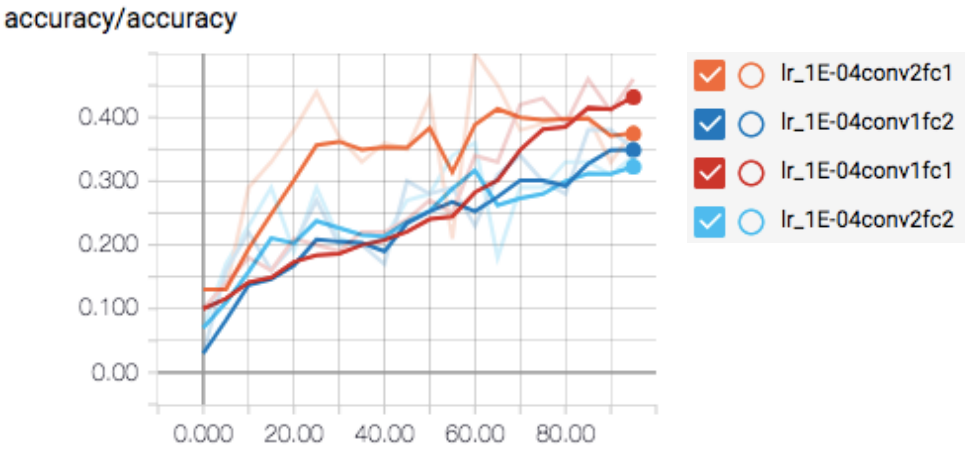
1. Change the learning rates and see the effect

Replace line `for learning_rate in [1E-4]` with `for learning_rate in [1E-1, 1E-4, 1E-7]`



Conclusion: Changing the learning rate seems to have made it worse. I reverted back to 1E-4.

2. Change the number of fully connected and convoluted layers



Conclusion: This didn't really seem to help either.

Karan A. Bhandarkar

Assignment 12
CSCI E 63 - Big Data Analytics

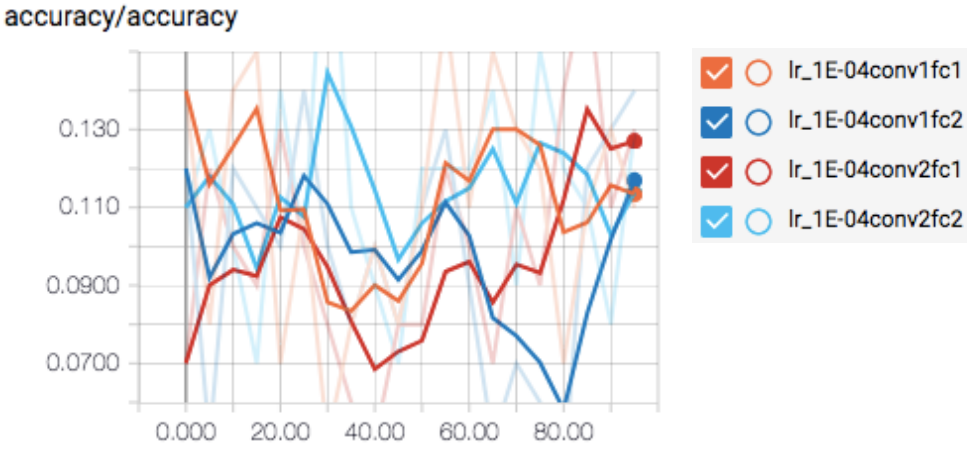
3. Change the weights and bias along with the layers

From:

```
w = tf.Variable(tf.truncated_normal([5, 5, size_in, size_out], stddev=0.1),
name="W")
b = tf.Variable(tf.constant(0.1, shape=[size_out]), name="B")
```

To:

```
w = tf.Variable(tf.zeros([5, 5, size_in, size_out]), name="W")
b = tf.Variable(tf.zeros([size_out]), name="B")
```



Conclusion: Not a good idea at all.

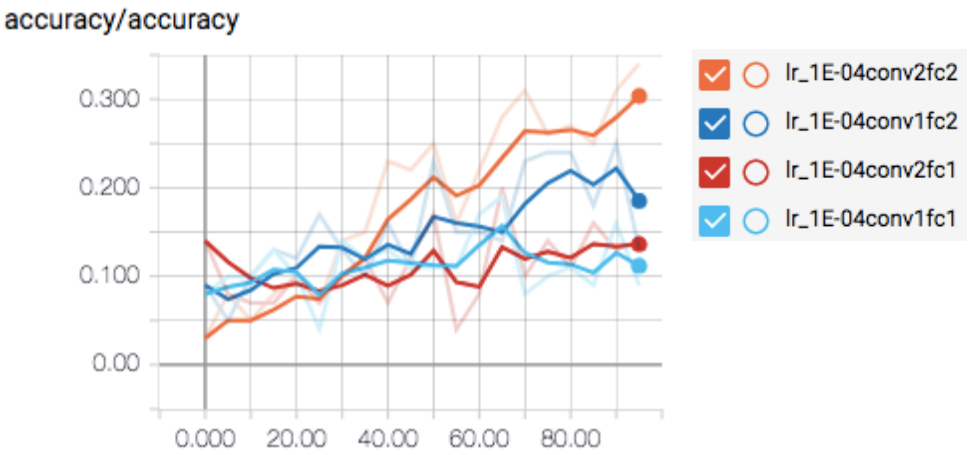
4. Change the Optimizer from Adam to Momentum

From:

```
train_step = tf.train.AdamOptimizer(learning_rate).minimize(xent)
```

To:

```
train_step = tf.train.MomentumOptimizer(learning_rate, 0.9).minimize(xent)
```



Conclusion: This does not seem to have helped

Karan A. Bhandarkar

Assignment 12
CSCI E 63 - Big Data Analytics

5. Change the Activation function for two Fully Connected layers

From:

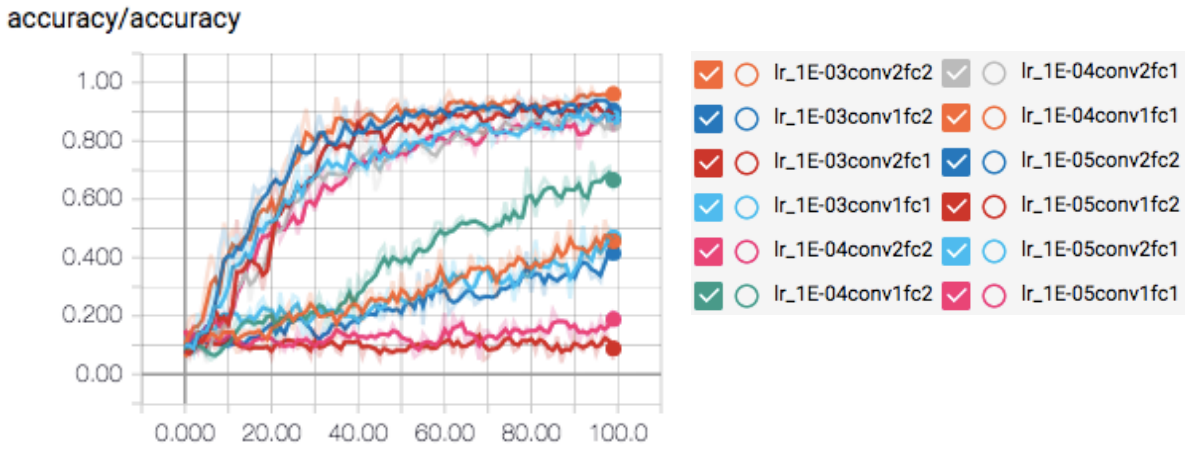
```
act = tf.nn.relu(tf.matmul(input, w) + b)
```

To:

```
act = tf.nn.relu(tf.matmul(input, w) + b) if hidden_layer else (tf.matmul(input, w)
+ b)
```

Add the hidden layer indication as:

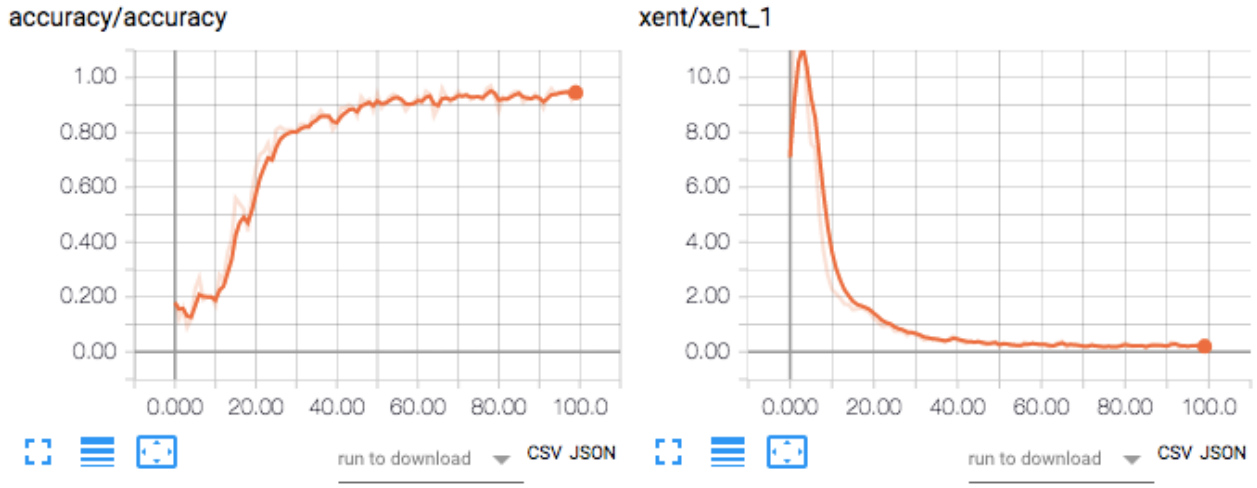
```
if use_two_fc:
    fc1 = fc_layer(flattened, 7 * 7 * 64, 1024, "fc1", hidden_layer = True)
```



Conclusion: It worked!

→ Clean up the conditions in the code for a specific run (Fixed cope submitted separately)

```
(tensorflow) karanbhandarkar@Karans-MacBook-Air:~/Projects/PythonProjects/CSCI E 63 Big Data Anlytics/Assignment 12$ python mnist2_fixed.py
Extracting log3/data/train-images-idx3-ubyte.gz
Extracting log3/data/train-labels-idx1-ubyte.gz
Extracting log3/data/t10k-images-idx3-ubyte.gz
Extracting log3/data/t10k-labels-idx1-ubyte.gz
Starting run for lr_1E-03conv2fc2
2017-12-02 03:01:02.713899: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary
was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA
Generation # 99. Train Acc: 97.00
```



Karan A. Bhandarkar

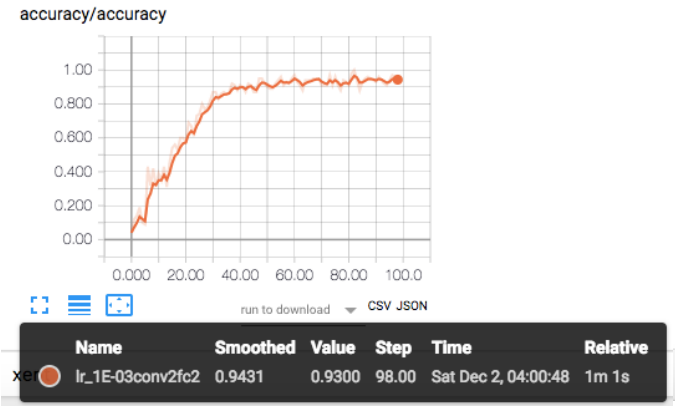
Assignment 12
CSCI E 63 - Big Data Analytics

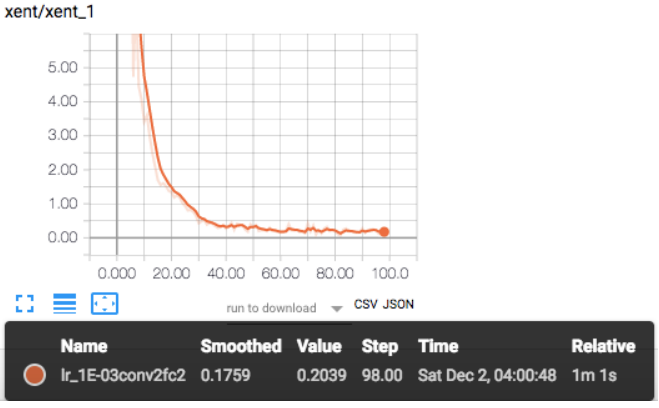
Problem 2:
Run corrected version of mnist2.py for 4 different architectures (2 conv, 1 conv, 2 fully connected, 1 fully connected layer) and 3 values of the learning rate. As one learning rate choose the one you selected in Problem 1 and then add one smaller and one larger learning rate around that one. Capture Accuracy (summary) graphs and One of Histograms to demonstrate to us that your code is working. Please be aware that you are running 12 models and the execution might take many minutes. You might want to run your models in smaller groups so that you see them finish their work without too much wait. Submit working code of `mnist2.py` used in this problem. Collect execution times, final (smoothed) accuracies and final cross entropies for different models and provide tabulated presentation of the final results of different models

Answer:

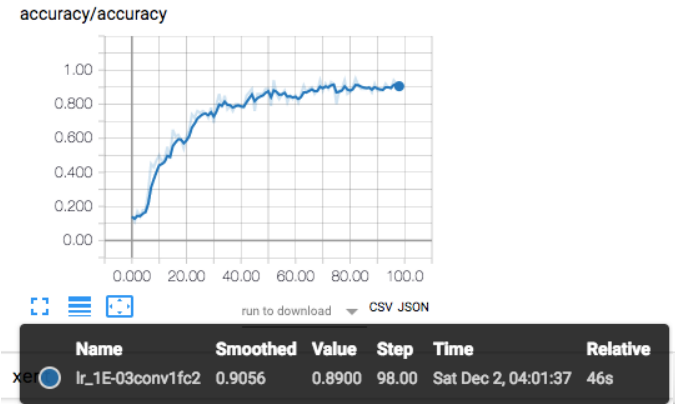
→

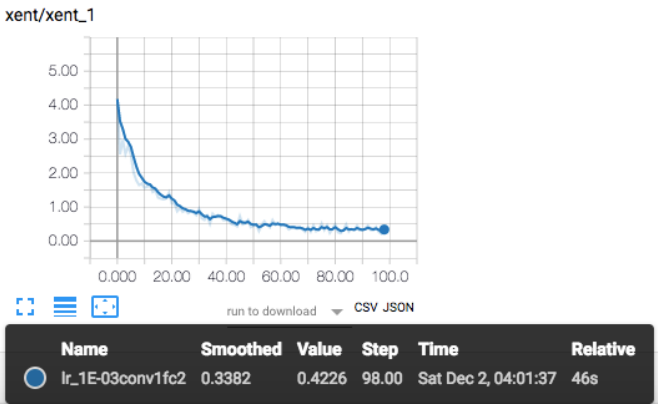
Learning Rate	CONV	FC
0.03	2	2





Learning Rate	CONV	FC
0.03	1	2



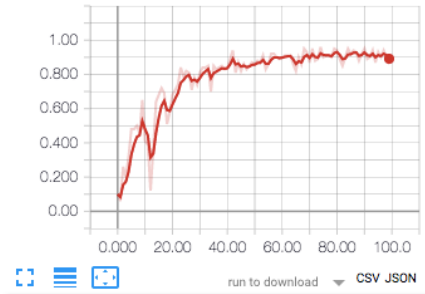


Karan A. Bhandarkar

Assignment 12
CSCI E 63 - Big Data Analytics

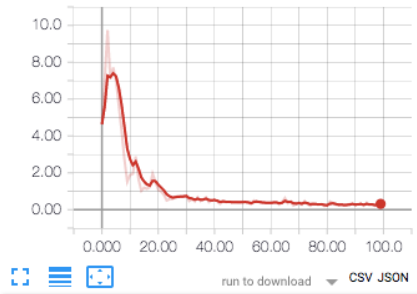
Learning Rate	CONV	FC
0.03	2	1

accuracy/accuracy



Name	Smoothed	Value	Step	Time	Relative
lr_1E-03conv2fc1	0.8913	0.8600	99.00	Sat Dec 2, 04:02:20	41s

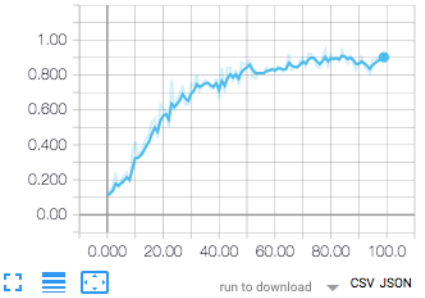
xent/xent_1



Name	Smoothed	Value	Step	Time	Relative
lr_1E-03conv2fc1	0.3141	0.3590	99.00	Sat Dec 2, 04:02:20	41s

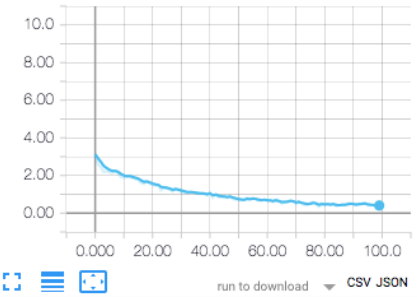
Learning Rate	CONV	FC
0.03	1	1

accuracy/accuracy



Name	Smoothed	Value	Step	Time	Relative
lr_1E-03conv1fc1	0.9016	0.9200	99.00	Sat Dec 2, 04:02:53	31s

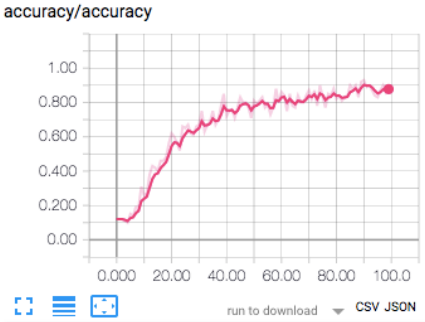
xent/xent_1



Name	Smoothed	Value	Step	Time	Relative
lr_1E-03conv1fc1	0.4043	0.3720	99.00	Sat Dec 2, 04:02:53	31s

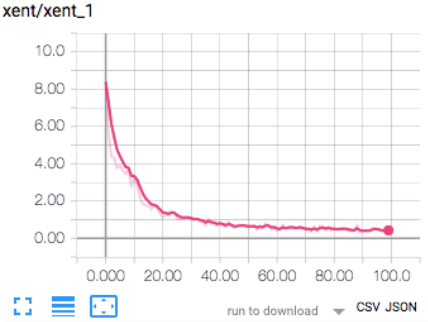
Learning Rate	CONV	FC
0.04	2	2

accuracy/accuracy



Name	Smoothed	Value	Step	Time	Relative
lr_1E-04conv2fc2	0.8777	0.8900	99.00	Sat Dec 2, 04:03:56	1m 1s

xent/xent_1

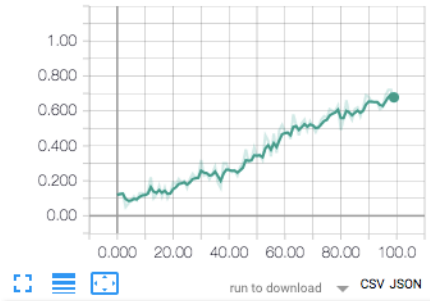


Name	Smoothed	Value	Step	Time	Relative
lr_1E-04conv2fc2	0.4268	0.4387	99.00	Sat Dec 2, 04:03:56	1m 1s

Assignment 12
CSCI E 63 - Big Data Analytics

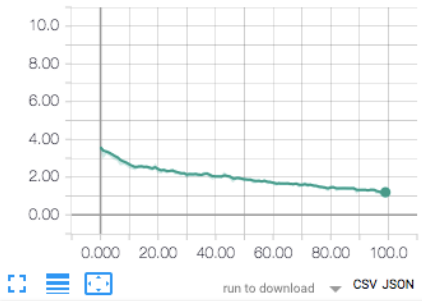
Learning Rate	CONV	FC
0.04	1	2

accuracy/accuracy



Name	Smoothed	Value	Step	Time	Relative
lr_1E-04conv1fc2	0.6776	0.6500	99.00	Sat Dec 2, 04:04:48	50s

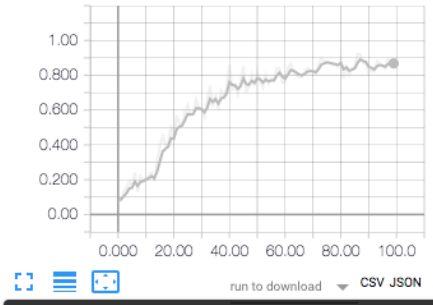
xent/xent_1



Name	Smoothed	Value	Step	Time	Relative
lr_1E-04conv1fc2	1.197	1.198	99.00	Sat Dec 2, 04:04:48	50s

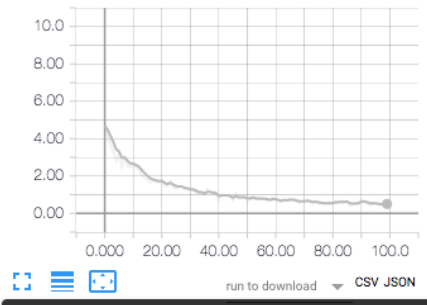
Learning Rate	CONV	FC
0.04	2	1

accuracy/accuracy



Name	Smoothed	Value	Step	Time	Relative
lr_1E-04conv2fc1	0.8684	0.8700	99.00	Sat Dec 2, 04:05:34	44s

xent/xent_1



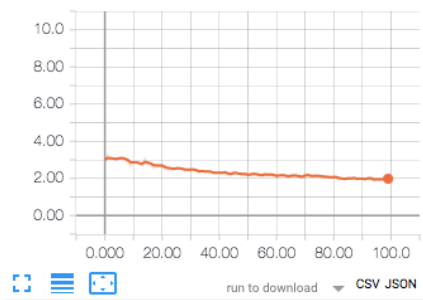
Name	Smoothed	Value	Step	Time	Relative
lr_1E-04conv2fc1	0.5044	0.5226	99.00	Sat Dec 2, 04:05:34	44s

Learning Rate	CONV	FC
0.04	1	1

accuracy/accuracy

Name	Smoothed	Value	Step	Time	Relative
lr_1E-04conv1fc1	0.3052	0.2600	99.00	Sat Dec 2, 04:06:11	35s

xent/xent_1

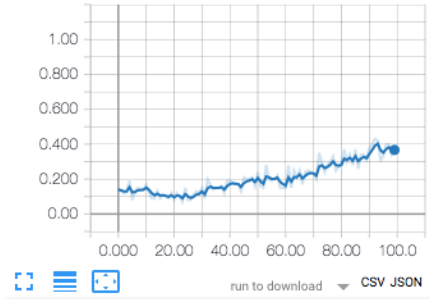


Name	Smoothed	Value	Step	Time	Relative
lr_1E-04conv1fc1	1.982	2.023	99.00	Sat Dec 2, 04:06:11	35s

Assignment 12
CSCI E 63 - Big Data Analytics

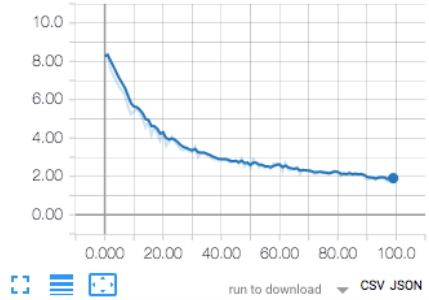
Learning Rate	CONV	FC
0.05	2	2

accuracy/accuracy



Name	Smoothed	Value	Step	Time	Relative
lr_1E-05conv2fc2	0.3673	0.3700	99.00	Sat Dec 2, 04:07:24	1m 11s

xent/xent_1



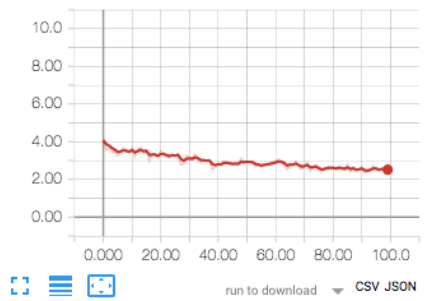
Name	Smoothed	Value	Step	Time	Relative
lr_1E-05conv2fc2	1.908	1.916	99.00	Sat Dec 2, 04:07:24	1m 11s

Learning Rate	CONV	FC
0.05	1	2

accuracy/accuracy

Name	Smoothed	Value	Step	Time	Relative
lr_1E-05conv1fc2	0.1070	0.1100	99.00	Sat Dec 2, 04:08:21	54s

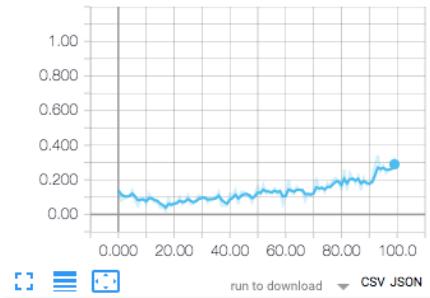
xent/xent_1



Name	Smoothed	Value	Step	Time	Relative
lr_1E-05conv1fc2	2.515	2.448	99.00	Sat Dec 2, 04:08:21	54s

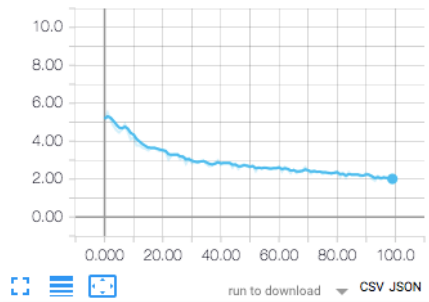
Learning Rate	CONV	FC
0.05	2	1

accuracy/accuracy



Name	Smoothed	Value	Step	Time	Relative
lr_1E-05conv2fc1	0.2899	0.3300	99.00	Sat Dec 2, 04:09:11	47s

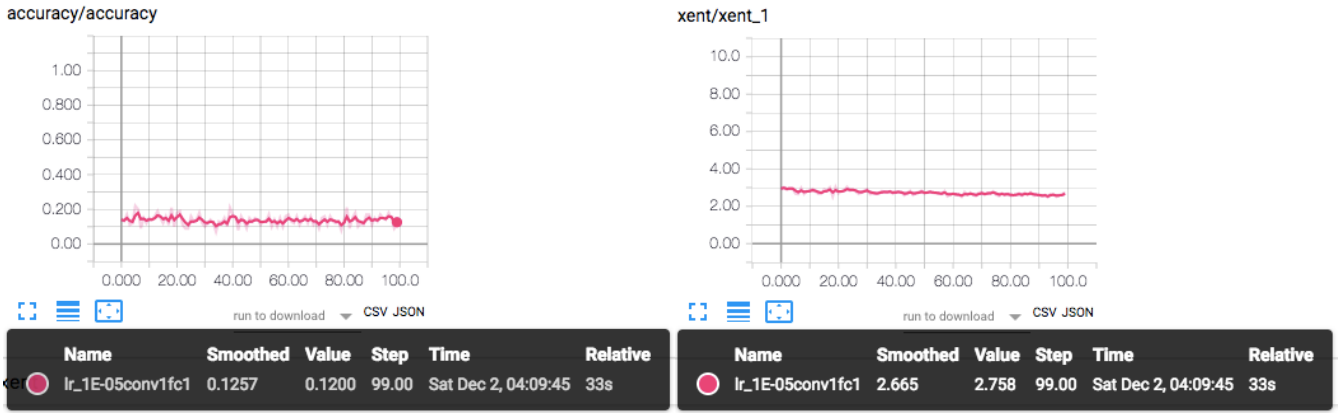
xent/xent_1



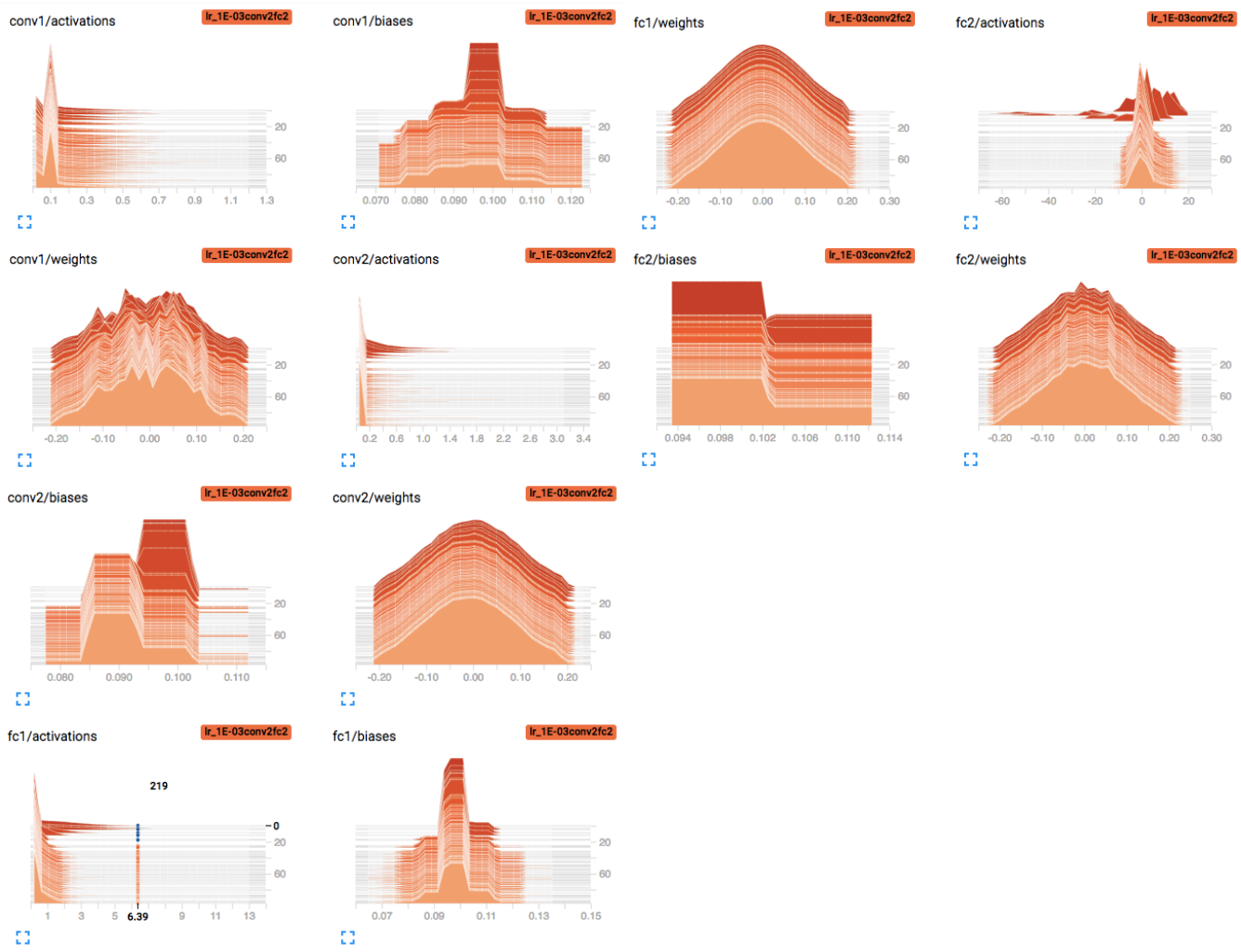
Name	Smoothed	Value	Step	Time	Relative
lr_1E-05conv2fc1	2.013	1.990	99.00	Sat Dec 2, 04:09:11	47s

Assignment 12
CSCI E 63 - Big Data Analytics

Learning Rate	CONV	FC
0.05	1	1



→ Capture one of the histograms to demonstrate working code



Karan A. Bhandarkar

Assignment 12
CSCI E 63 - Big Data Analytics

Problem 3:
Modify file `cnn_mnist.py` so that it publishes its summaries to the TensorBoard. Describe changes you are making and provide images of Accuracy and Cross Entropy summaries as captured by the Tensor Board. Provide the Graph of your model. Describe the differences if any between the graph of this program and the graph generated by `mnist2.py` script running with 2 convolutional and 2 fully connected layers. Provide working code.

Answer:

→ Modify file `cnn_mnist.py` so that it publishes its summaries to the TensorBoard. Describe changes you are making.

1. Add lines to create a writer for the graph

```
…
# Create an optimizer
my_optimizer = tf.train.MomentumOptimizer(learning_rate, 0.9)
train_step = my_optimizer.minimize(loss)

summ = tf.summary.merge_all()

# Initialize Variables
init = tf.global_variables_initializer()
sess.run(init)
writer = tf.summary.FileWriter(data_dir)
writer.add_graph(sess.graph)
…
```

2. Add lines to calculate accuracy using tensorflow and add summary scalars for accuracy and loss

```
…
# Declare Loss Function (softmax cross entropy)
loss =
tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits=model_output,
labels=y_target))
#write loss
tf.summary.scalar("loss", loss)
…
…
# Declare accuracy function
correct_prediction = tf.equal(tf.cast(tf.argmax(model_output, 1), tf.float32),
tf.cast(y_target, tf.float32))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
#write accuracy
tf.summary.scalar("accuracy", accuracy)
…
```

Karan A. Bhandarkar

Assignment 12
CSCI E 63 - Big Data Analytics

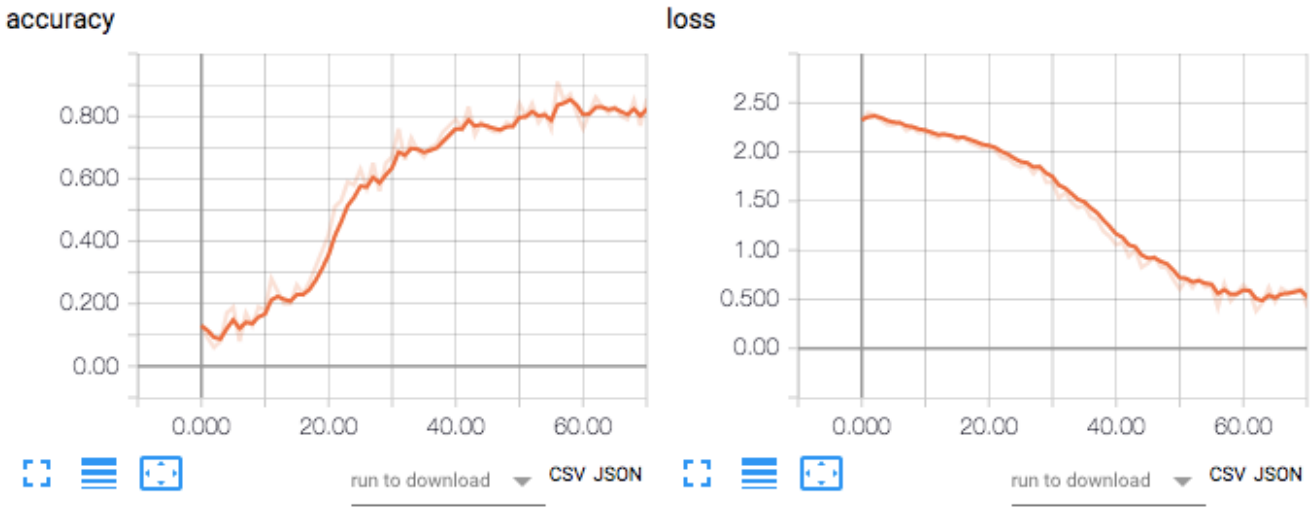
3. Change sess.run to call acuracy and write it to the graph

```
...
# temp_train_loss, temp_train_preds = sess.run([loss, prediction],
feed_dict=train_dict)
# temp_train_acc = get_accuracy(temp_train_preds, rand_y)

temp_train_acc, temp_train_loss, temp_train_preds, s = sess.run([accuracy, loss,
prediction, summ], feed_dict=train_dict)
    writer.add_summary(s, i)
...
```

→ Provide images of Accuracy and Cross Entropy summaries as captured by the Tensor Board.

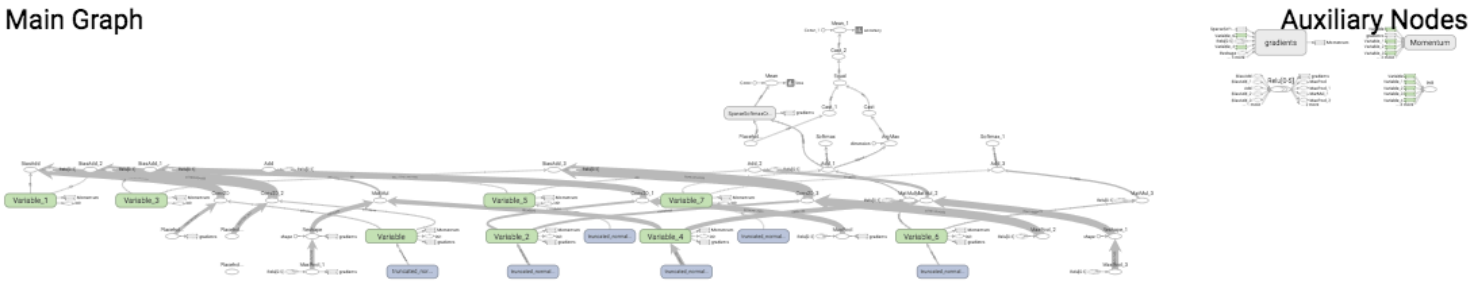
```
(tensorflow) karanbhandarkar@Karans-MacBook-Air:~/Projects/PythonProjects/CSCI E 63 Big Data Anlytics/Assignment 12$ tensorboard
--logdir temp
TensorBoard 0.4.0rc2 at http://Karans-MacBook-Air.local:6006 (Press CTRL+C to quit)
```



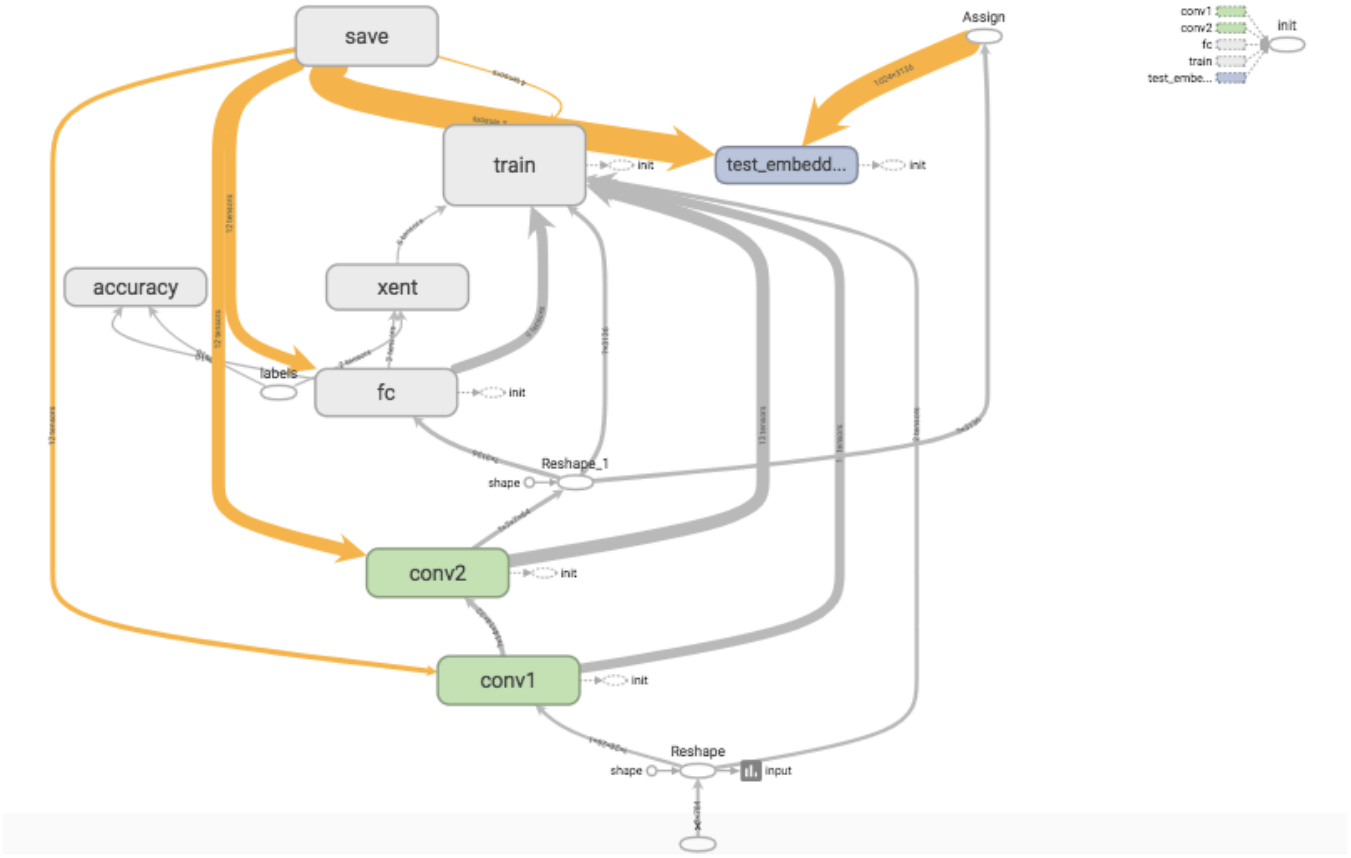
Karan A. Bhandarkar

Assignment 12
CSCI E 63 - Big Data Analytics

→ Provide the Graph of your model.



→ Graph generated by `mnist2.py` script



→ Differences

The two graphs show how the architecture is very different from each other.
The variations can be seen in the convolution layer, fully connected layer and also the pooling.

Karan A. Bhandarkar