## HU Extension Assignment 12 E63 Big Data Analytics

### Handed out: 11/17/2017 Due by 4:00 PM EST on Saturday, 12/02/2017

**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. (**45%)**

I played around with various combinations of the architecture, choosing 2C + 1F , then 1C + 2F, then 2C +2F

I also tried various combinations of the learning rate with values provided [1E-3, 1E-4, 1E-5], then .005, and finally .002

I changed the range to 300. I started off by using 100 first, and later kept increasing it to get higher accuracy.

I also played around switching back and forth between the AdamOptimizer and MomentumOptimizer.

Further optimization came from shrinking the FC layer down multiple iterations to finally 40. I was able to reach accuracy of 90+% although the end result came to be 86%

Sample output :

**Generation # 256. Train Acc (Test Acc): 93.33**

Generation # 257. Train Acc (Test Acc): 91.33

Generation # 258. Train Acc (Test Acc): 88.67

Generation # 259. Train Acc (Test Acc): 87.33

Generation # 260. Train Acc (Test Acc): 92.67

Generation # 261. Train Acc (Test Acc): 89.33

Generation # 262. Train Acc (Test Acc): 84.00

Generation # 263. Train Acc (Test Acc): 83.33

**Generation # 264. Train Acc (Test Acc): 92.00**

Generation # 265. Train Acc (Test Acc): 87.33

Generation # 266. Train Acc (Test Acc): 85.33

Generation # 267. Train Acc (Test Acc): 88.00

Generation # 268. Train Acc (Test Acc): 88.00

Generation # 269. Train Acc (Test Acc): 88.00

Generation # 270. Train Acc (Test Acc): 86.67

Generation # 271. Train Acc (Test Acc): 88.00

Generation # 272. Train Acc (Test Acc): 86.67

Generation # 273. Train Acc (Test Acc): 90.67

Generation # 274. Train Acc (Test Acc): 89.33

Generation # 275. Train Acc (Test Acc): 85.33

Generation # 276. Train Acc (Test Acc): 85.33

Generation # 277. Train Acc (Test Acc): 86.67

Generation # 278. Train Acc (Test Acc): 90.00

Generation # 279. Train Acc (Test Acc): 89.33

Generation # 280. Train Acc (Test Acc): 87.33

Generation # 281. Train Acc (Test Acc): 89.33

Generation # 282. Train Acc (Test Acc): 82.67

Generation # 283. Train Acc (Test Acc): 86.67

Generation # 284. Train Acc (Test Acc): 89.33

Generation # 285. Train Acc (Test Acc): 89.33

Generation # 286. Train Acc (Test Acc): 88.67

Generation # 287. Train Acc (Test Acc): 86.67

Generation # 288. Train Acc (Test Acc): 88.67

Generation # 289. Train Acc (Test Acc): 88.00

**Generation # 290. Train Acc (Test Acc): 90.67**

Generation # 291. Train Acc (Test Acc): 87.33

Generation # 292. Train Acc (Test Acc): 89.33

Generation # 293. Train Acc (Test Acc): 87.33

Generation # 294. Train Acc (Test Acc): 87.33

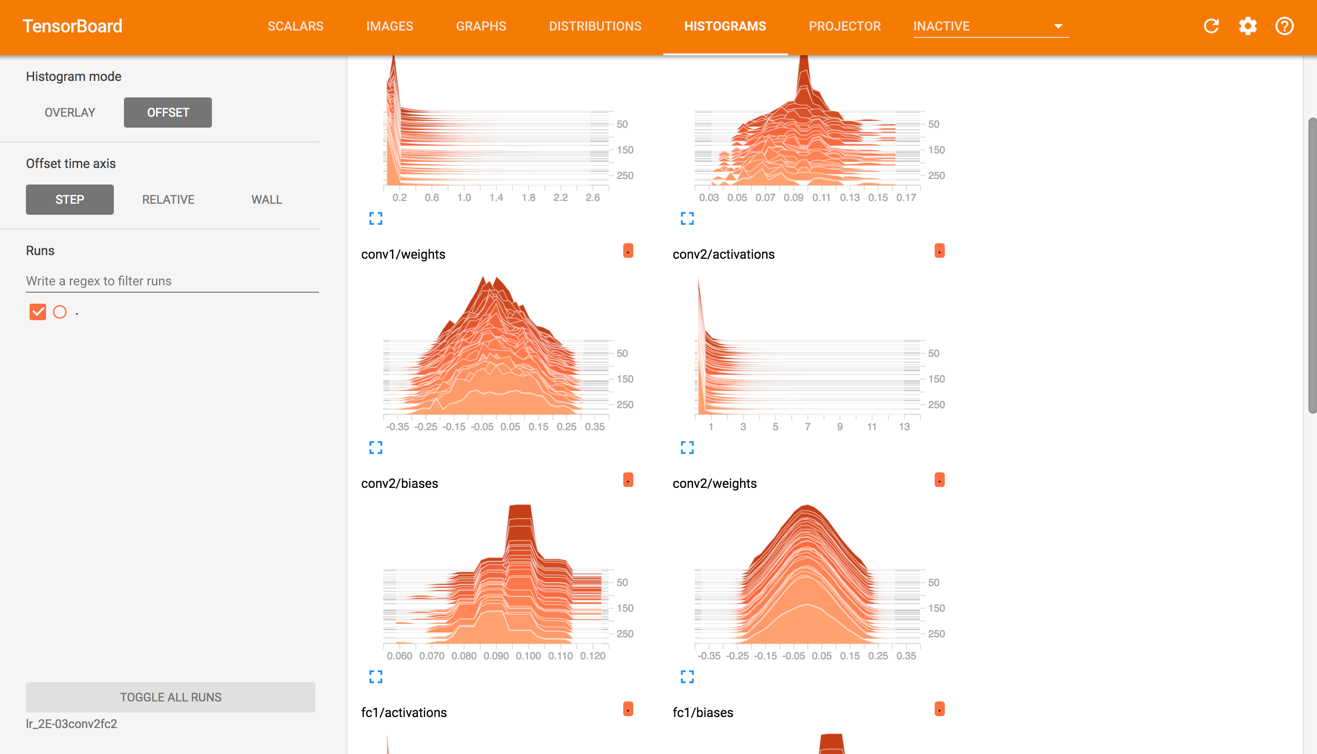
Generation # 295. Train Acc (Test Acc): 82.67

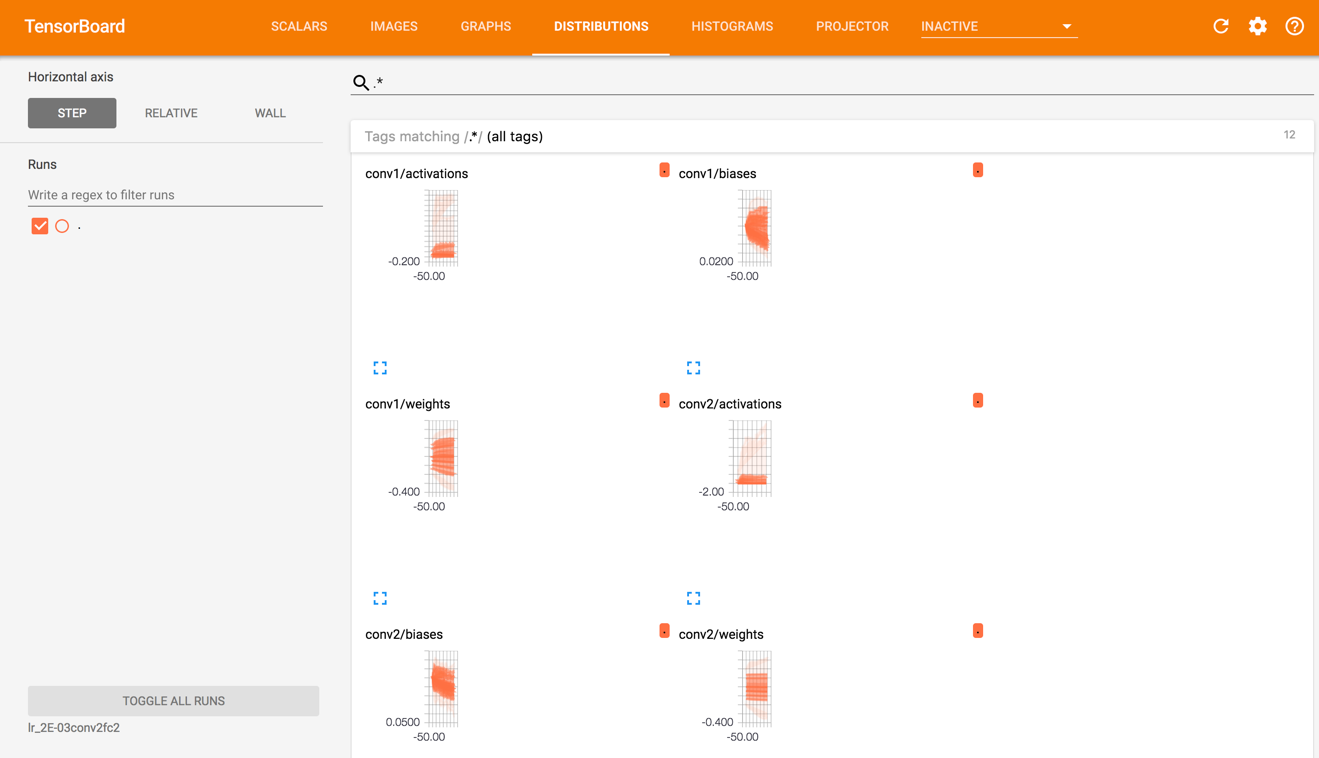
Generation # 296. Train Acc (Test Acc): 89.33

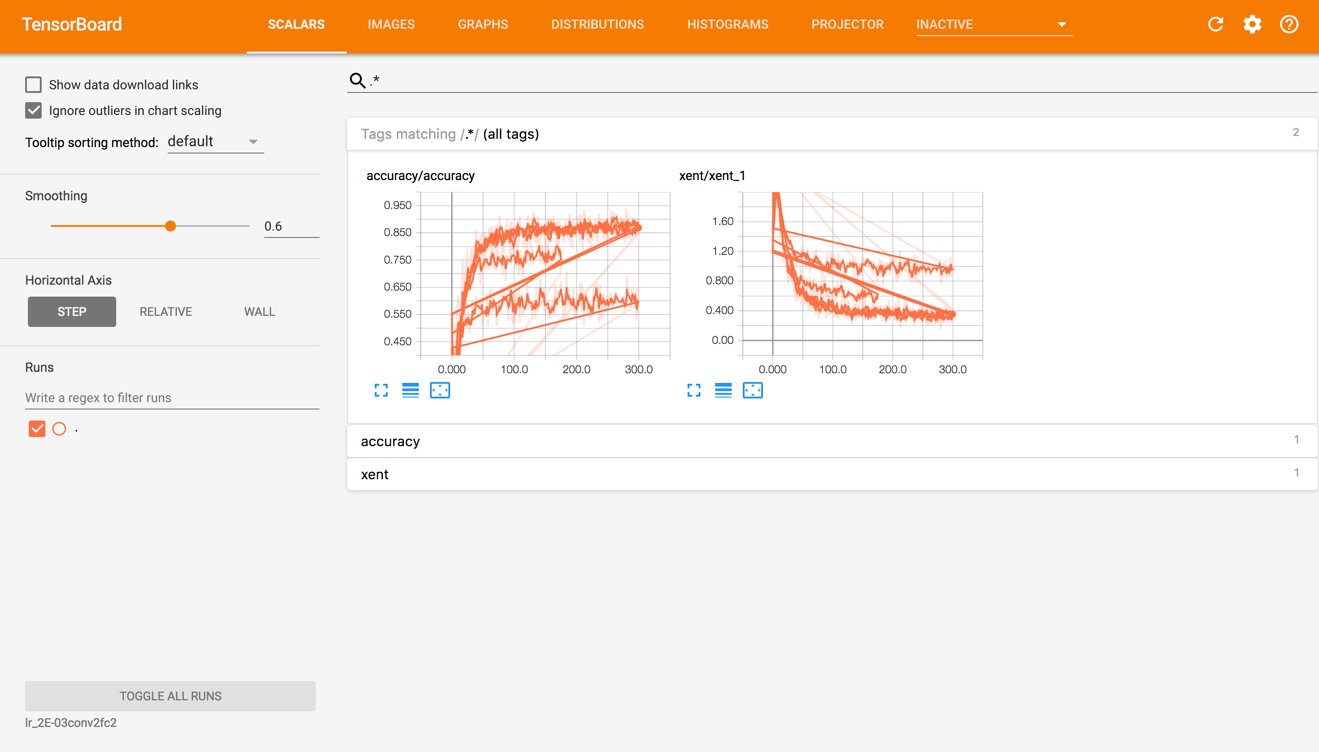
Generation # 297. Train Acc (Test Acc): 86.00

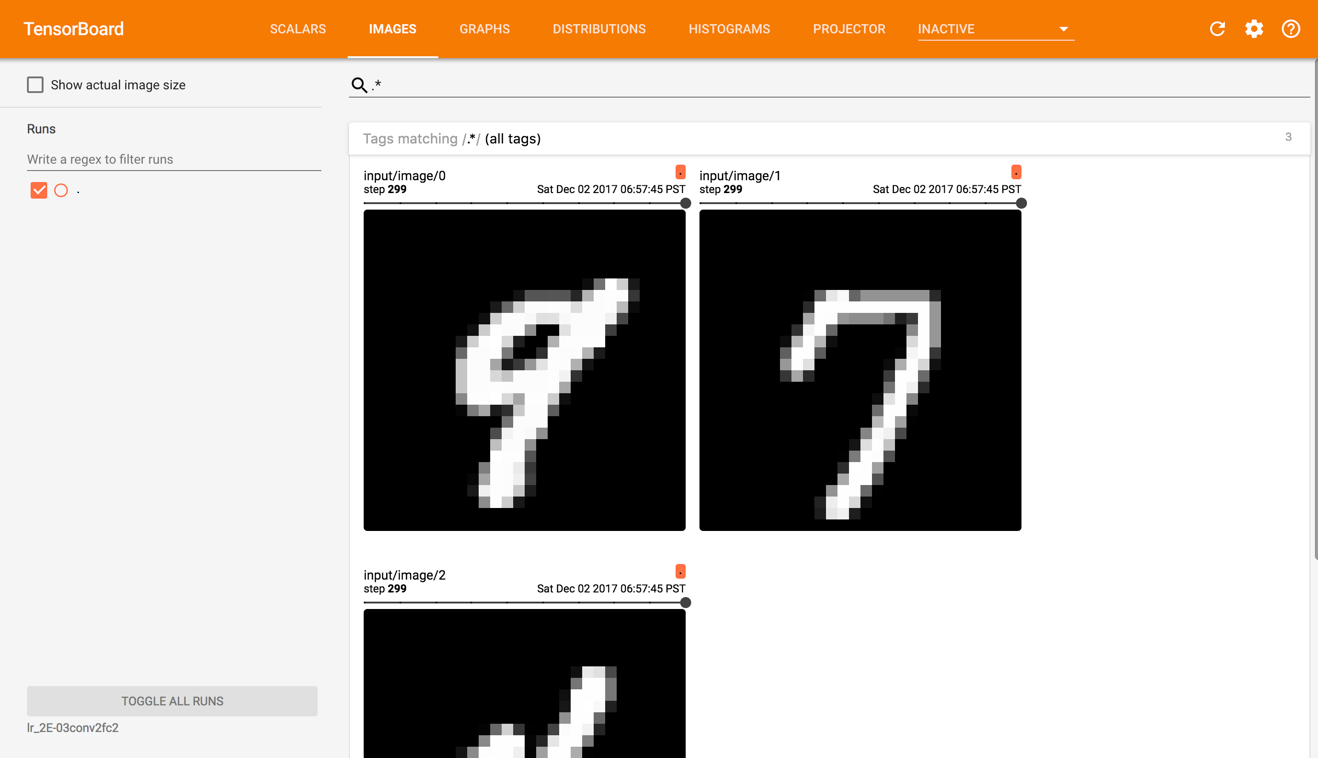
Generation # 298. Train Acc (Test Acc): 88.67

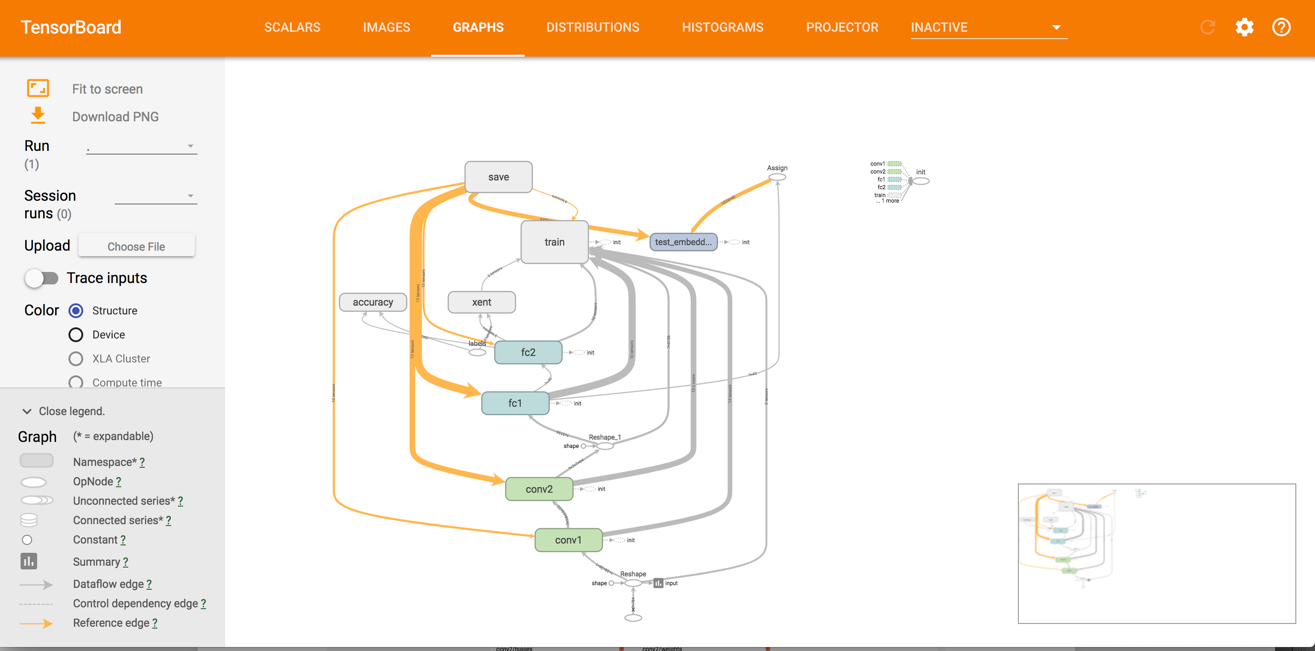
**Generation # 299. Train Acc (Test Acc): 86.67**





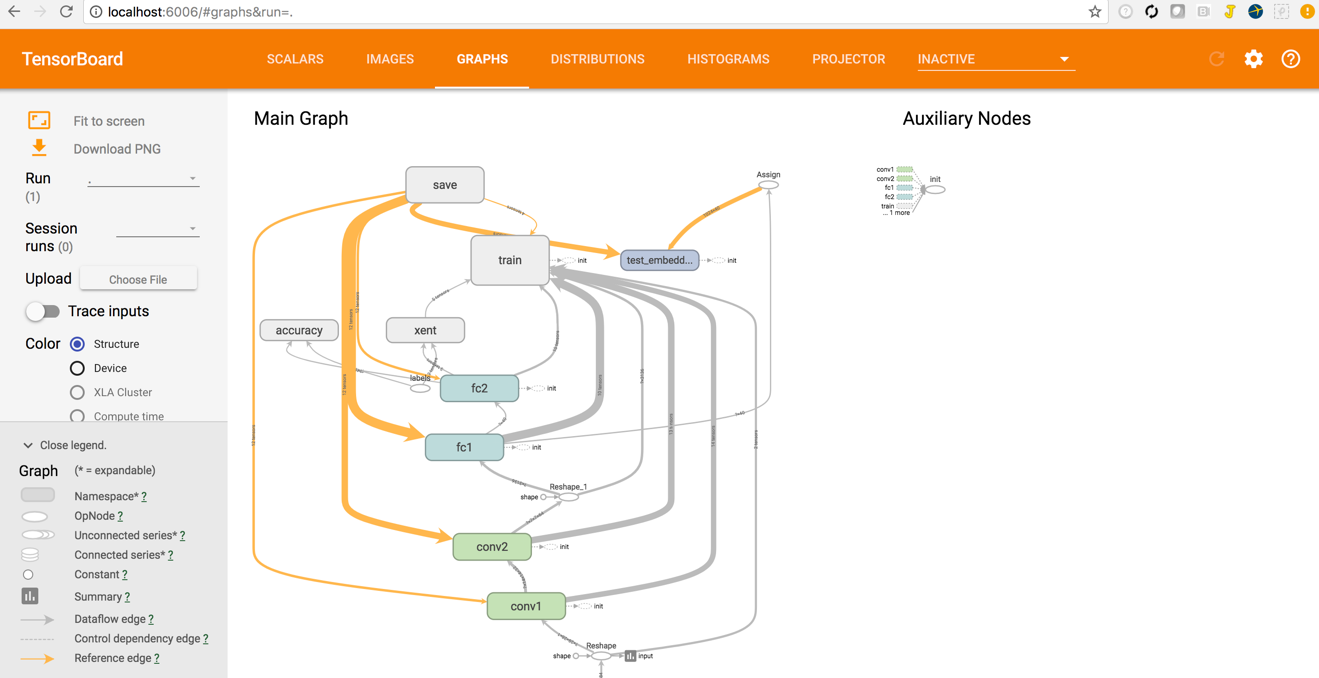


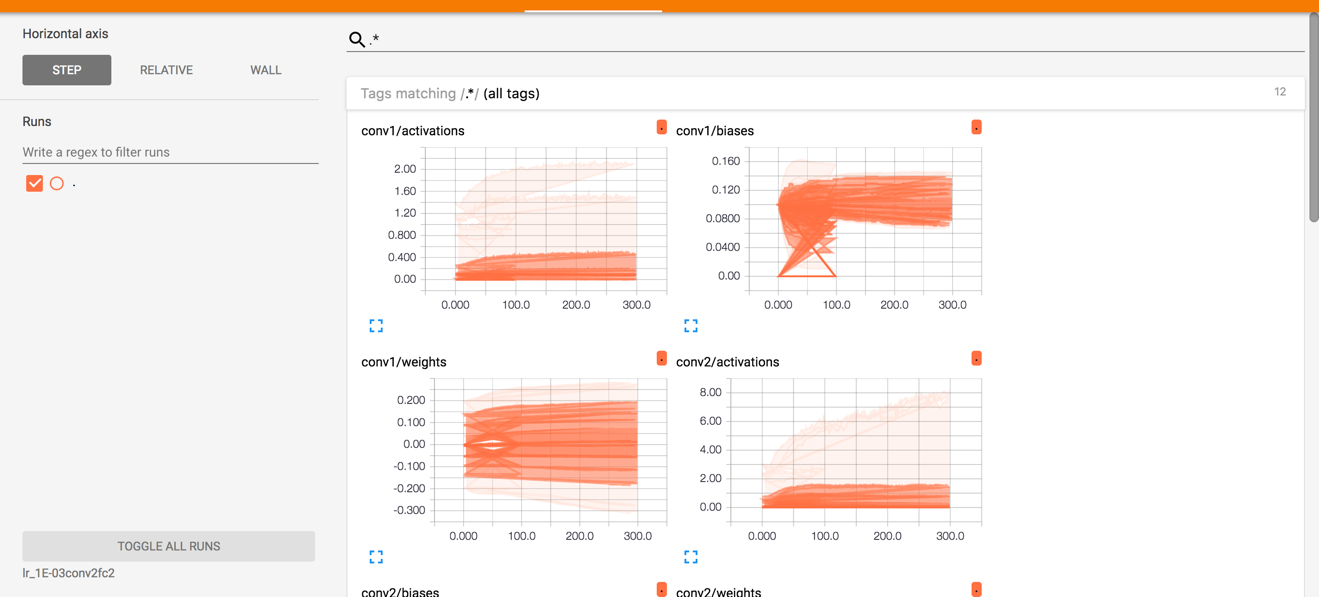
****

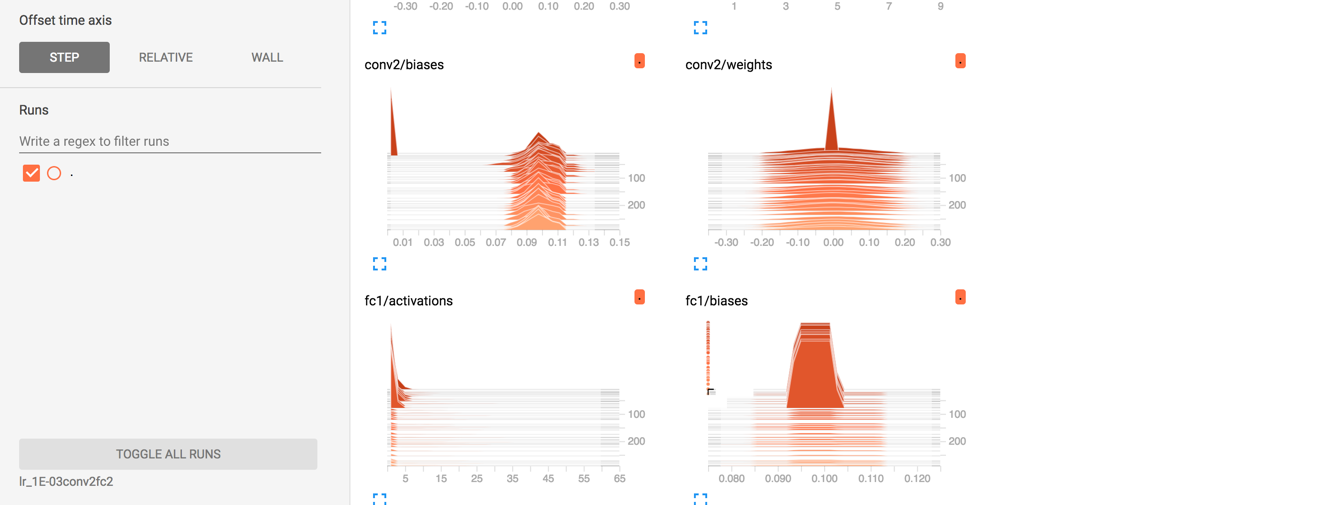
****

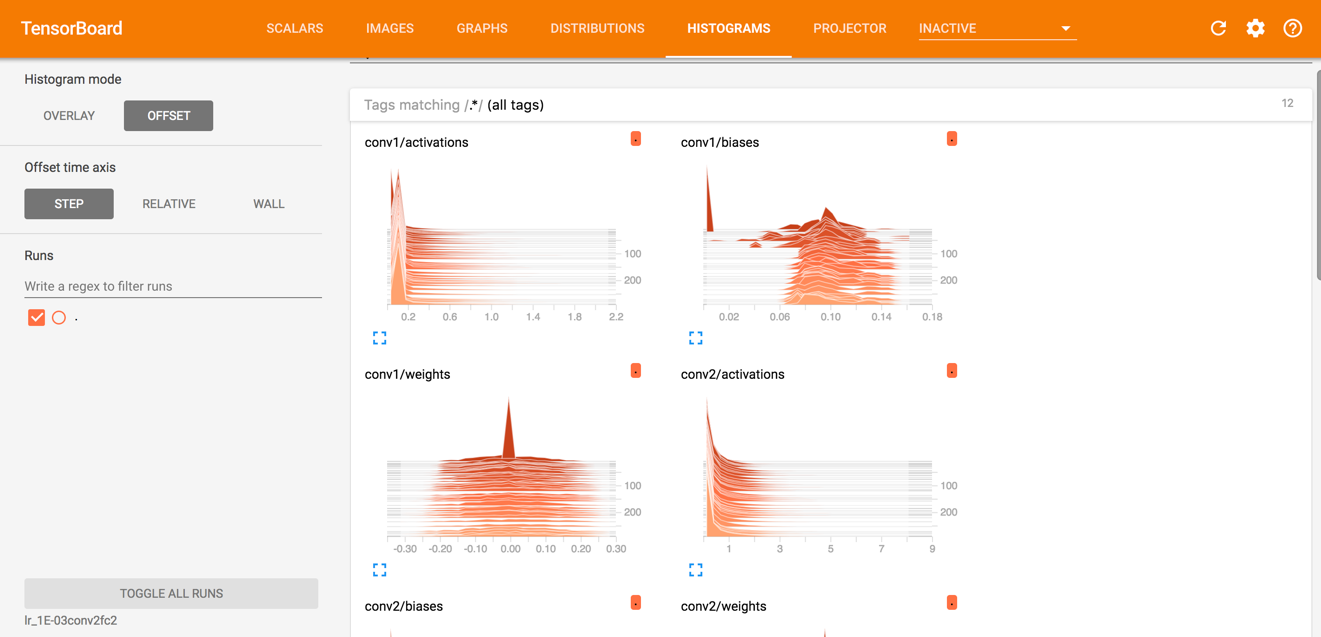
**Problem 2.** Run corrected version of mnist2.py for 4 different architectures (2 conv, 1 conv, 2 fully connected, 1 fully connected layer) and 3 values of the learning rate. As one learning rate choose the one you selected in Problem 1 and then add one smaller and one larger learning rate around that 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. 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%)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Learning rate*** | ***# Conv layers*** | ***# FC***  ***layers*** | ***Accuracy*** | ***Cross entropy*** | ***Execution time (sec)*** |
| .001 | 2 | 2 | 80.00 | .54 | 86.56 |
| .001 | 1 | 2 | 70.67 | .81 | 86.846 |
| .001 | 2 | 1 | 46.67 | 1.3 | 92.658 |
| .001 | 1 | 1 | 52.00 | 1.3 | 90.52 |
| .002 | 2 | 2 | 90.00 | .3 | 87.43 |
| .002 | 1 | 2 | 64.67 | .8 | 87.577 |
| .002 | 2 | 1 | 48.00 | 1.3 | 90.9 |
| .002 | 1 | 1 | 47.33 | 1.3 | 88.943 |
| .005 | 2 | 2 | 66.67 | 2.3 | 86.232 |
| .005 | 1 | 2 | 8.67 | 2.3 | 85.018 |
| .005 | 2 | 1 | 14.67 | 2.3 | 87.349 |
| .005 | 1 | 1 | 30.67 | 1.8 | 84.493 |

****

****

****

****

Its was not clear if we have to show only one graph. I put the info in the table and showed one such graph here.

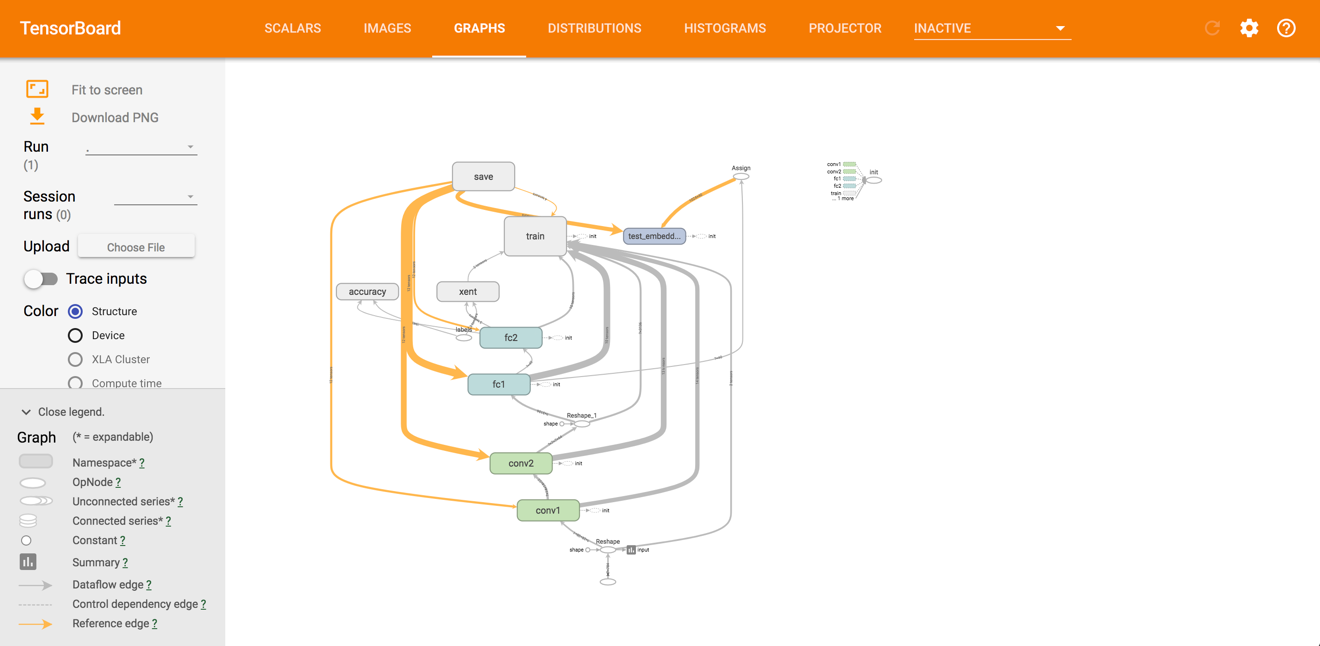
**Problem 3**. Modify file cnn\_mnist.py so that it publishes its summaries to the TensorBoard. Describe changes you are making and provide images of Accuracy and Cross Entropy summaries as captured by the Tensor Board. Provide the Graph of your model. Describe the differences if any between the graph of this program and the graph generated by mnist2.py script running with 2 convolutional and 2 fully connected layers. Provide working code. **(35%).**

I added lines to create a graph writer, and added lines to calculate accuracy (I was stuck here for sometime). Added scalar summary for loss and accuracy.Then made a sess.run call to accuracy and write it to the graph.

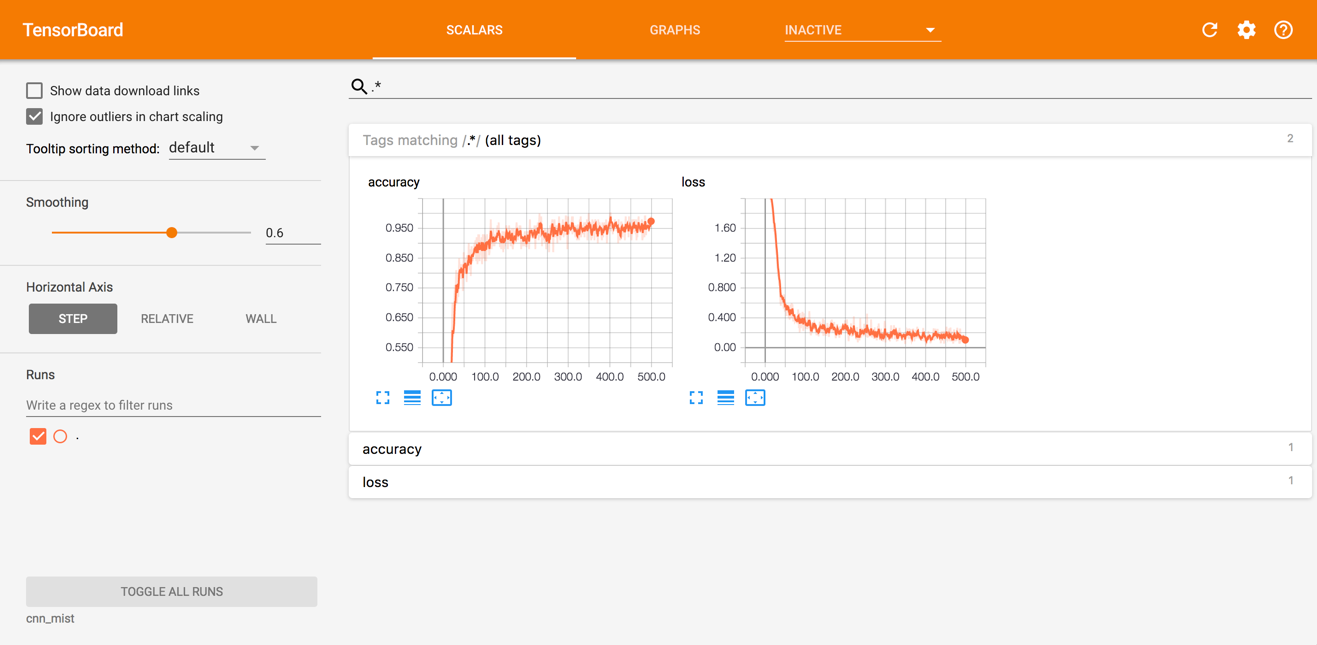
I got XEntropy 0.3 and accuracy of 90 for 2 convolutional and 2 fully connected layer using mnist2.py which I modified. I have attached the graphs here.

The graph for mnist2.py is more cleaner and easier to read because of the scopes defined

mnist2.py



cnn\_mnist.py

****



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.