## HU Extension Assignment 12 E63 Big Data Analytics

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

**STUDENT: HUYNH VO**

**Please note: I have changed the iterator in range(601) for all tensorflow because the range(2001) takes a very long time to perform. Thanks!**

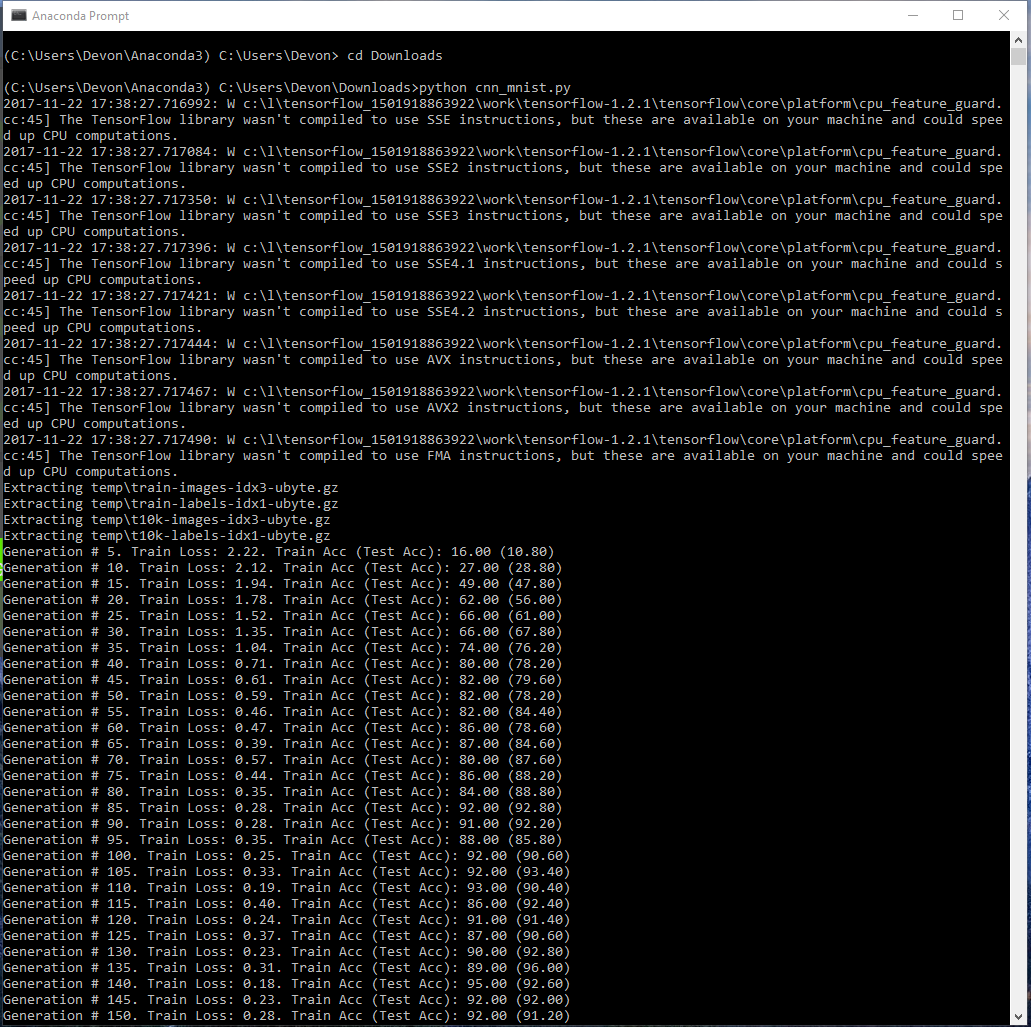
**Problem 1.** Please find attached 2 files from Google’s tutorials sets. I used file mnist2.py for preparation of my notes. If you read the file carefully you will see that you can run it in at least two modes. The way it is setup now it selects one learning rate and one particular neural network architecture 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%)**

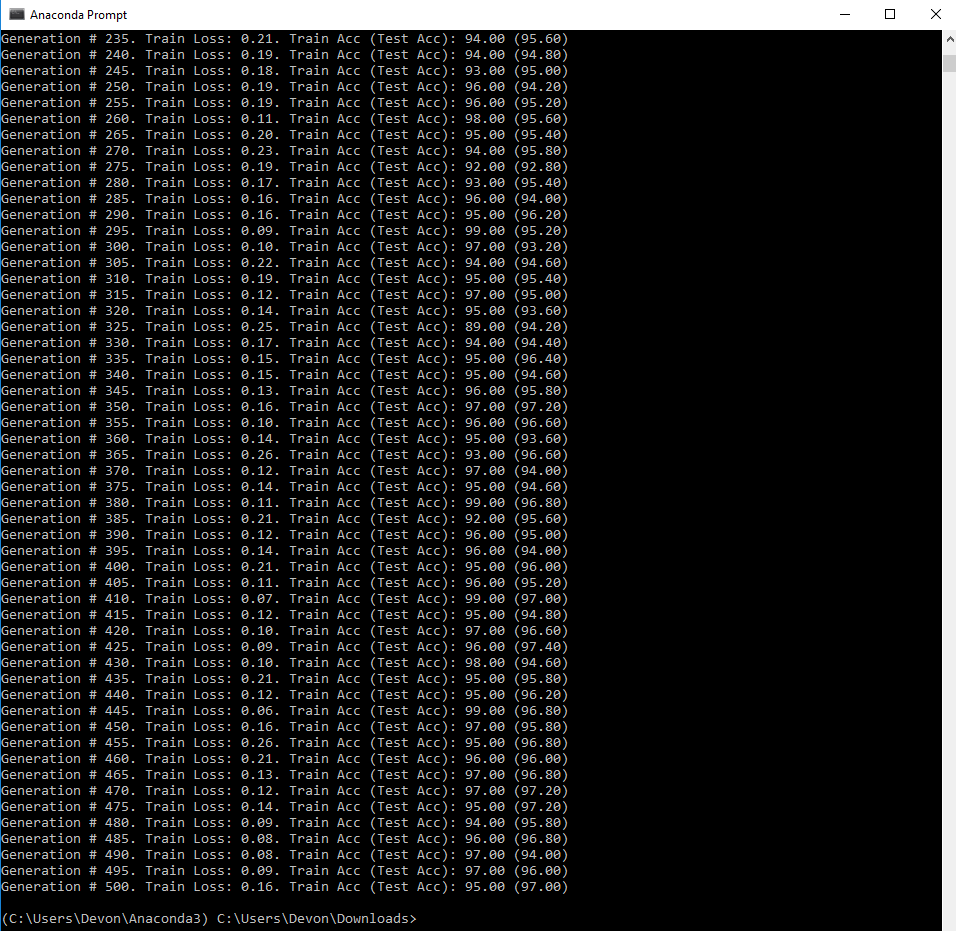
**Answer:**

**Below, I ran cnn\_mnist.py. It had a % accuracy of about 95%.**

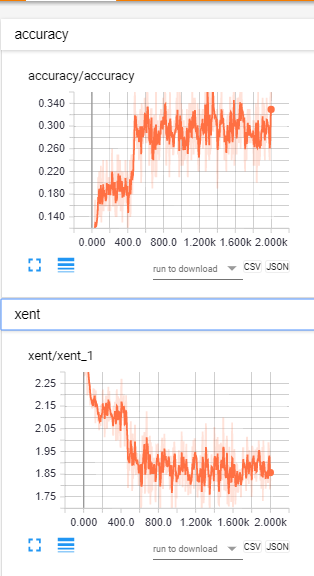
**(C:\Users\Devon\Anaconda3) C:\Users\Devon> cd Downloads**

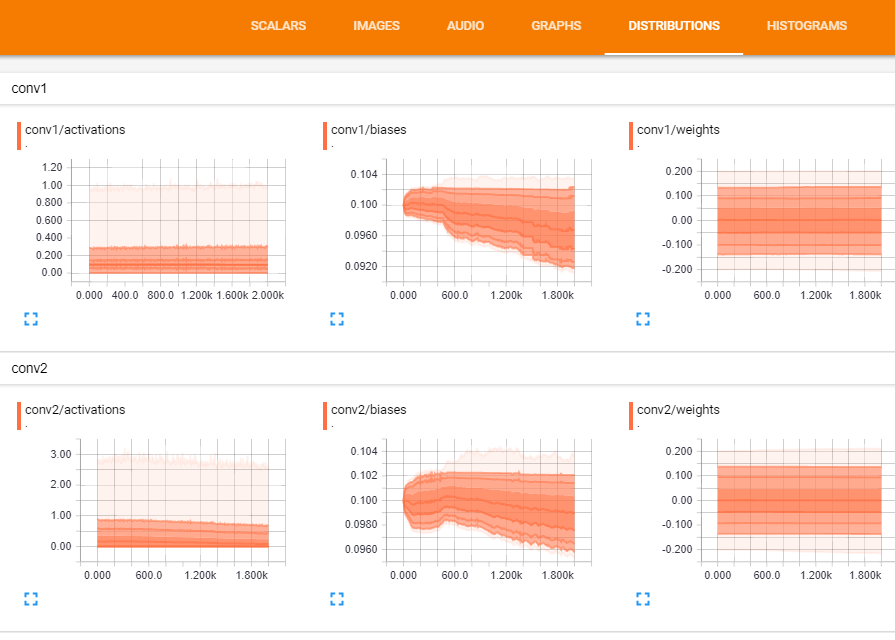
**(C:\Users\Devon\Anaconda3) C:\Users\Devon\Downloads>python cnn\_mnist.py**

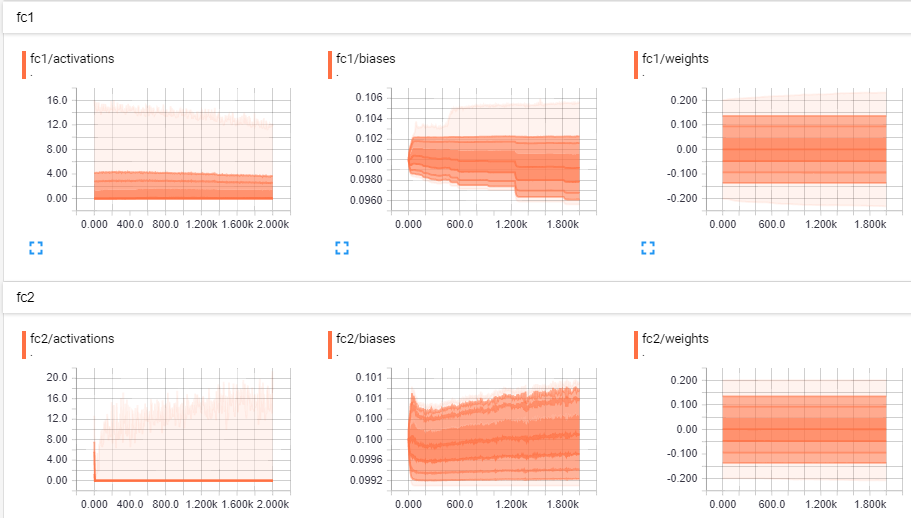




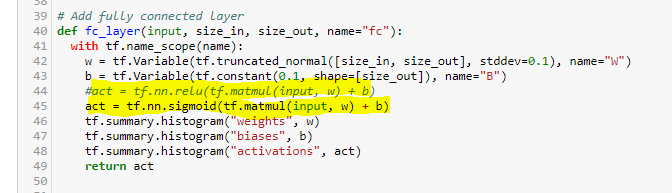
**Results below are from the original mnist2.py. I created mnist2\_HW12P1.ipynb to run it. Its original accuracy graph is below (about 35% accuracy).**

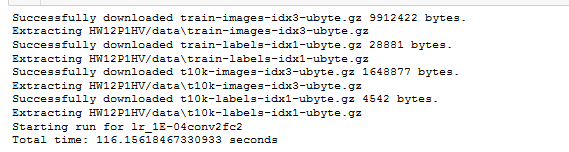






**In order to fix mnist2.py, I have modified and created mnist2\_HW12P1.ipynb with the below modifications. I modified the act function from tf.nn.relu to tf.nn.sigmoid. The %accuracy has increased to 95%.**

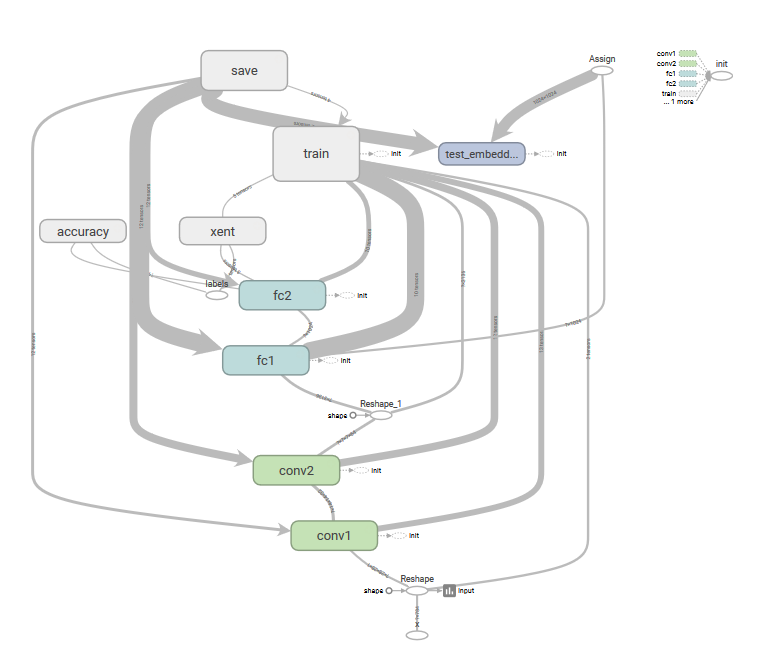




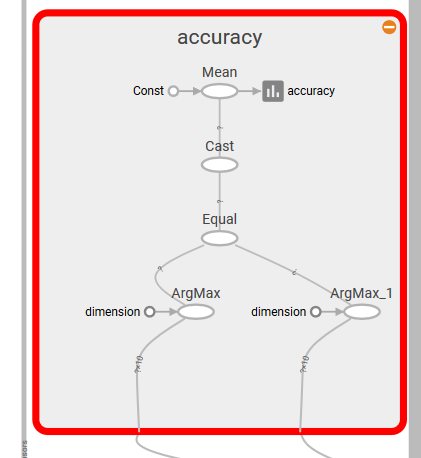
**I then opened Anaconda Prompt and activated Tensorboard. I exported the accuracy to Tensorboard so that the program’s progress could be followed on the Scalar tab.**



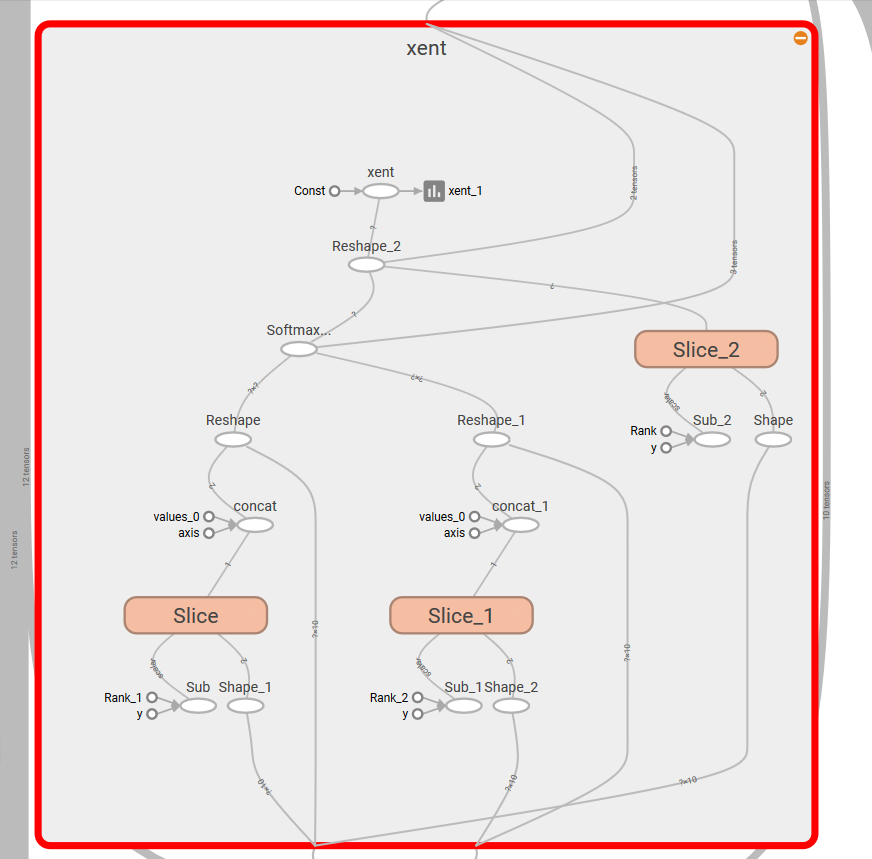
**Below is the overall Graph:**

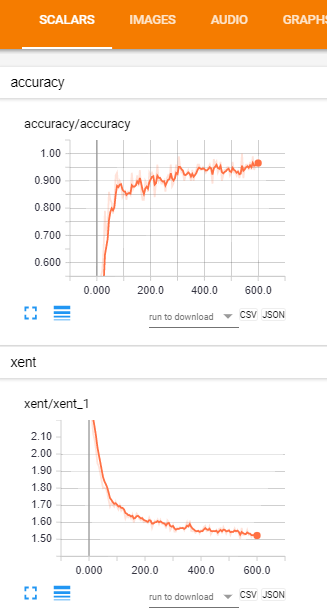


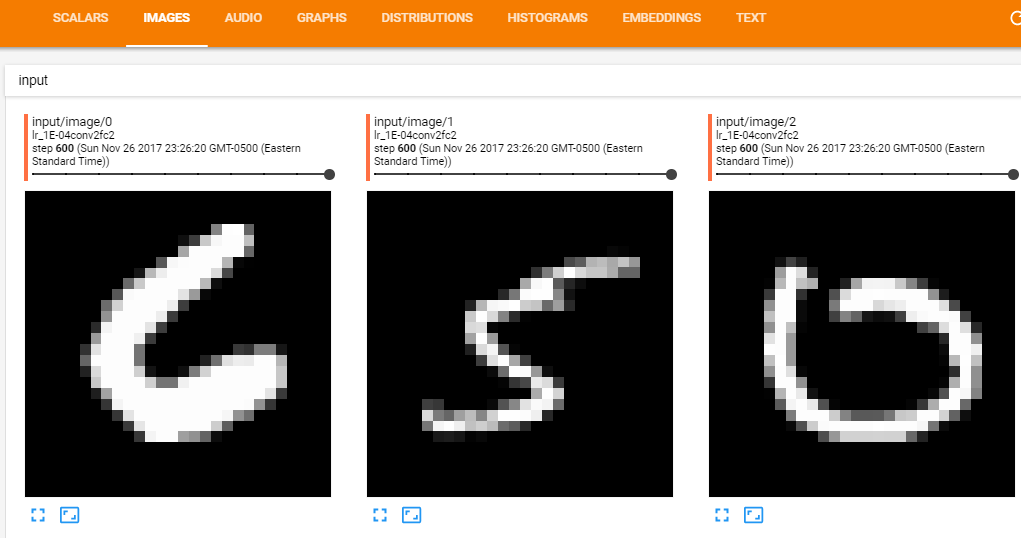
**Accuracy:**



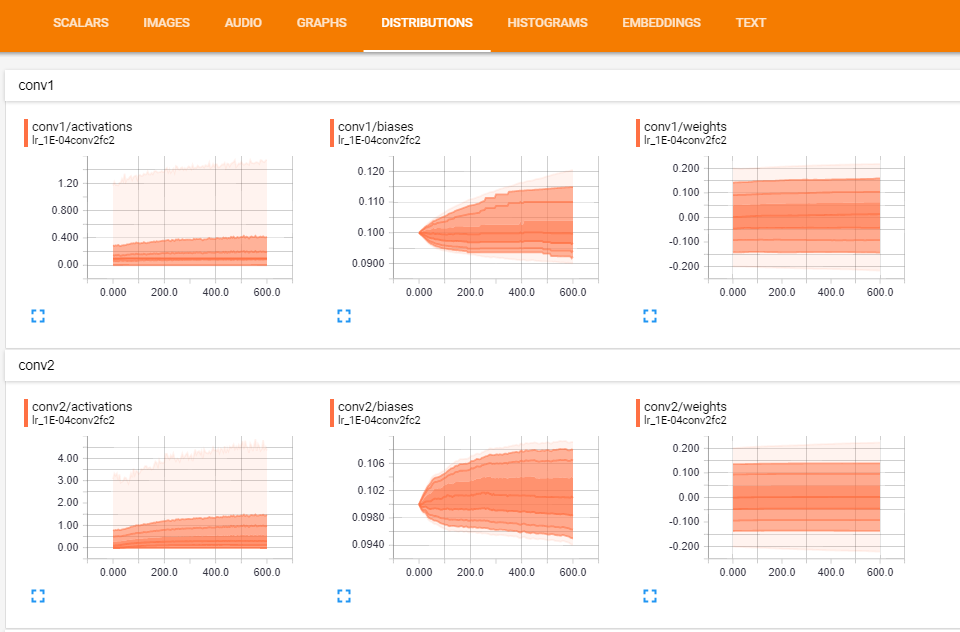
**Cross Entropy (xent):**





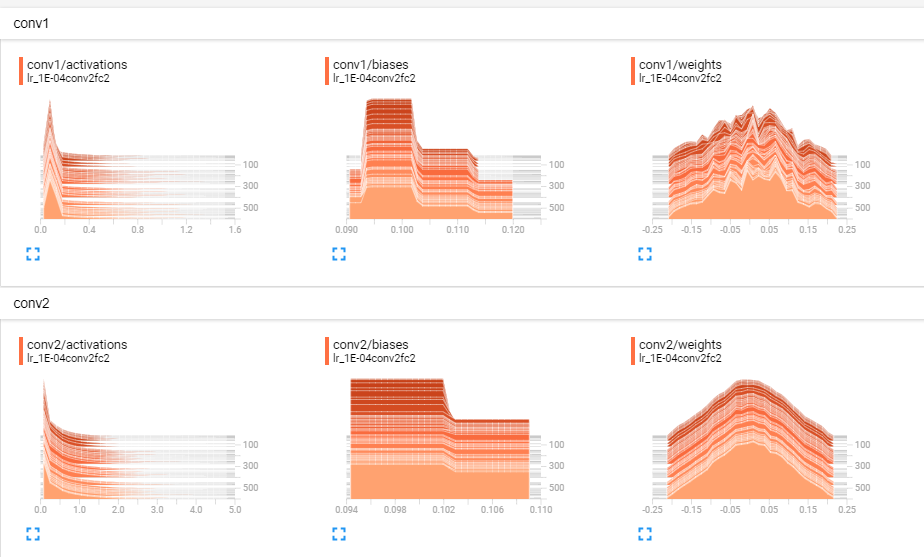


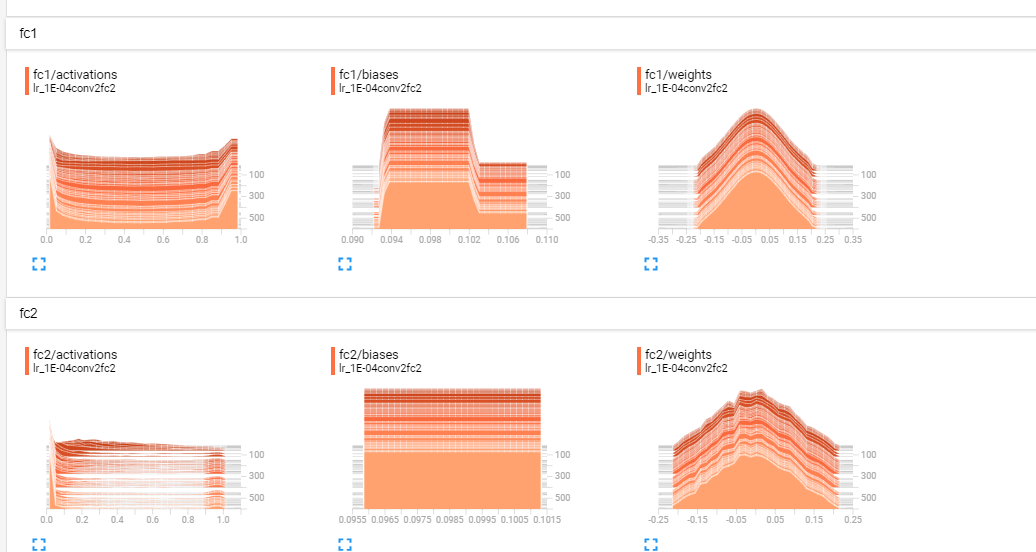
**Below are distributions:**





**Below are histograms:**





**I only fixed one thing (the ‘act’ operation) for a fully connected layer function. I believe the sigmoid function works better because it provides a type of smooth nonlinearity for use in a neural network. It computes the sigmoid of x element-wise. Specifically y = 1/ (1+exp(-x)).**

**Before I obtained this 95%accuracy, I played around with *use\_two\_fc in [True, False], for use\_two\_conv in [True, False]*. I noticed that without changing anything, just by switching to *for use\_two\_conv in [False],* my %accuracy went up to 85%. However, the best results I obtained (more than 95%) is when I kept *for use\_two\_fc in [True],* and  *for use\_two\_conv in [True]* , in addition to changing the relu operation to sigmoid at the act2 belongs to *fully connected layer* (please see mnist2\_corrected\_HV.ipynb).**

**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%)**

**Answer:**

**Below are captures of Accuracy (summary) graphs and One of Histograms to demonstrate that my code is working.**

**You can also open mnist2\_corrected\_HV.ipynb to run the code, you will see colorful graphs and more detailed information.**

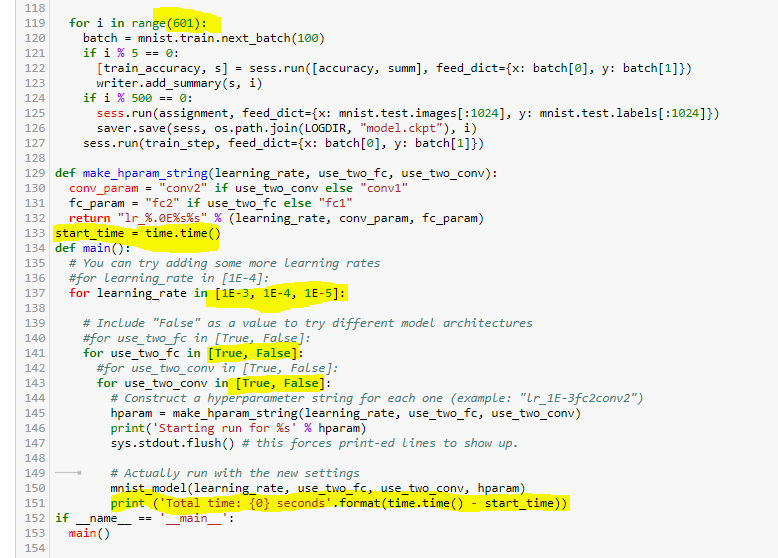
**In order to obtain execution times for each model, I added time operation to my file.**

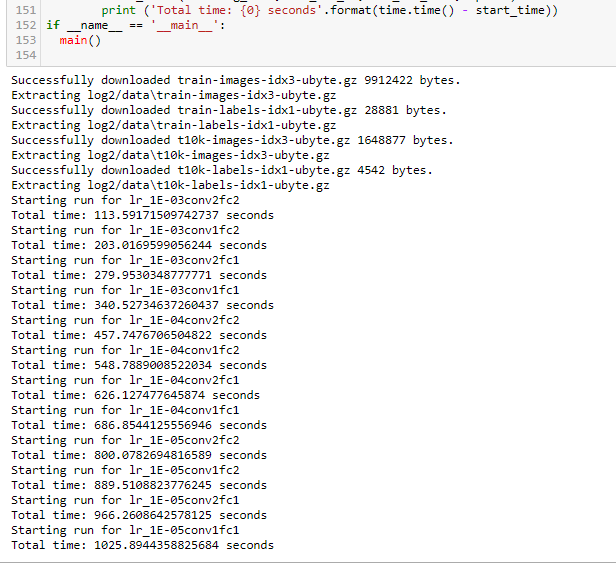
**My observation after running 12 models is that learning\_rate = 0.001 has the best model comparing to 0.0001 and 0.00001. The model that using learning\_rate = 0.00001 has the least %accuracy. Also, using conv2 and fc2 provide the best accuracy when combining to learning\_rate = 0.001.**



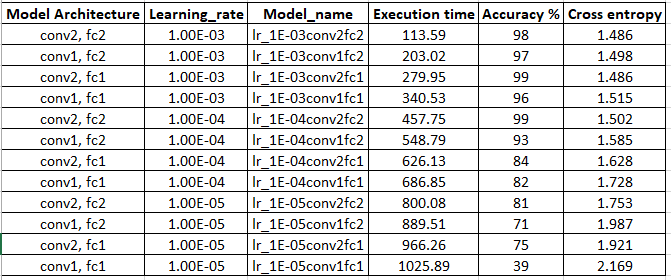






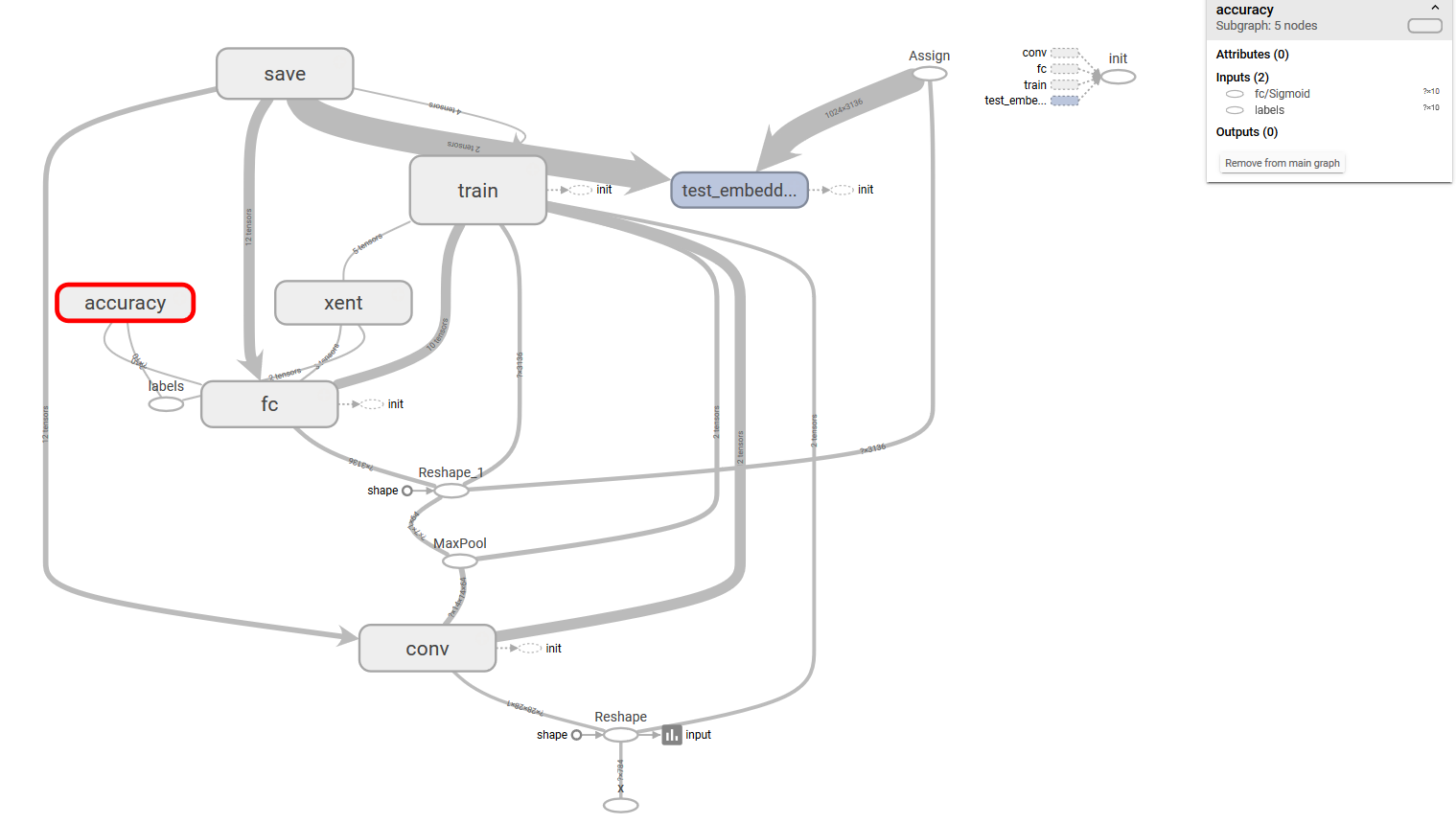
**Below are the collected execution times, accuracies, cross entropies and histograms**:

**Below table is summary of execution times, final accuracies and final cross entropies for 12 models:**

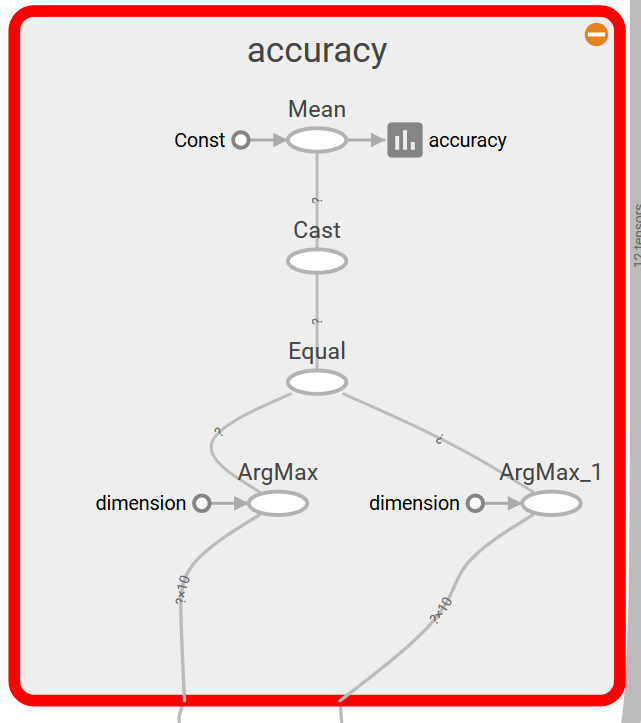




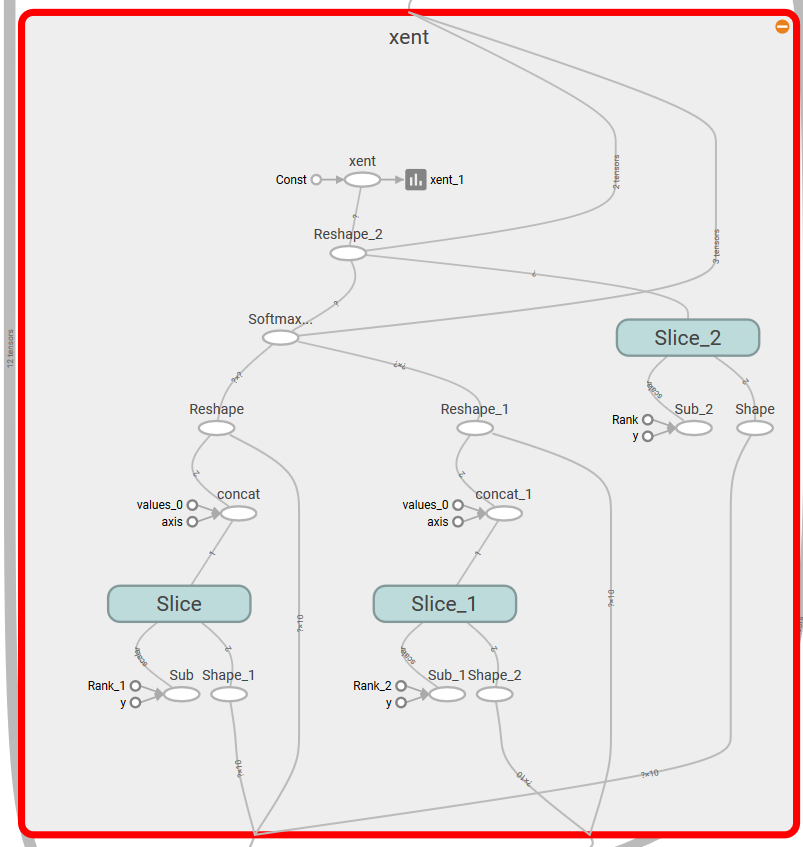
**Accuracy and Cross Entropy (summary) graphs:**

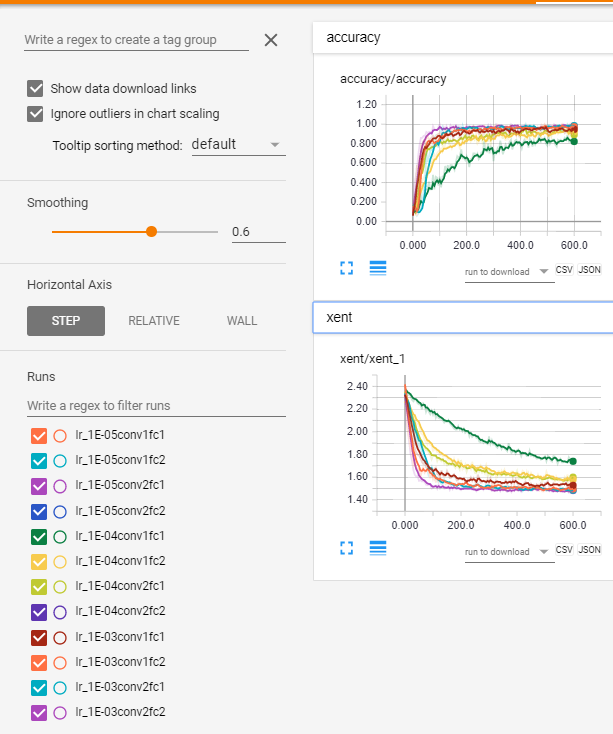


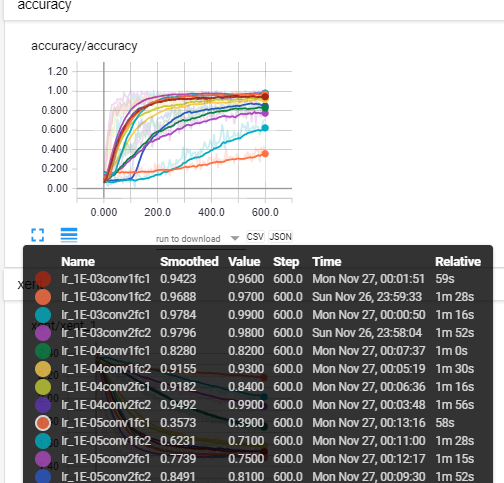
**Accuracy:**

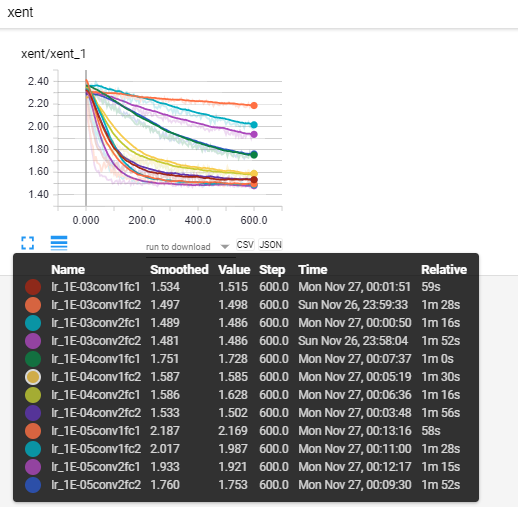


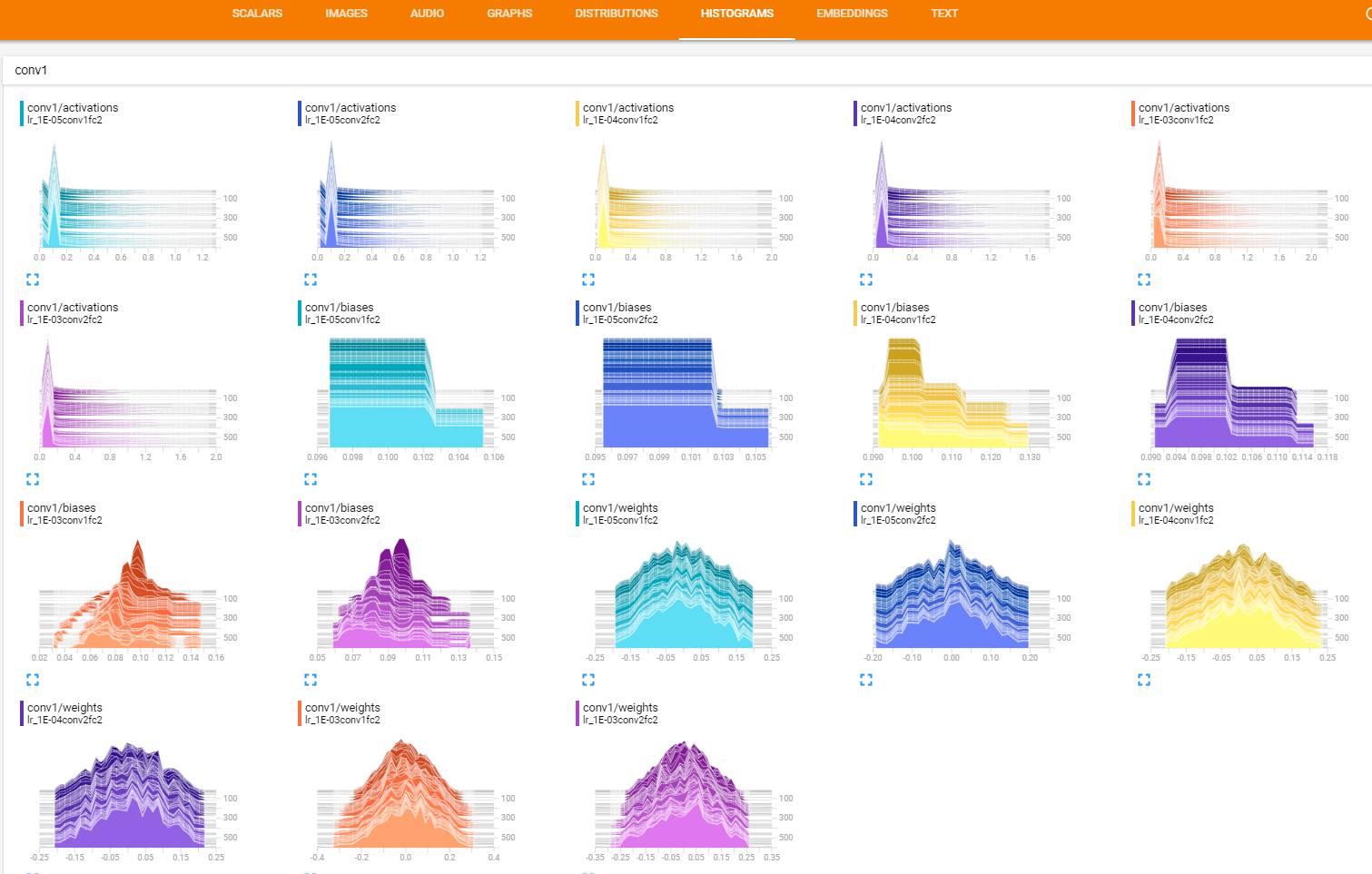
**Cross Entropy (xent):**

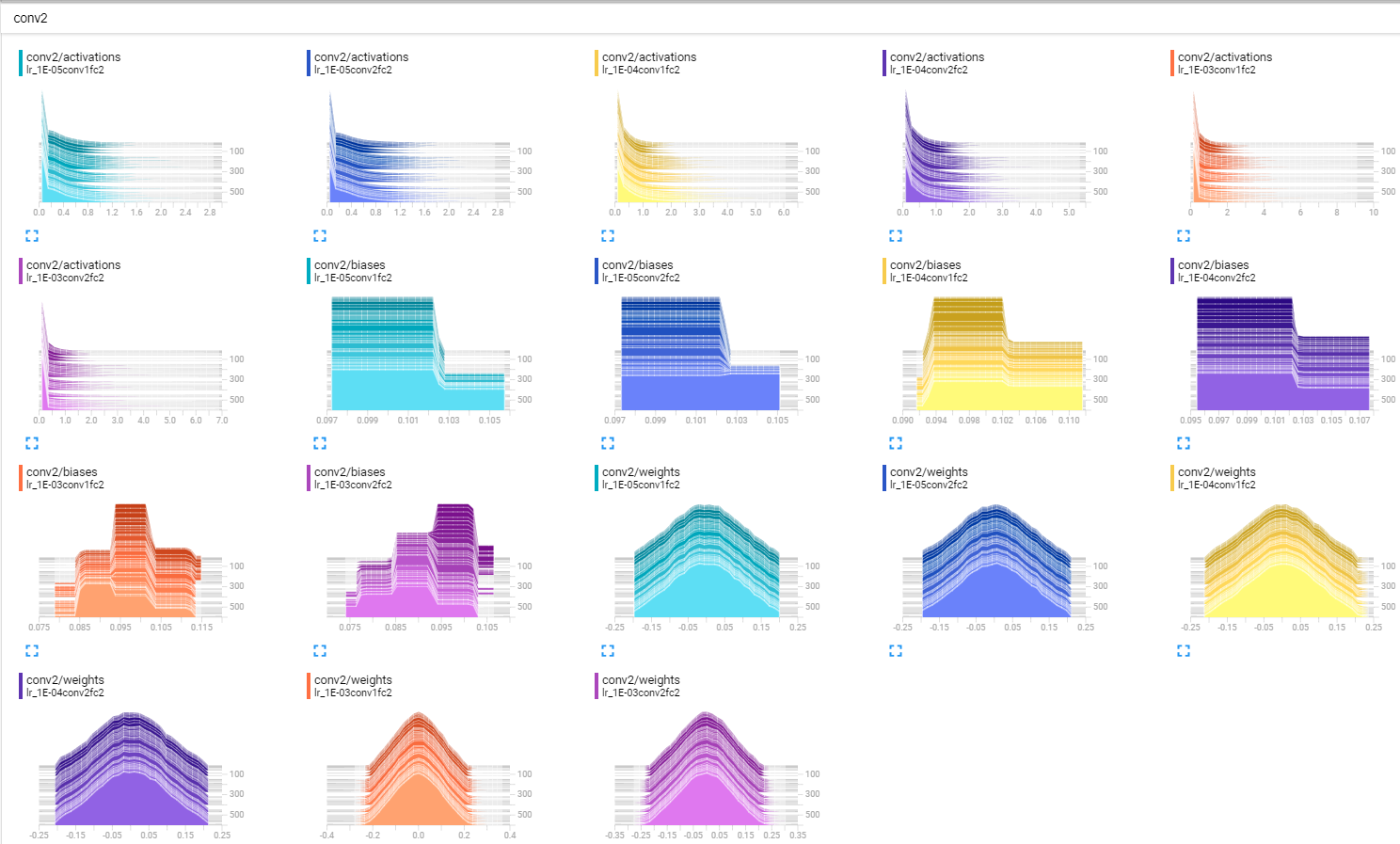


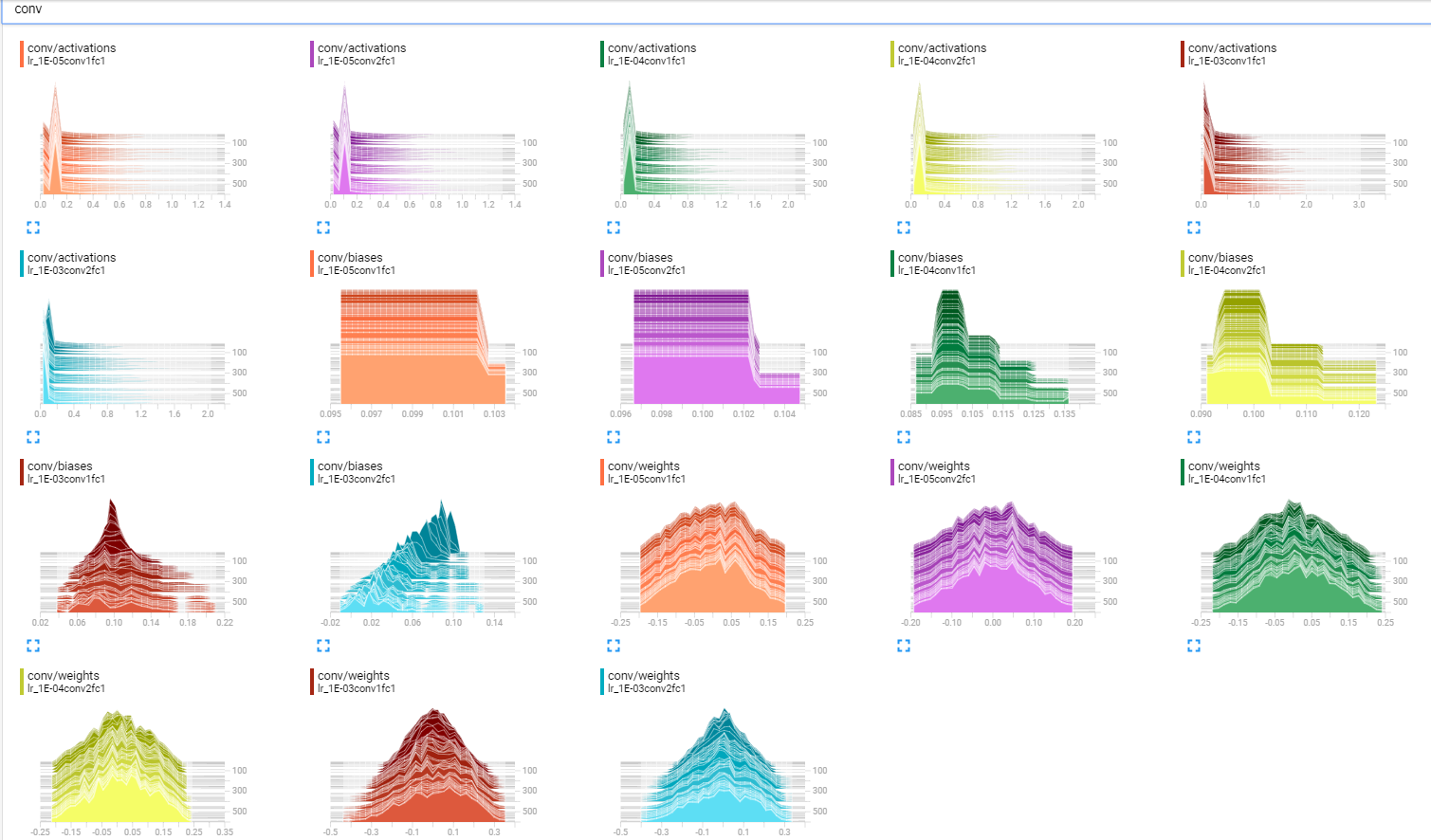


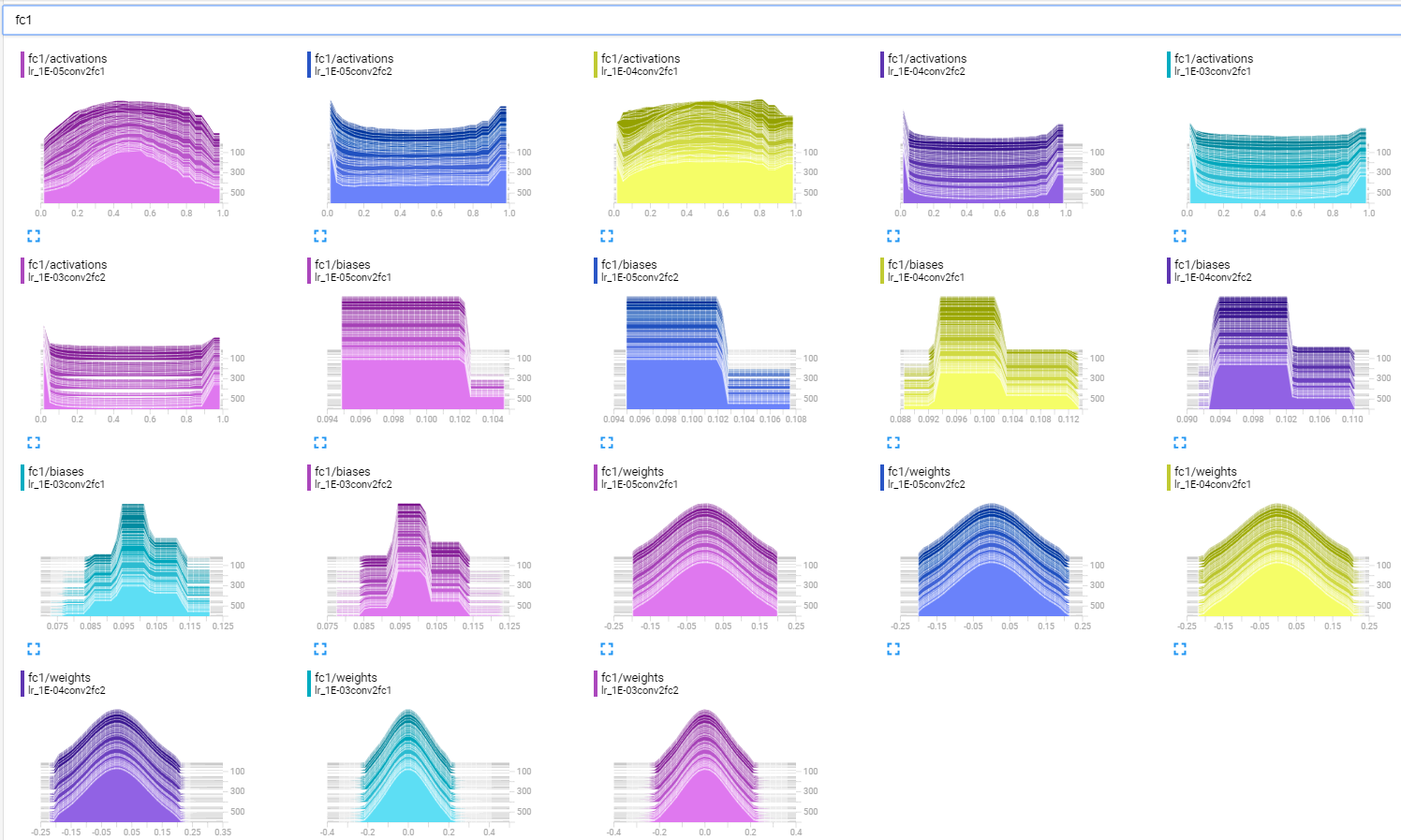




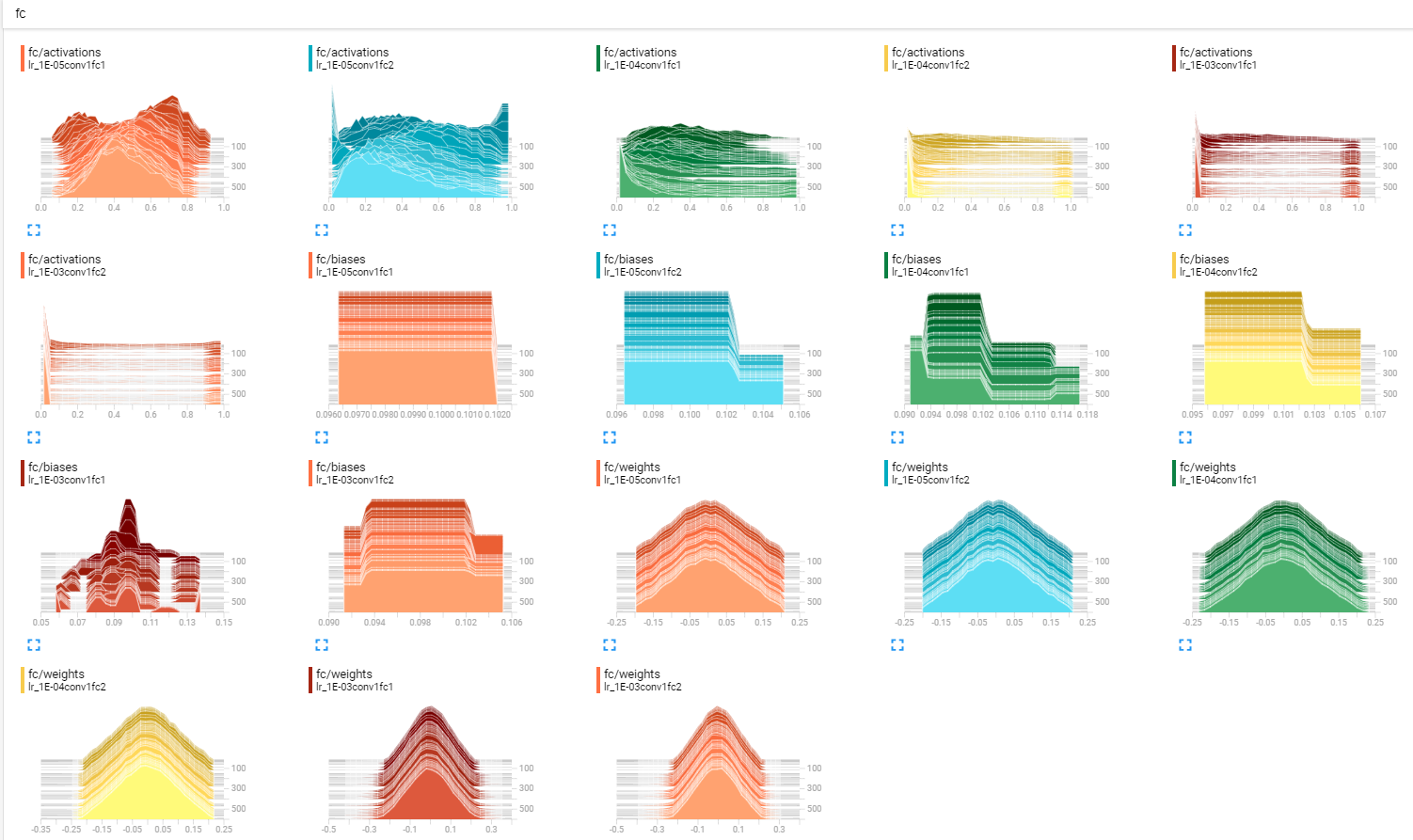


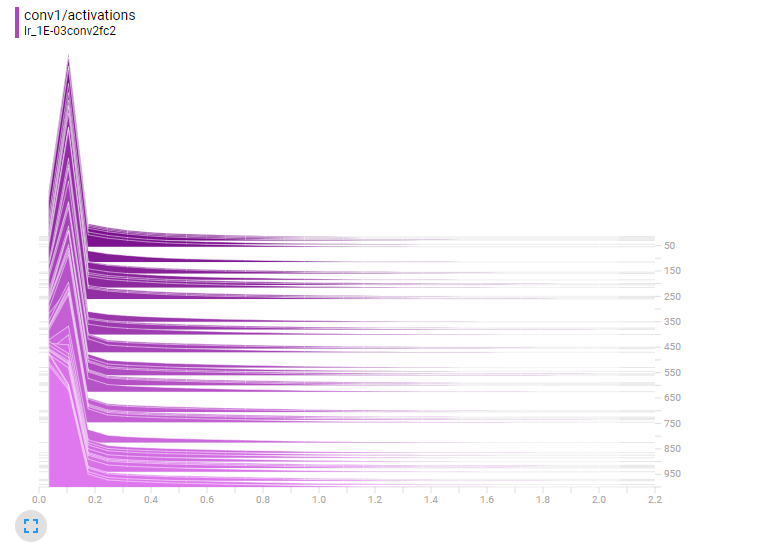


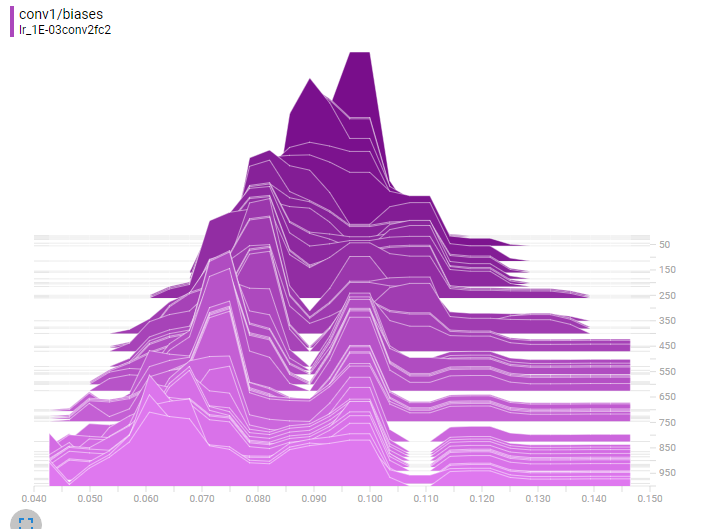


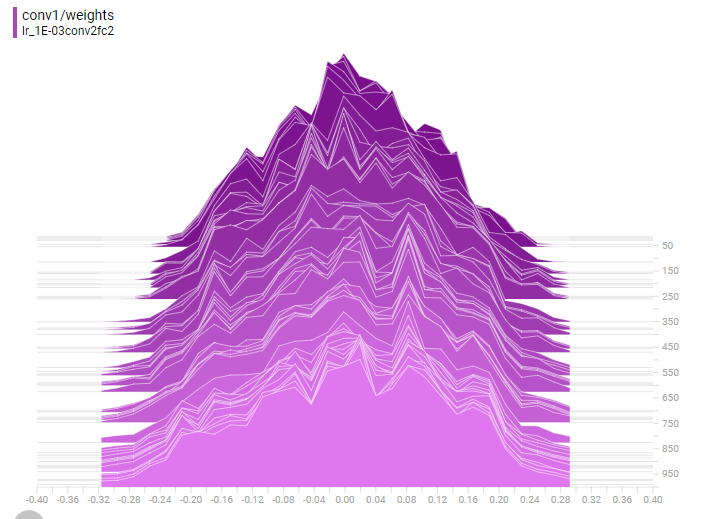


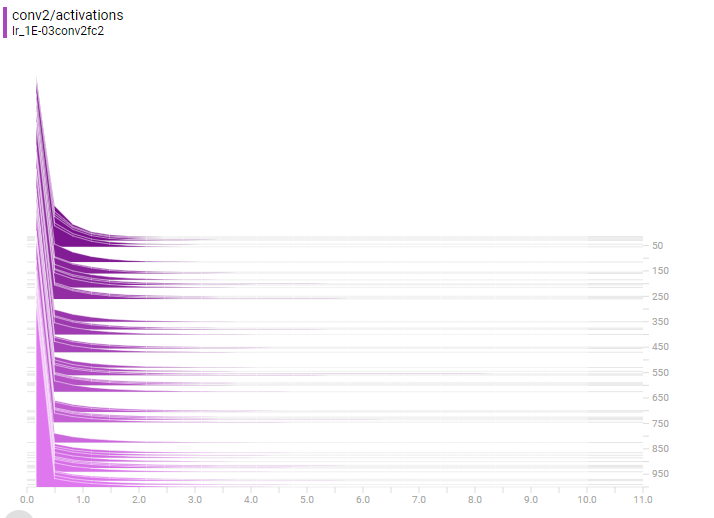


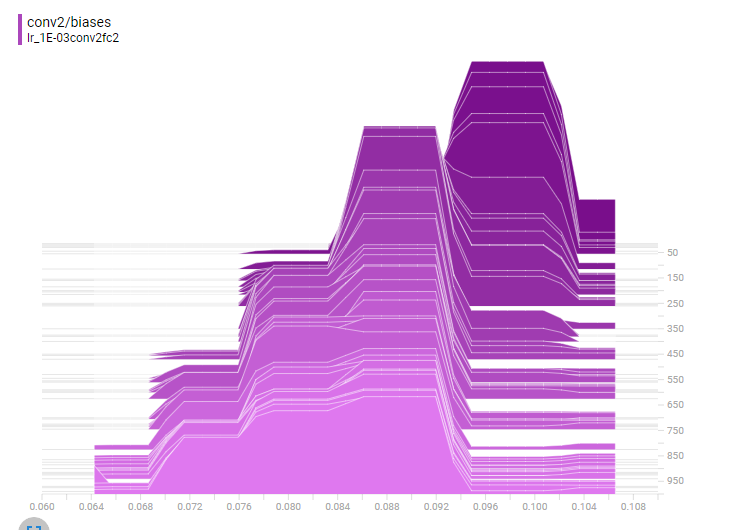


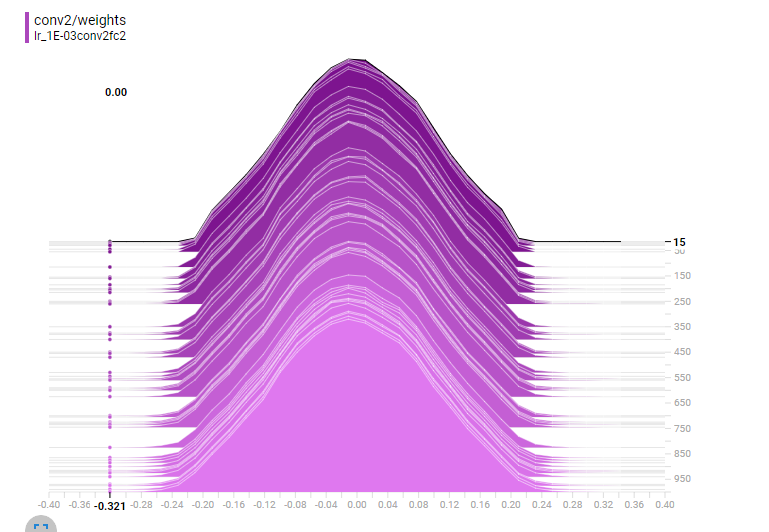


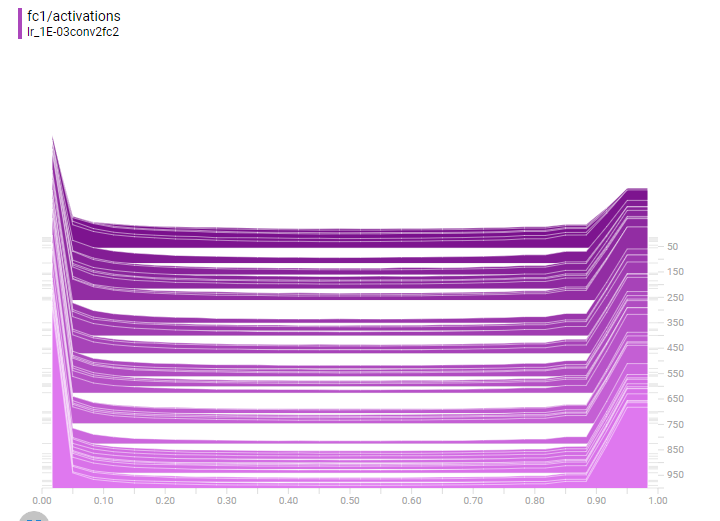


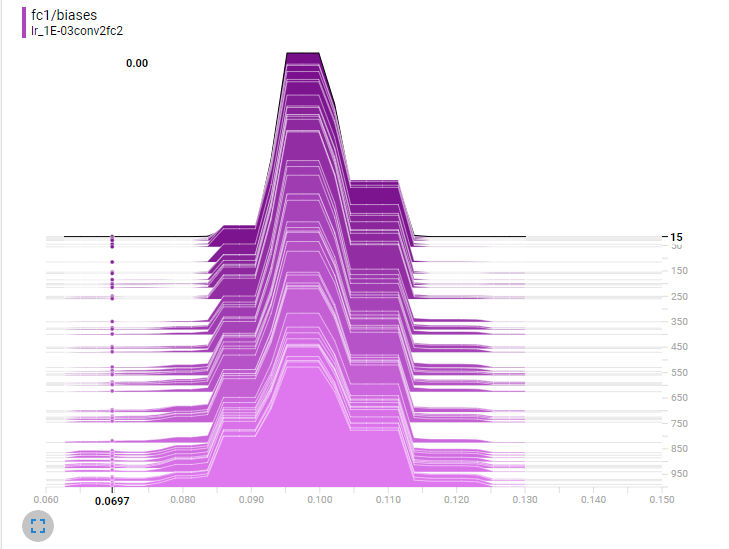


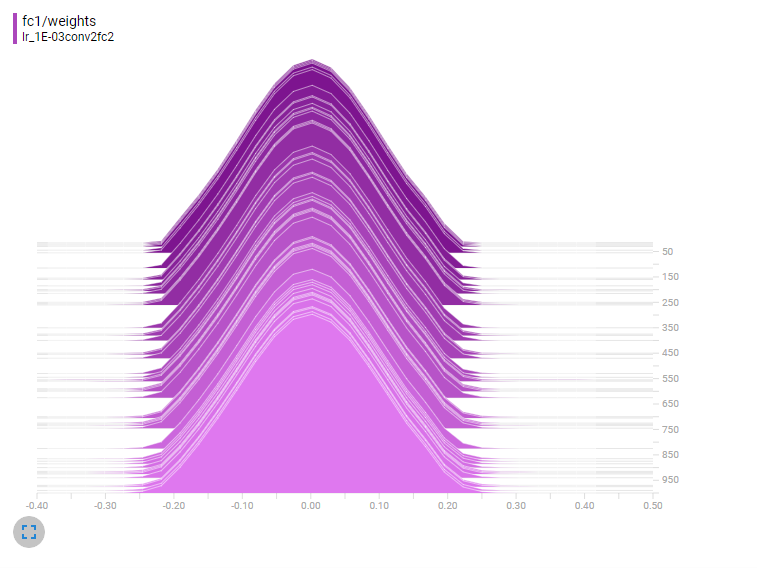


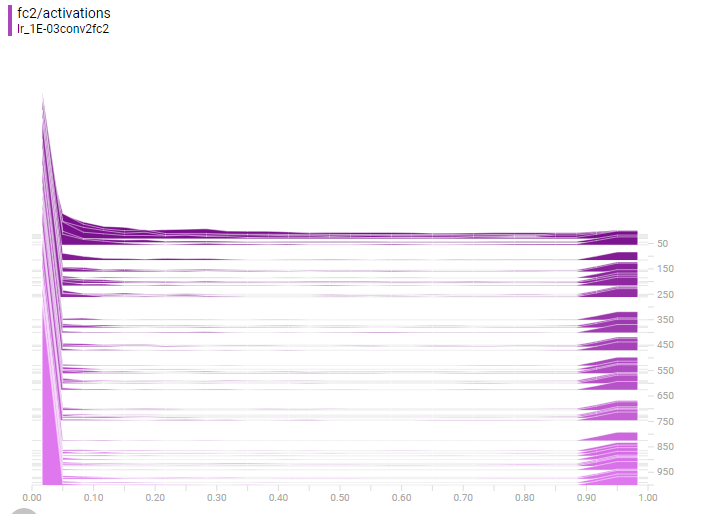


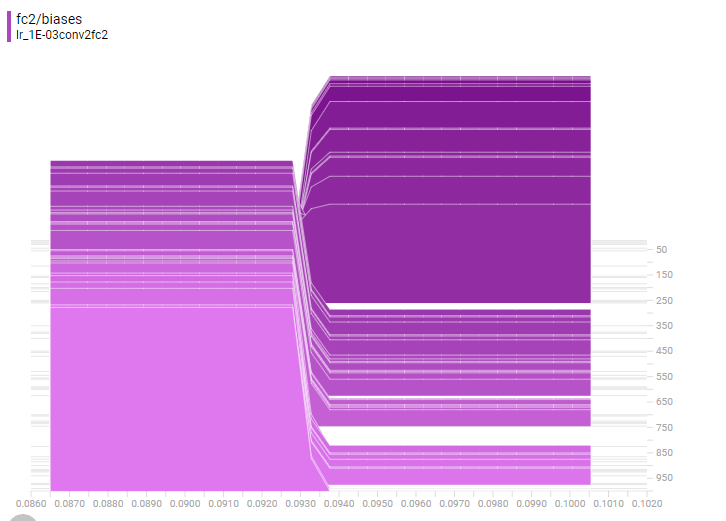


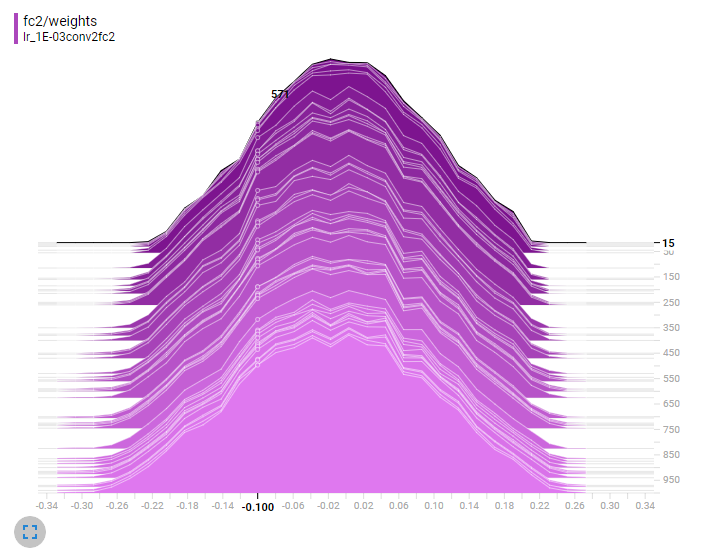












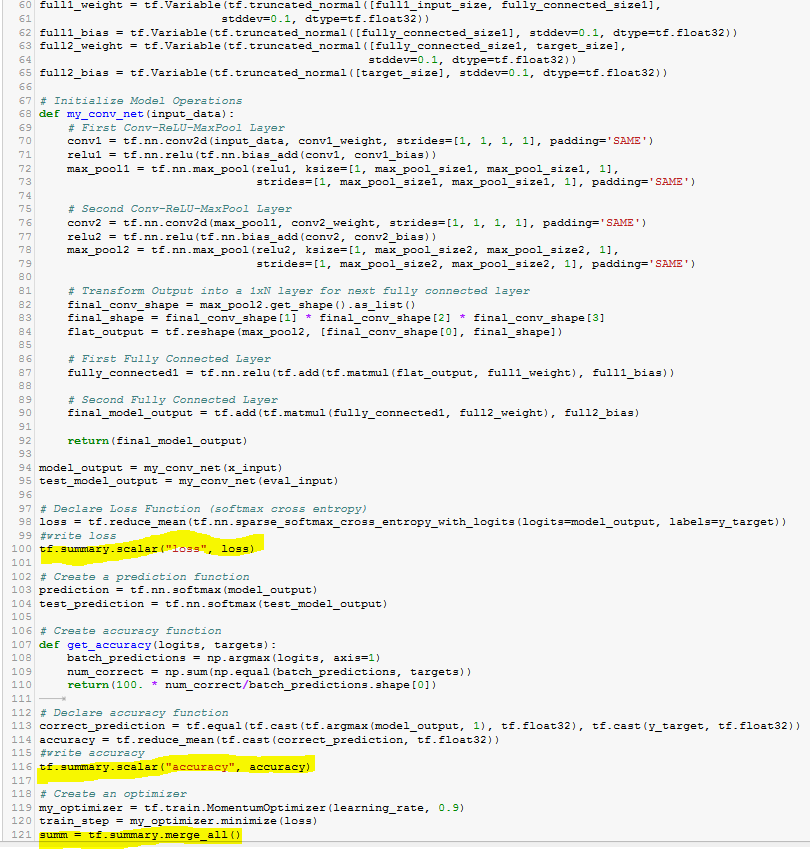
**Problem 3**. Modify file cnn\_mnist.py so that it publishes its summaries to the TensorBoard. Describe changes you are making and provide images of Accuracy and Cross Entropy summaries as captured by the Tensor Board. Provide the Graph of your model. Describe the differences if any between the graph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

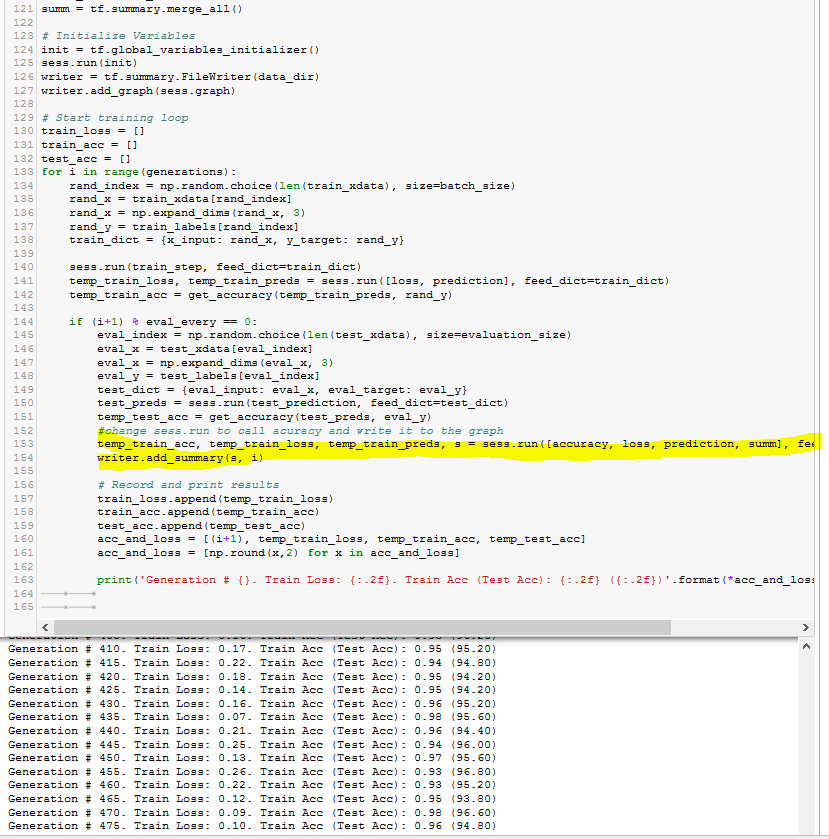
**Answer:**

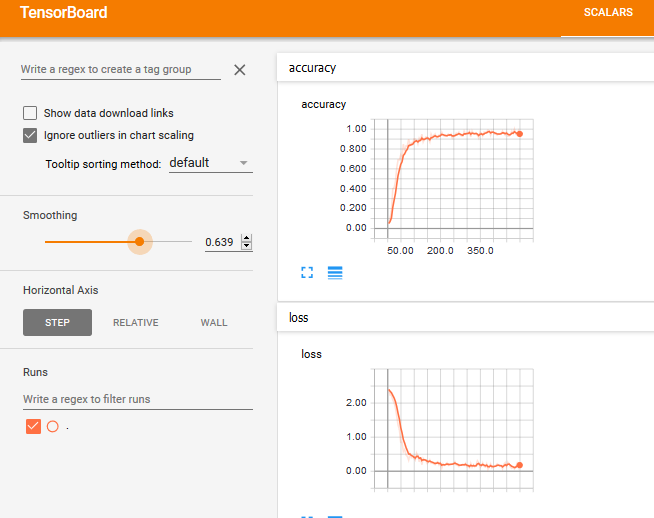
**Please see cnn\_mnist\_HW12P3.ipynb for modified/added in the code to graph the model. I also captured the file and yellow higlighted the code that I added.**

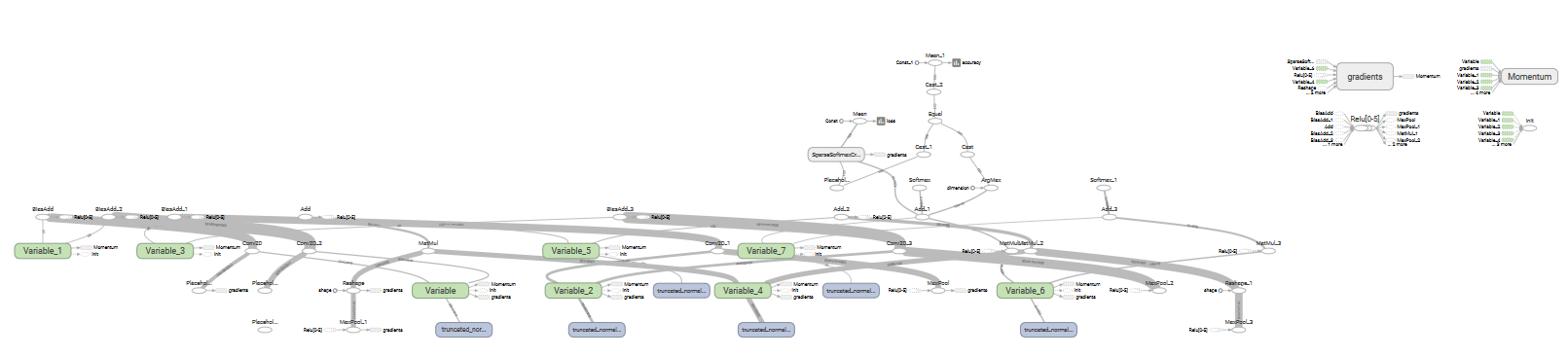
**The difference is the accuracy function, one has tf.cast and the other does not. During step ‘Add fully connected layer’, ‘mnist2’ uses act = tf.nn.relu(tf.matmul(input, w) + b) for activation. While ‘cnn\_mnist’ uses the sigmoid function for activation.**





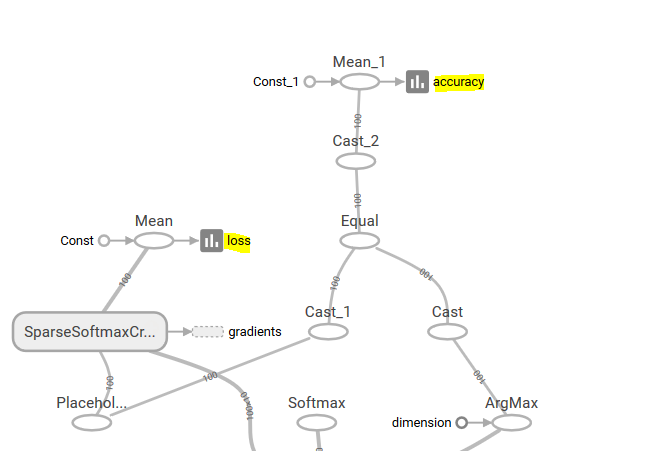


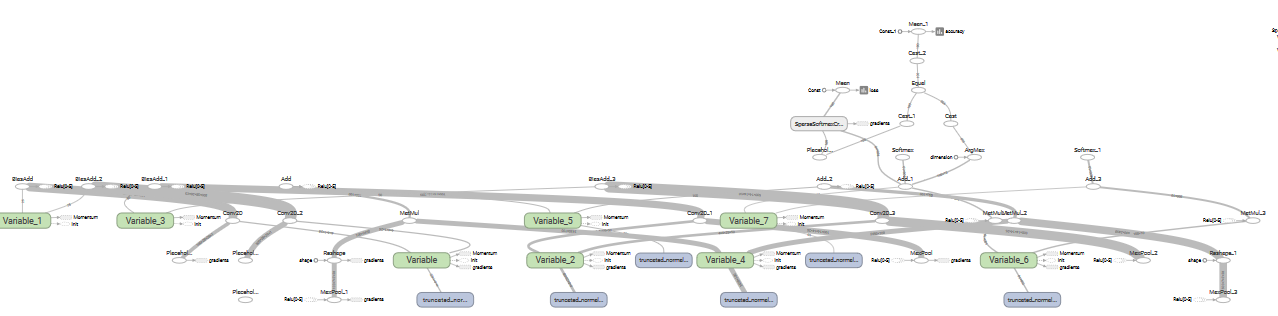




**Below is Zoomed in graph, I tried:**







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.