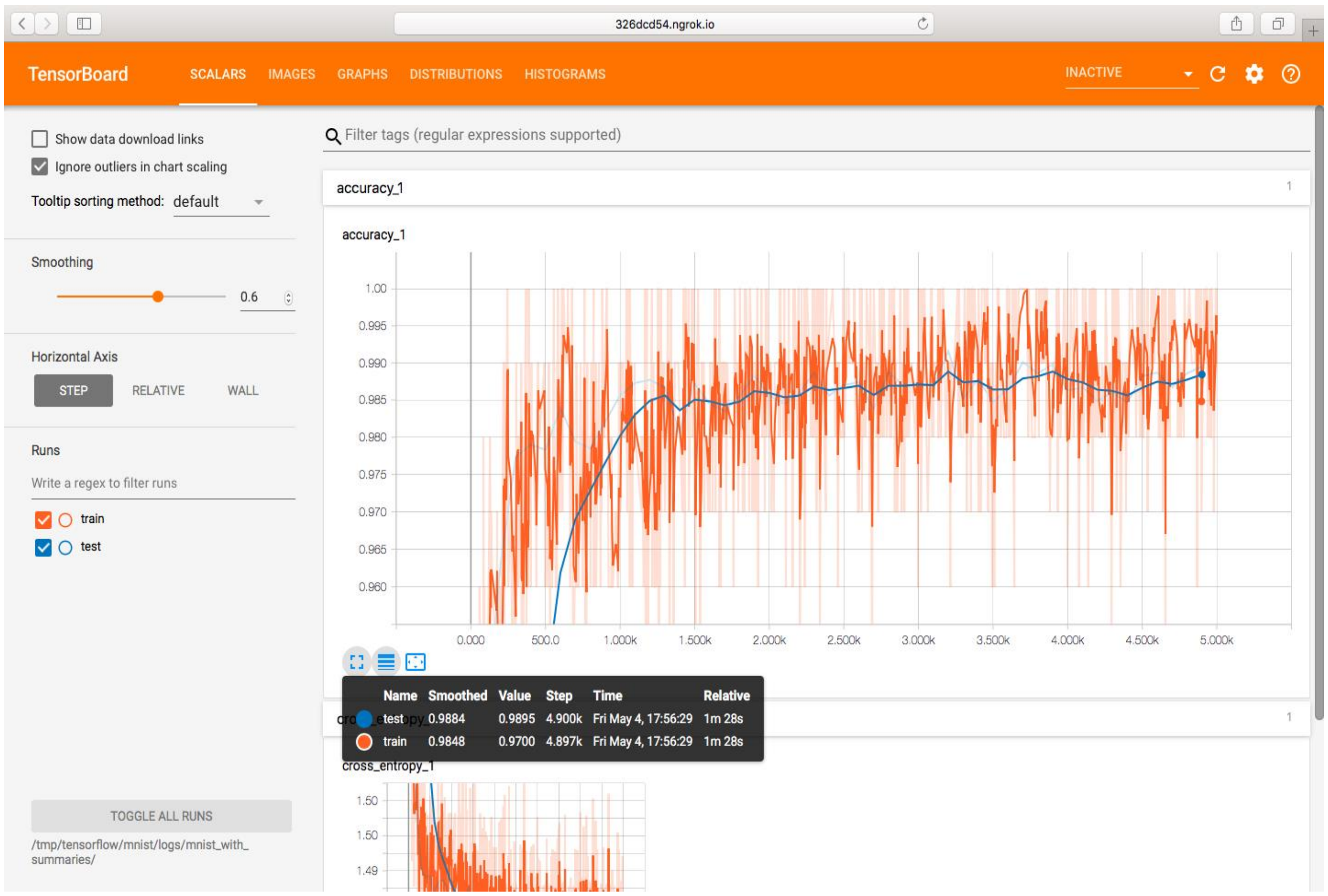
HOMEWORK 9

Problem 1(MNIST Baseline):

We used the MNIST and Tensor board tutorial and combined them so that we can train a network to classify the handwritten digits and also generate some summary from time to time. We ran the optimizer for 5000 batches and captured the accuracy on unseen(test) data at every 100 batch, this info was also written to the summary writer so that we can view this on tensor board graph. We managed to get an accuracy of **98.8%** on this baseline version. The architecture that we followed was as below:

We used 2 convolutional layers of depth 32 and 64 along with pooling performed on the output of each layer. This was then inputted to the dense layer after which it went to dropout section and finally we had the SoftMax classifier which brought down the values between 0 and 1. We used cross entropy as a measure of cost and this cost was minimized using Gradient Descent optimizer. We used the default values for learning rate and dropout as given in the tutorial. There were separate summary writers for both training and testing data so that we could visualize them together on tensor board.

Following is the accuracy plot we obtained:



☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP

☑ train

☑ test

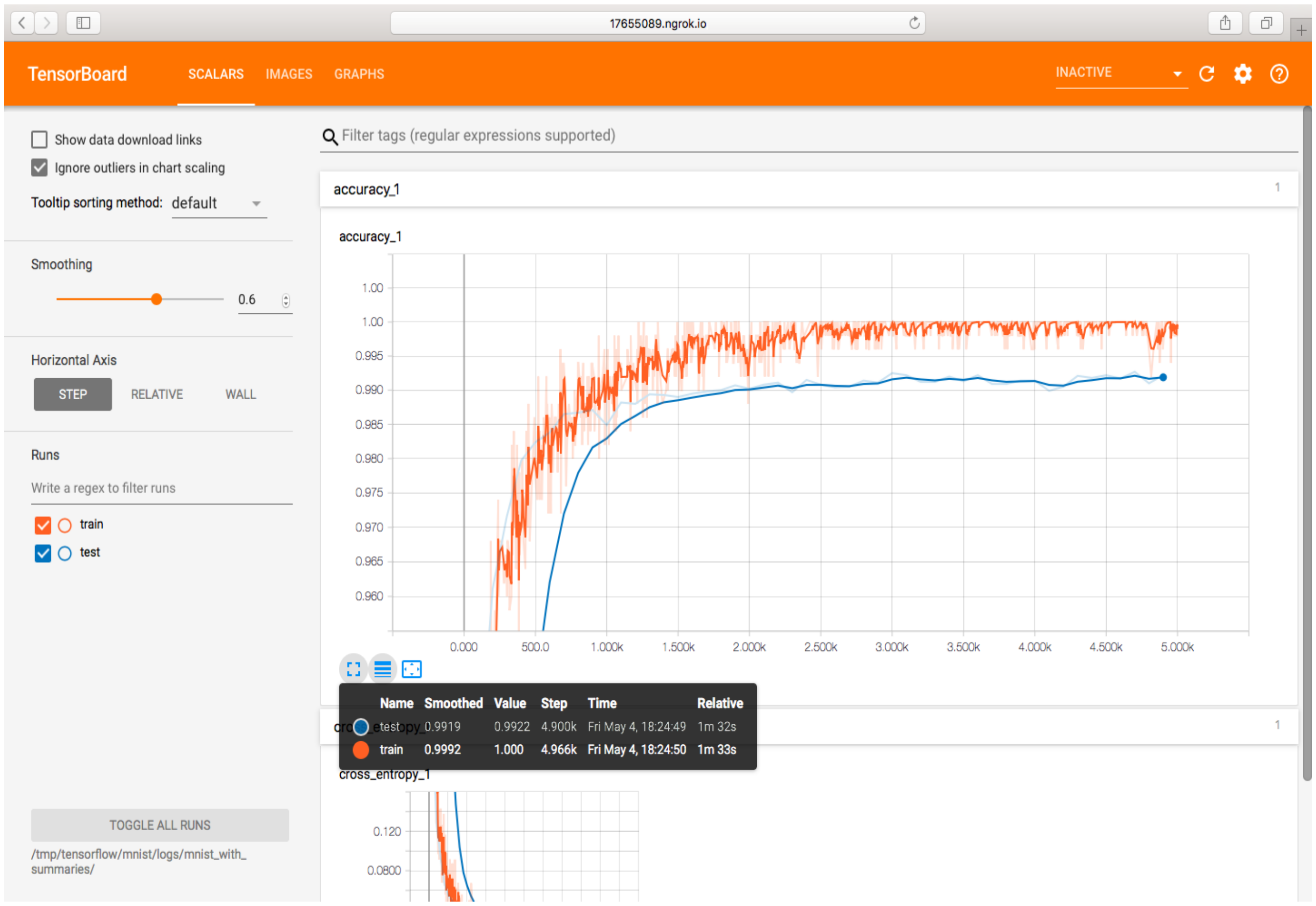
/tmp/tensorflow/mnist/logs/mnist_with_summaries/

	Name	Smoothed	Value	Step	Time	Relative
●	test	0.9884	0.9895	4.900k	Fri May 4, 17:56:29	1m 28s
●	train	0.9848	0.9700	4.897k	Fri May 4, 17:56:29	1m 28s

Problem 2(MNIST Modified):

We worked on the script that we created in the previous script to improve the accuracy of the classifier, we planned to modify the architecture of the network by doing the following changes:

We now used 3 convolutional layer of depth 8, 16 and 1024 followed by a dropout, we thought that changing the depth might give the network to adjust for more weights at each layer. The rest of the things were same as above with slight changes as follows we used cross entropy for cost, used Adam Optimizer for reducing cost(optimization), we adjusted the learning rate a bit but the dropout rate was kept the same. After doing all these changes we again monitored the accuracy saw a bump in the rate and now we had accuracy equal to **<u>99.22%</u>** and following is the plot:



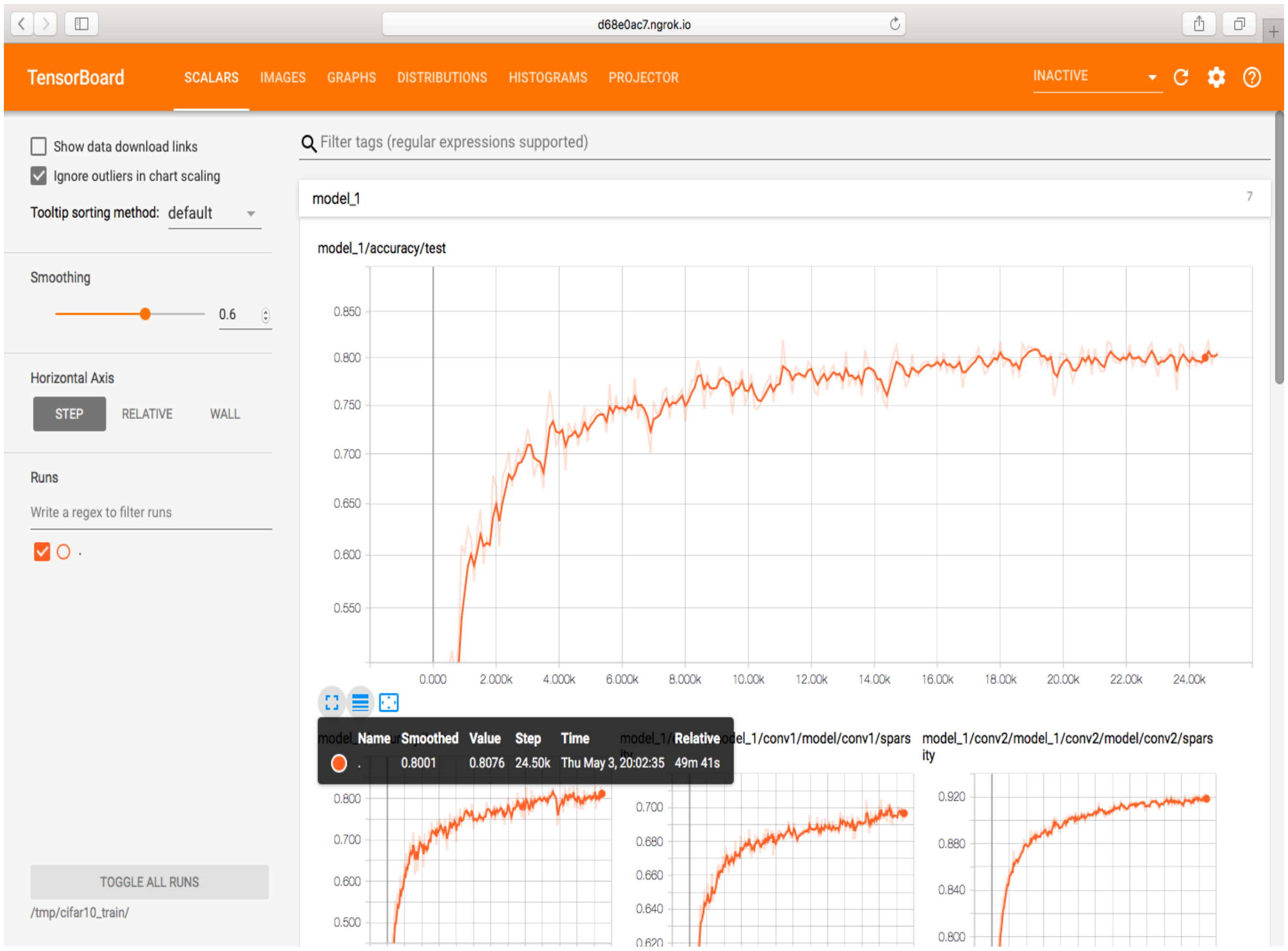
Show data download links: ☐
Ignore outliers in chart scaling: ✓
Tooltip sorting method: default
Smoothing: 0.6
Horizontal Axis: [✓ STEP] RELATIVE WALL
Runs: ✓ train, ✓ test

	Name	Smoothed	Value	Step	Time	Relative
●	test	0.9919	0.9922	4.900k	Fri May 4, 18:24:49	1m 32s
●	train	0.9992	1.000	4.966k	Fri May 4, 18:24:50	1m 33s

<u>Problem 3(CIFAR10 Baseline):</u>

We went through the cifar10 tutorials provided in the instructions and tried understanding the workflow and the logic was splitted across 4 files and the training and evaluation script had to run at the same time to get the accuracy plotted as the model got trained. Since the requirement was to log the accuracy on test data for every 100 batch we first decided to modify the code to create checkpoints after every 100 batch and make the eval script to pick these checkpoints and calculate the accuracy. Although this worked reasonably well but was still an overhead to run 2 scripts parallel. Finally, after doing some modification in the train script we managed to generate the accuracy results for both test and train data and log that to summary writer. The accuracy that we achieved with the baseline version was **80.01%** with the architecture as follows:

There were 2 convolutional layers of depth 64 along with pooling and normalization done at each layer after which there was a SoftMax function. Default values were used for the hyper parameters like log accuracy, batch size. We ran this for 25k steps and log the accuracy for every 100th run. The accuracy plot is as below:

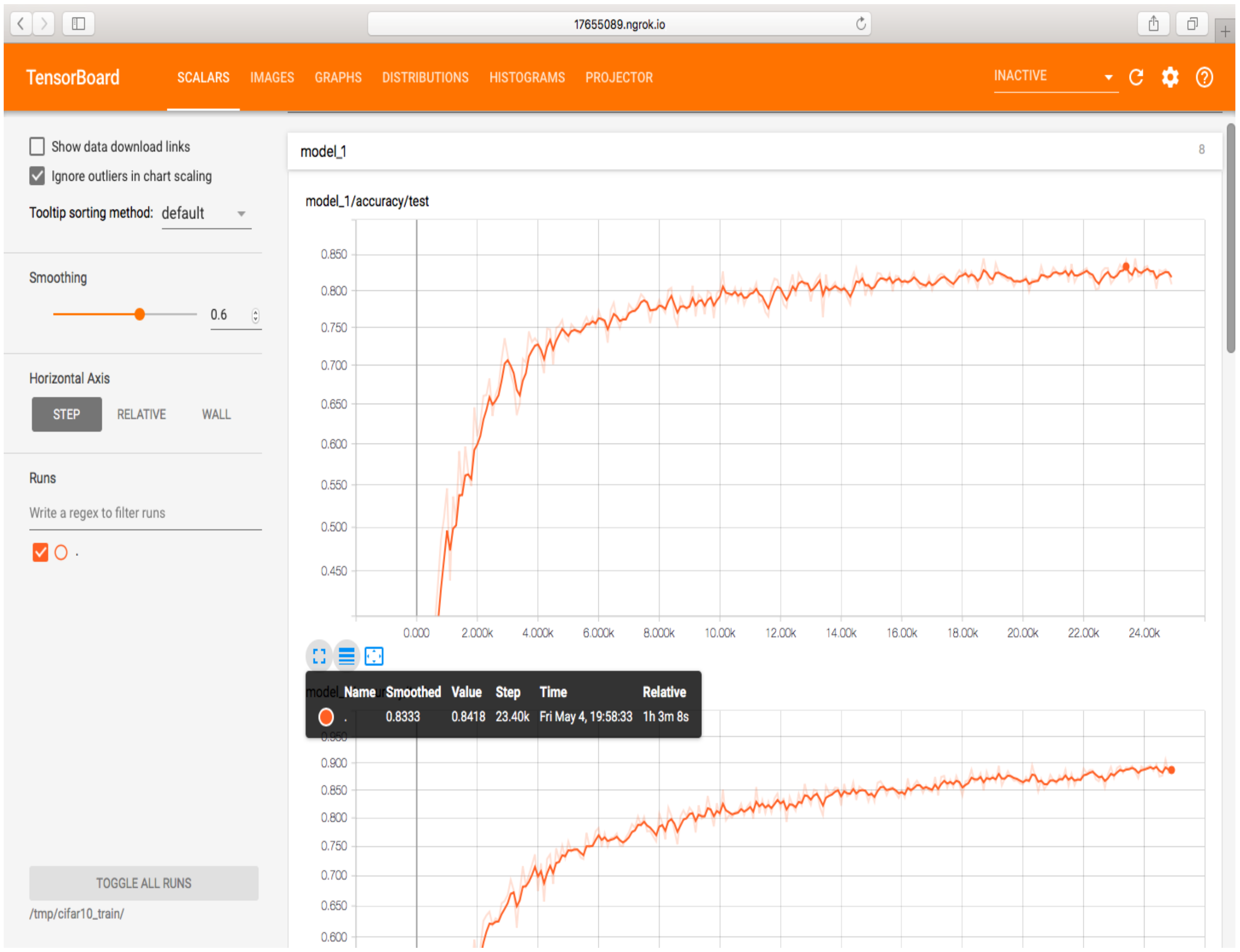


Name	Smoothed	Value	Step	Time	Relative
.	0.8001	0.8076	24.50k	Thu May 3, 20:02:35	49m 41s

Problem 4(CIFAR10 Modified):

We worked on the script which we created above and planned to modify some architecture of the network which is as follows:

We now had 3 convolutional layers but with depth 32, 64, 1024 along with pooling and normalization implemented at each layer. We thought that changing the depth will help adjust for the no of parameters that have to be modified. The output of this was fed to SoftMax layer. We used the default value of learning rate and batch size was kept to 128. We ran this for 25k steps and logged the accuracy for every 100 batch. The accuracy we achieved was **84.18%** which was almost 4% increase from the baseline one. The graph is as follow:



Name	Smoothed	Value	Step	Time	Relative
.	0.8333	0.8418	23.40k	Fri May 4, 19:58:33	1h 3m 8s

<u>CITATIONS</u>**:** We looked at the tensor board tutorials, also some reading about neural network. We looked out for some of the tensor board functionality out there on stack overflow.