## HU Extension Assignment 11 E63 Big Data Analytics

### Handed out: 04/15/2017 Due by 11:59AM EST on Saturday, 04/22/2017

You are welcome to implement TensorFlow problems in this problem set in any of supported languages.

**Problem 1.** Please find 2 files from Google’s tutorials sets. I used file mnist2.py in class yesterday and 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 the same architecture, sets the same or similar parameters, but does much better job. Its accuracy is in high 90%-s. Run two files compare results and then fix the first file (mnist2.py) based on what you saw in file cnn\_mnist.py.

These are the TensorBoard results for accuracy summaries. As mentioned in the problem, accuracy for the mnist2.py (orange) is lower compared to cn-mnist.py (cyan).

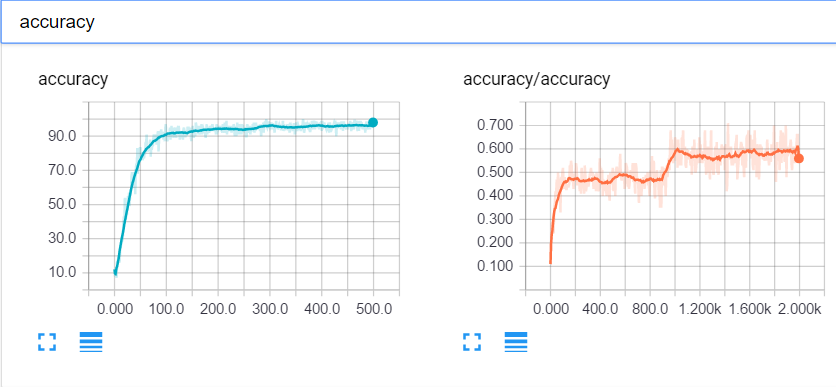


Figure 1 - Initial accuracy values as training progresses

Graph for mnist2.py file is produced by the default code, for cnn\_mnist I have modify the code to get the graph summary according to the below:

Added summary to measure accuracy:

|  |
| --- |
| trainAccuracy = tf.Variable(0.0, name="trainAccuracy")  tf.summary.scalar(tensor=trainAccuracy, name="accuracy") |

Added graph creation and summary writer

|  |
| --- |
| init = tf.global\_variables\_initializer()  summary\_op = tf.summary.merge\_all()  sess.run(init)  writer = tf.summary.FileWriter("problem1\_cnn\_2")  writer.add\_graph(sess.graph) |

Finally push data updates to the summary:

|  |
| --- |
| writer.add\_summary(sess.run(summary\_op, {trainAccuracy: temp\_train\_acc}), global\_step=i) |

And flush and close the writer:

|  |
| --- |
| writer.flush()  writer.close() |

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. (**45%)**

To fix and increase accuracy for mnist2.py, I studied in detail what in the other script that gets very high accuracy. According to that I slightly modify the following elements to be able to use the same values and hyperparameters.

1. Weight/filters dimensions. This weight applies to convolutional and fully connected layers

|  |
| --- |
| #1. changed W to 4x4 also size in and out hyperparameters are changed as per comment #2.  w = tf.Variable(tf.truncated\_normal([4, 4, size\_in, size\_out], stddev=0.1), name="W")  b = tf.Variable(tf.constant(0.1, shape=[size\_out]), name="B") |

1. Convolutional layers dimensions

|  |
| --- |
| #2.changed conv layers sizes  if use\_two\_conv:  conv1 = conv\_layer(x\_image, 1, 25, "conv1")  conv\_out = conv\_layer(conv1, 25, 50, "conv2")  else:  conv1 = conv\_layer(x\_image, 1, 50, "conv")  conv\_out = tf.nn.max\_pool(conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding="SAME")  flattened = tf.reshape(conv\_out, [-1, 7 \* 7 \* 50]) |

1. Fully connected layers dimensions

|  |
| --- |
| #3. Change fully connected layers  if use\_two\_fc:  fc1 = fc\_layer(flattened, 7 \* 7 \* 50, 100, "fc1")  embedding\_input = fc1  embedding\_size = 100  logits = fc\_layer(fc1, 100, 10, "fc2")  else:  embedding\_input = flattened  embedding\_size = 10  logits = fc\_layer(flattened, 7\*7\*50, 10, "fc") |

Not that the result is erratic, sometimes accuracy reaches 95%, but others hardly reach 90%. Anyway, results are quite better than initially. Below images show TensorBoard graphs for both accuracy and entropy. I have tried to make results more stable by modifying batch\_size, learning\_rate and some other values, but the results were worse, so I kept the original figures for these values.

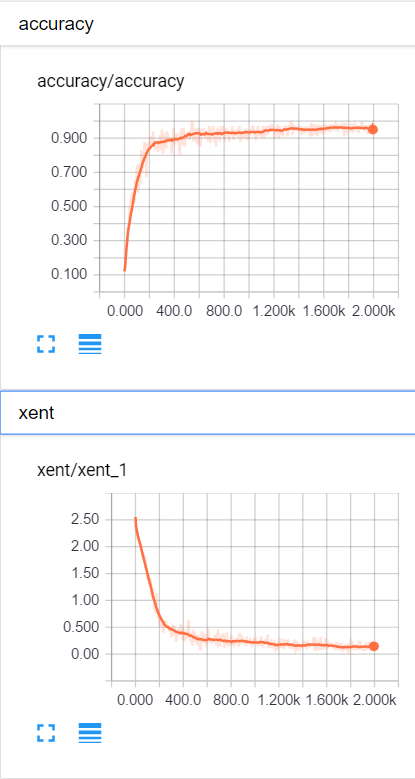


Figure 2 - Accuracy an entropy TensorBoard graphs for the fixed file

**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 also capture an image of “colorful” T-SNE Embedding. 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.

I have run the script one at a time for each model, manually changing the parameter required to fulfill the requirements of the problem.

* 2 CLs (convolution layer) and 2 FCLs (fully connected layer) for 1E-3, 1E-4 AND 1E-5 LRs (learning rates)
* 2CLs, 1FCL, 1E-3, 1E-4 AND 1E-5 LRs
* 1CL, 2FCLs, 1E-3, 1E-4 AND 1E-5 LRs
* 1CL, 1FCL, 1E-3, 1E-4 AND 1E-5 LRs

Below images show results for previous executions:

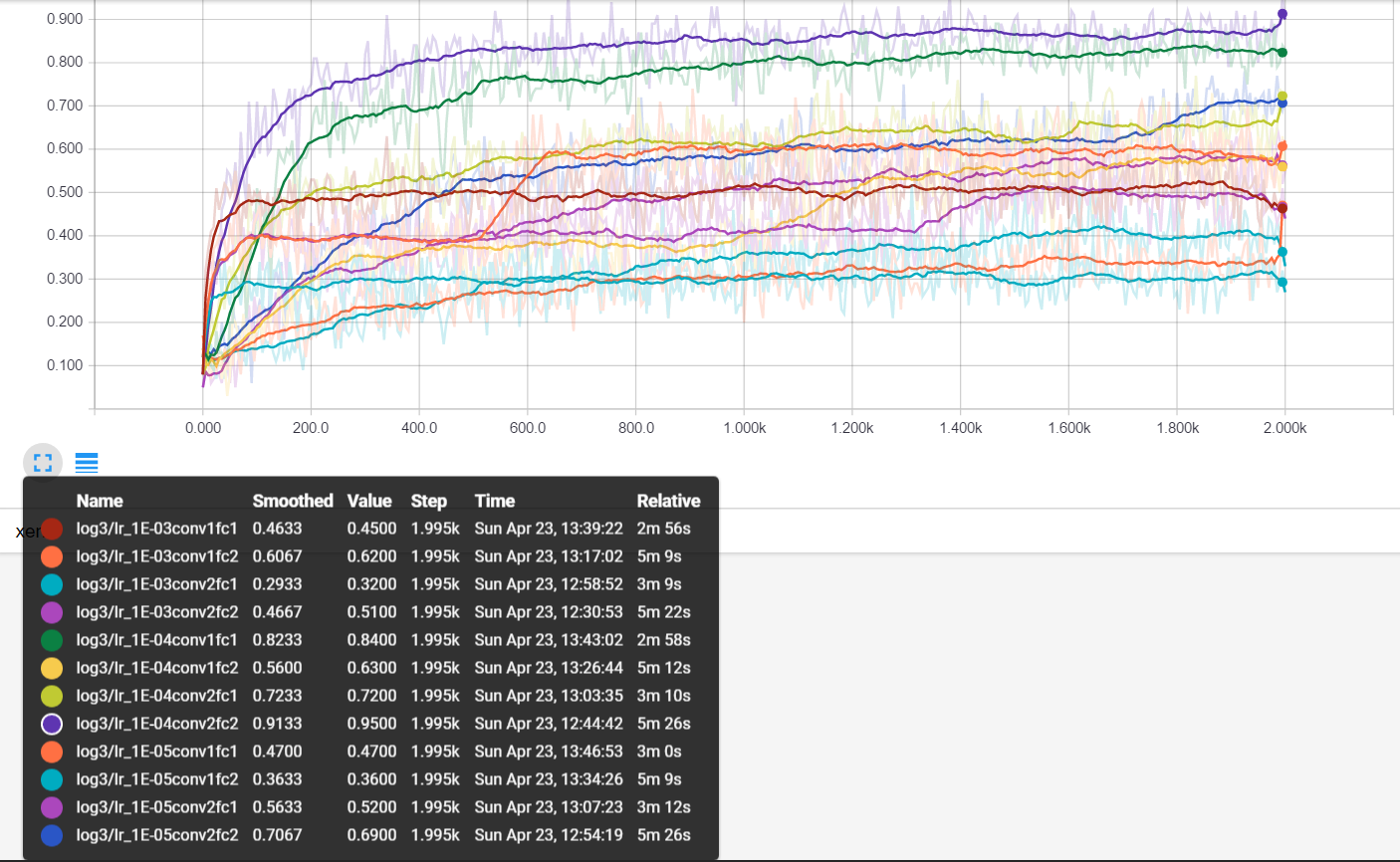


Figure 3 - Accuracy results for the 12 different configurations

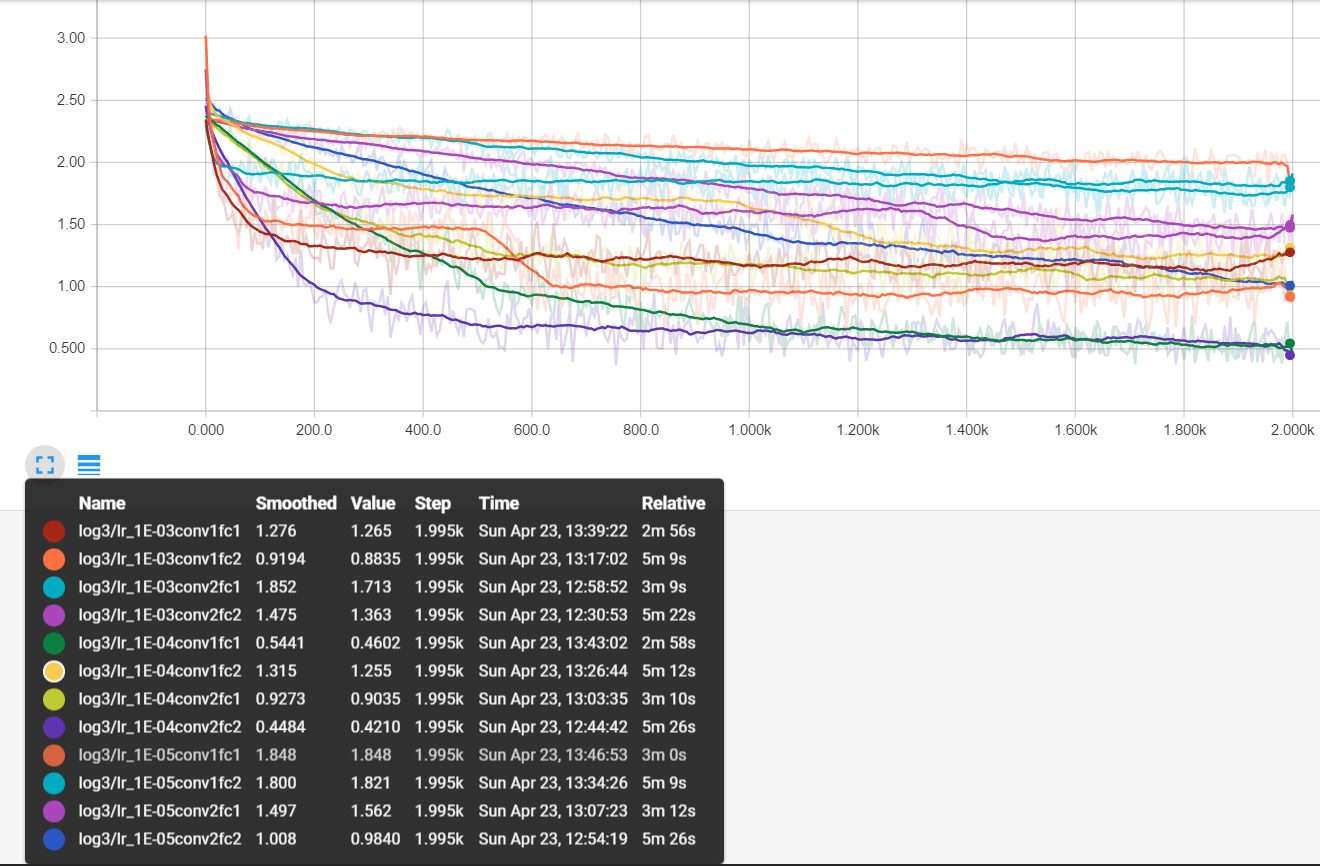


Figure 4 - Entropy results for the 12 different configurations



Figure 5 - Histogram results for one of the 12 configurations

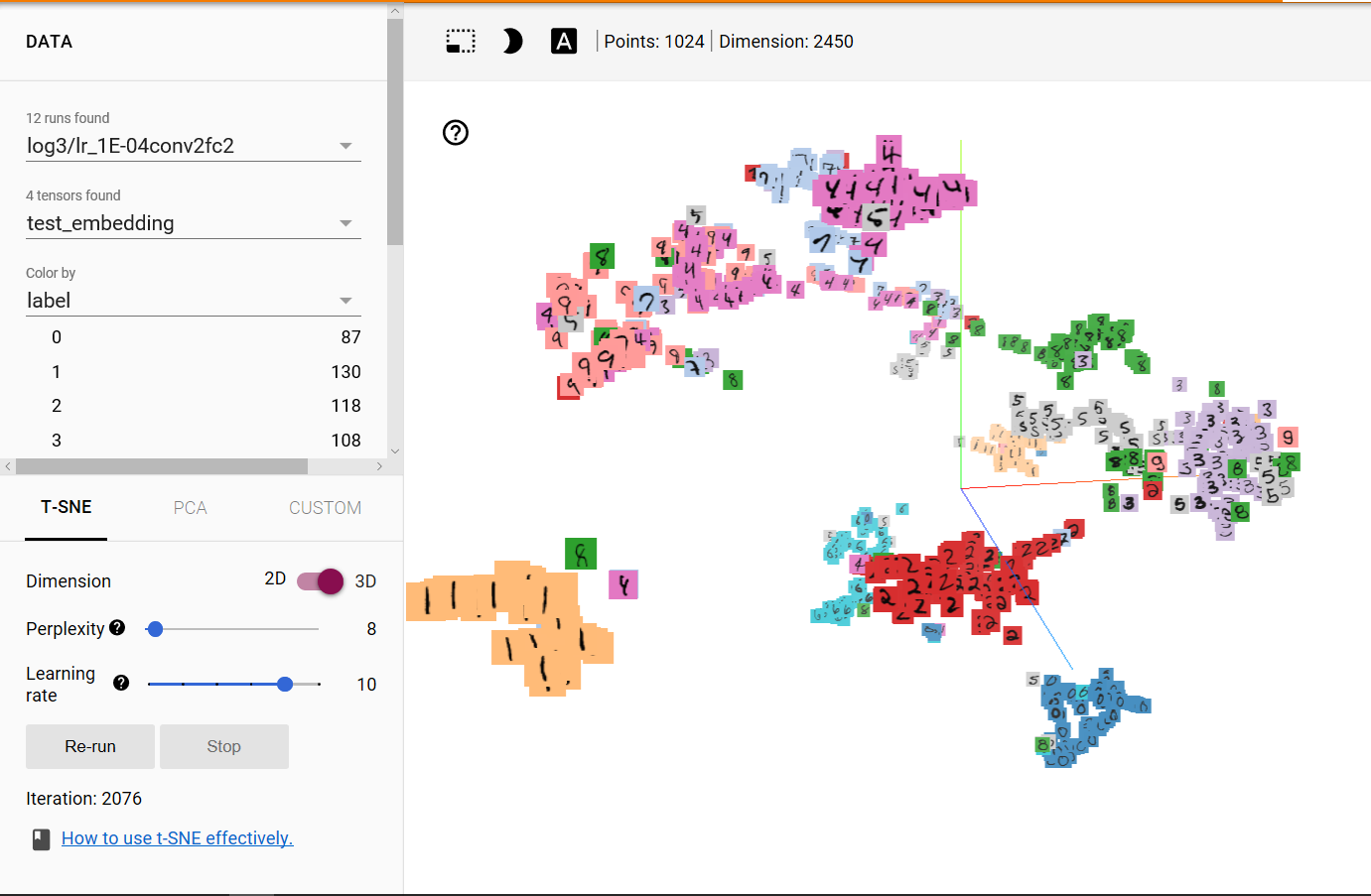


Figure 6 - “colorful” T-SNE Embedding image for one of the configurations

Collect execution times, final (smoothed) accuracies and final cross entropies for different models and provide tabulated presentation of the final results of different models **(20%)**

According to above configuration executions, the following results are obtained:

| **CL** | **FCL** | **LR** | **Execution time (s)** | **Final (smoothed) accuracy** | **Final cross entropy** |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1.00E-03 | 176 | 0.4633 | 1.265 |
| 1 | 2 | 1.00E-03 | 309 | 0.6067 | 0.8835 |
| 2 | 1 | 1.00E-03 | 189 | 0.2933 | 1.713 |
| 2 | 2 | 1.00E-03 | 322 | 0.4667 | 1.363 |
| 1 | 1 | 1.00E-04 | 178 | 0.8233 | 0.4602 |
| 1 | 2 | 1.00E-04 | 312 | 0.56 | 1.255 |
| 2 | 1 | 1.00E-04 | 190 | 0.7233 | 0.9035 |
| 2 | 2 | 1.00E-04 | 326 | 0.9133 | 0.421 |
| 1 | 1 | 1.00E-05 | 180 | 0.47 | 1.848 |
| 1 | 2 | 1.00E-05 | 309 | 0.3633 | 1.821 |
| 2 | 1 | 1.00E-05 | 192 | 0.5633 | 1.562 |
| 2 | 2 | 1.00E-05 | 326 | 0.7067 | 0.984 |

**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.

The below changes need to be done on the code to publish summaries to TensorBoard. For problem 1 I already included accuracy, but the script needs some additional changes to cope with loss in TensoBoard as well

1. Added summary to measure accuracy and loss/xent:

|  |
| --- |
| trainAccuracy = tf.Variable(0.0, name="trainAccuracy")  tf.summary.scalar(tensor=trainAccuracy, name="accuracy") |

1. Added graph creation and summary writer

|  |
| --- |
| init = tf.global\_variables\_initializer()  summary\_op = tf.summary.merge\_all()  sess.run(init)  writer = tf.summary.FileWriter("problem1\_cnn")  writer.add\_graph(sess.graph) |

1. Finally push data updates to both summaries:

|  |
| --- |
| writer.add\_summary(sess.run(summary\_op, {trainAccuracy: temp\_train\_acc, trainLoss: temp\_train\_loss}), global\_step=i) |

1. Last, flush and close the writer:

|  |
| --- |
| writer.flush()  writer.close() |

And the results in Tensorboard:

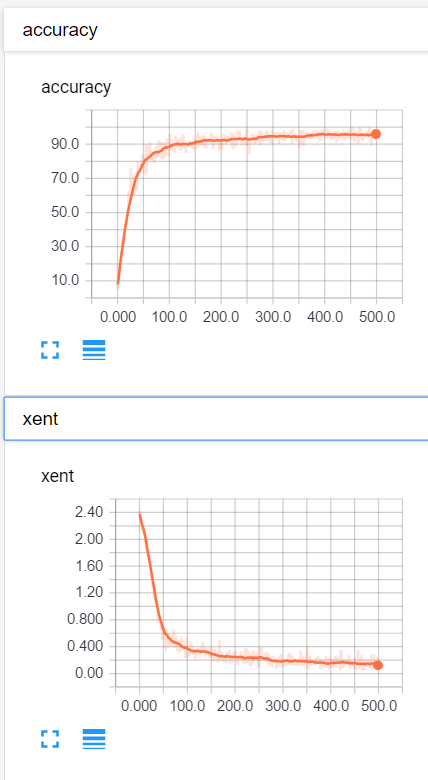


Figure 7 Accuracy and Loss summaries for cnn\_mnist.py in Tensorborad

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. **(35%).**

The initial difference is that values for mnist2 are poorer, lower accuracy and higher loss, what is depicted on the below images:

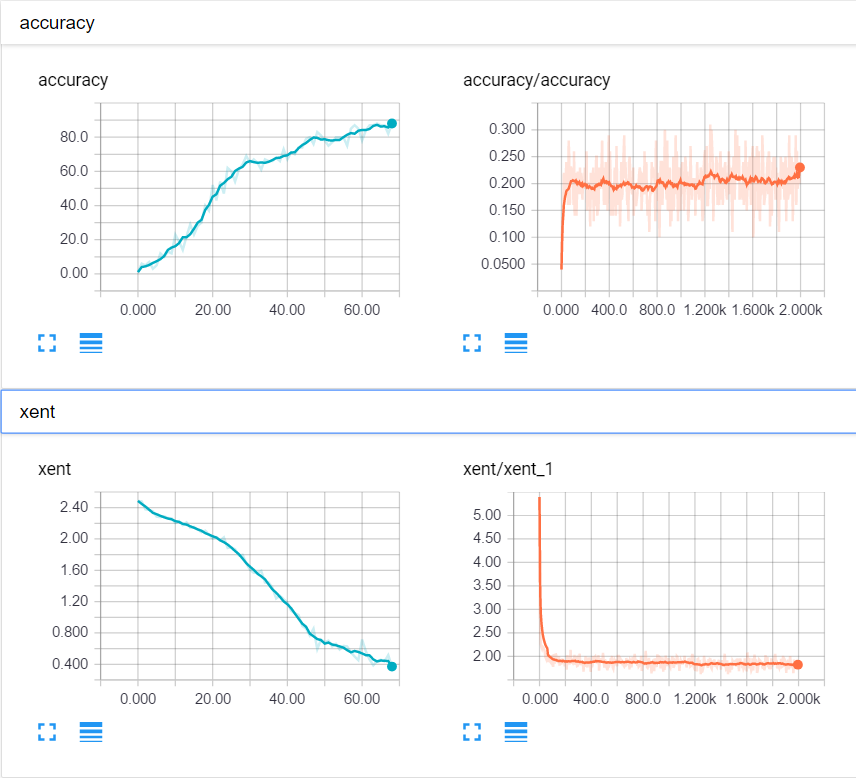


Figure 8 - Two scripts summary graphs comparison

Another noticeable point, is that [mnist2.py](http://35.184.207.137:8888/edit/mnist2.py) (orange color), increases accuracy and decreases loss very quick but then stabilized, and there is no hardly improvement on both scalars. On the other hand cnn\_mnist.py file (cyan color) improves both values with a lower slope, but if finally reach much better results as the training process progresses. In problem 1, mnist2.py was modified to use same hyperparameters as in cnn\_mnist.py to achieve much higher accuracies (above 90%).

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of all essential command and snippets of results into the Word document with explanations of the purpose of those commands. We cannot retype text that is in JPG images. Please, always submit a separate copy of the original, working scripts and/or class files you used. Sometimes we need to run your code and retyping is too costly. Please include in your MS Word document only relevant portions of the console output or output files. Sometime either console output or the result file is too long and including it into the MS Word document makes that document too hard to read. PLEASE DO NOT EMBED files into your MS Word document. For issues and comments visit the class Discussion Board.

If you are working with Jupyter Notebooks please provide clear and full comments for all of important steps or changes you are making. Please provide the notebook itself (ipynb file) and the PDF version of the file. Canvas cannot read ipynb file and if you do not provide an MS Doc or PDF version of your work, you will be penalized.

It is not acceptable that you describe your solution of any of these problems on Piazza.