ELEC 576 / COMP 576 Fall 2020 : Assignment1

Mst Afroja Akter

October 2020

1 Backpropagation in a Simple Neural Network

(a). Dataset

We will use make-moon dataset from Scikit-learn. This dataset has two two classes (e.g. "male" and "female"). Figure 1 is the visualization of this dataset.

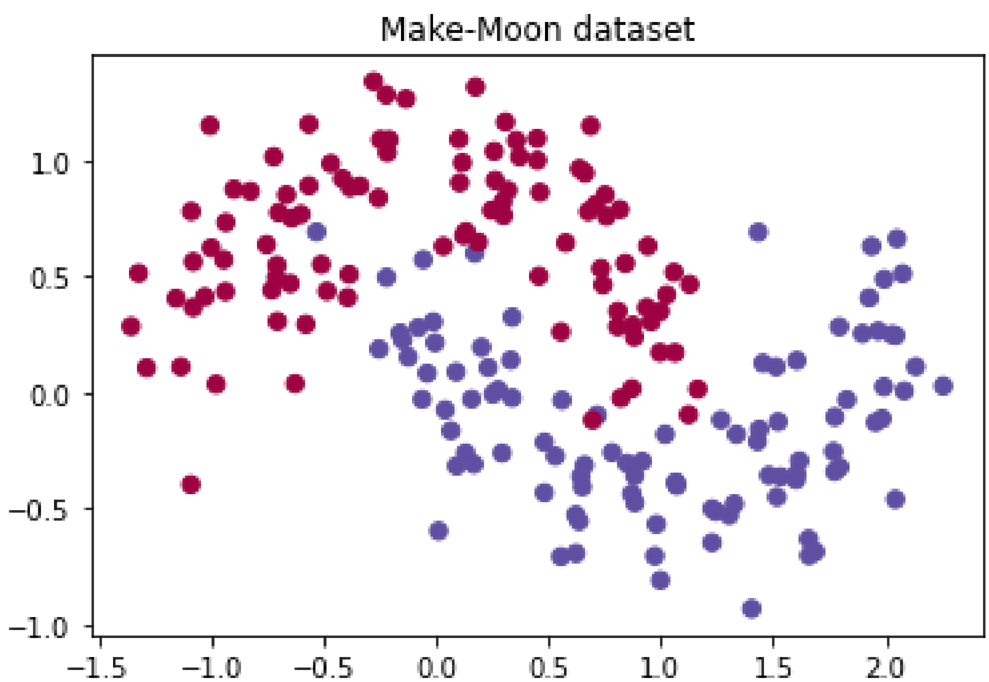


Figure 1: Make-Moon dataset

Activation Function

This part of the problem is in the code three-layer-neural-network.py We derived three activaiton functions 'Tanh', 'sigmoid' and 'ReLU' in the function actFun(self, z, type) and and their derivative in the fucntion diff-actFun.

(c) Build the Neural Network

The Neural Network has three layers: one input layer, one hidden layer and one output layer. The number of neurons per layer are 2, 4, 2 respectively. The input vector of the network is denoted by X and the output/target vector is denoted by y. The output layer has softmax activation function. For this network we used Entrophy loss function

$$L(y, \hat{y}) = -\frac{1}{N}\Sigma_{n\in N}\Sigma_{i\in C}y_{n,i}\log\hat{y}_{n,i}$$

Here, y are one-hot-encoded vectors and $\hat{y}$ are vectors of probabilities.

(e)

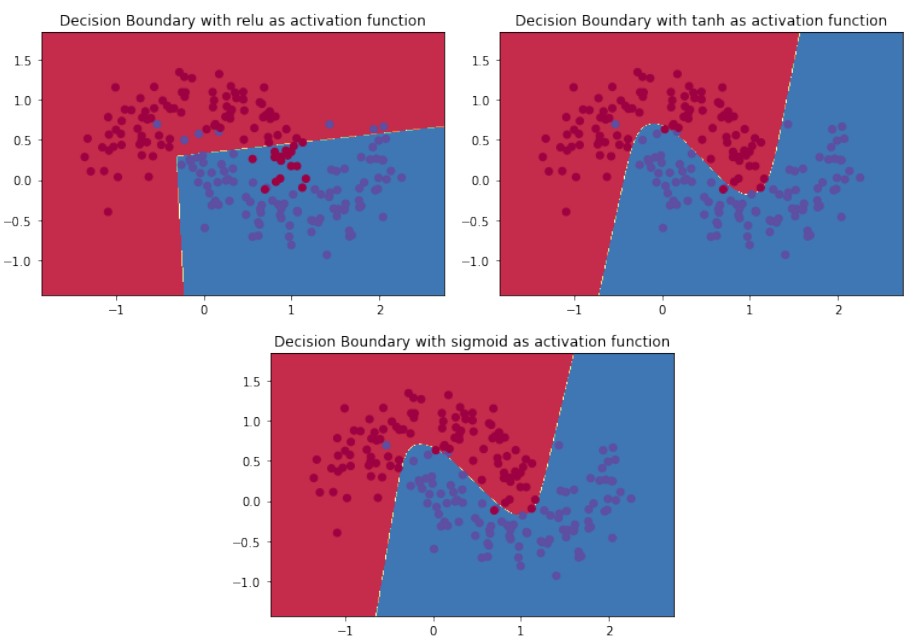


Figure 2: Decision Boundary with 'ReLU', 'tanh' and 'sigmoid' activation functions

The figure 2 shows the decision boundary plot with 'ReLU', 'tanh' and 'sigmoid' activation functions. Figure 3 shows the respective loss function keeping the netwrok architecture same. Here we see that decision boundary with 'tanh' and 'sigmoid' performs better than the decision boundary with 'ReLU' activation function. The loss function with relu activation function is unstable, which relates to the bad performance of decision boundary.

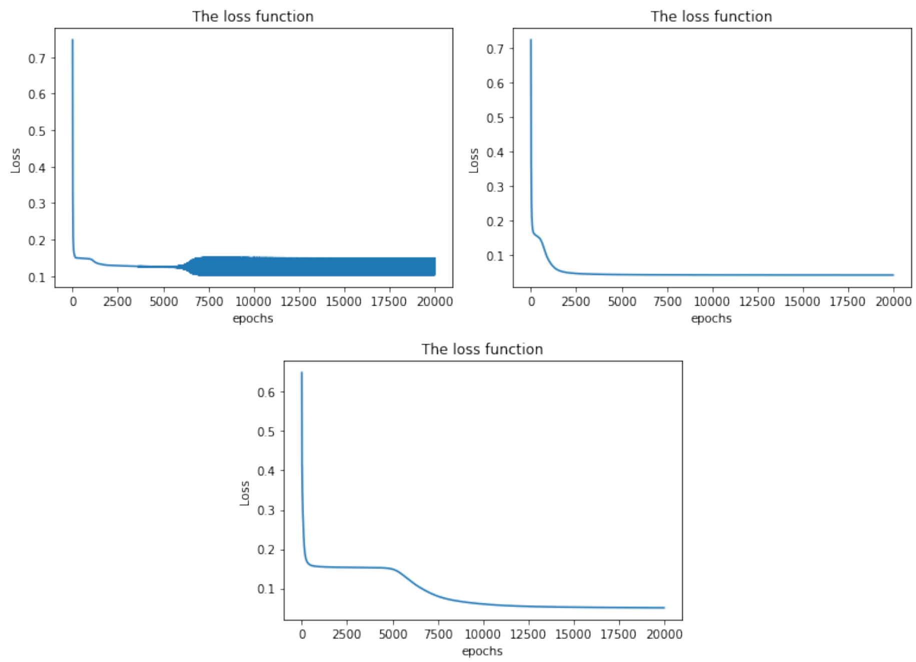


Figure 3: The Loss with 'ReLU', 'tanh' and 'sigmoid' activation functions

Figure 4 shows with higher number of neurons in the hidden layer leads to overfitting of the decision boundary. One obvious reason this overfitting could be due to the memorization of training data. In this situation neural network model performs better in training data and performs poorly in the test data.

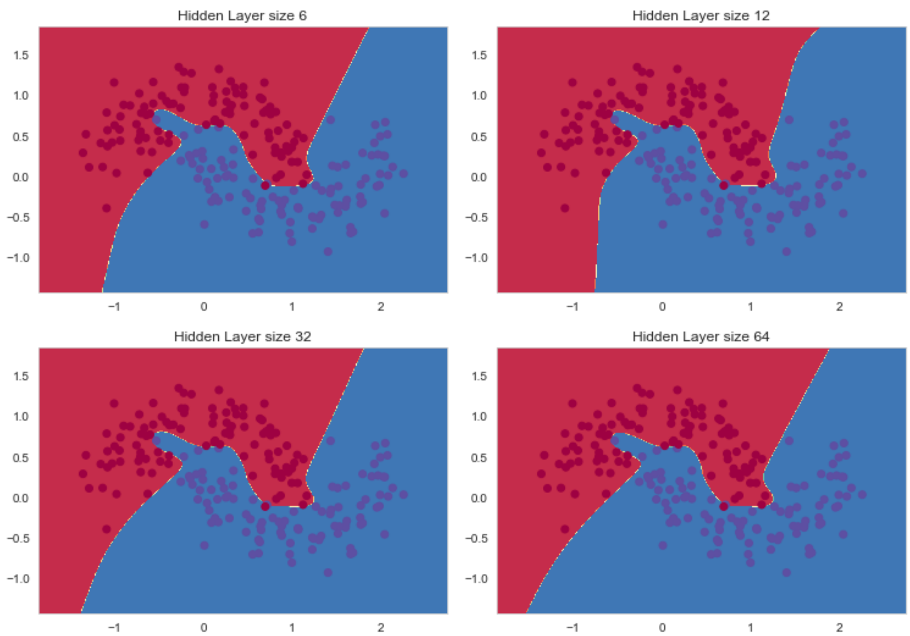


Figure 4: Decision Boundary with 6, 12, 32 and 64 hidden layers and 'tanh' activation function

(f)Deep Neural Network[1]

Result in figure 5 generated with a deep neural network with two hidden layers and 3 neurons in each hidden layer. The input layer has two neuron and the output layer has two neuron. I used sigmoid activation function with learning rage 0.05 in 1000 iteration. I tried with higher iteration number but I couldn't get better result.

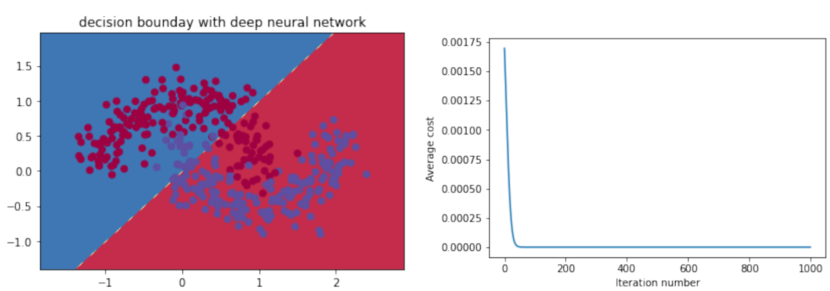


Figure 5: Decision boundary with deep neural network

[1]I didn't use Class from the 'three layer neural network: I am not that comfortable yet with classes.

2 Training a Simple Deep Convolutional Network on MNIST

The simple deep convolutional network architecture:

conv1(5 - 5 - 1 - 32) - ReLU - maxpool(2 - 2) - conv2(5 - 5 - 32 - 64) - ReLU - maxpool(2 - 2) - fc(1024) - ReLU - DropOut(0.5) - Softmax(10)

(a)

Figure 6 and 7 shows the training loss and training accuracy of the simple deep convolutional network with the above mentioned architecture. The loss curves from figure 8 shows that the validation and test loss follows the training loss and the same for accuracy curves. This is a good indication of Network learning.

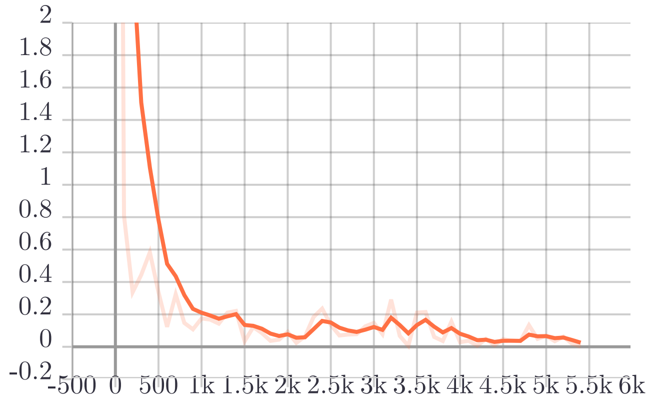


Figure 6: Train Loss

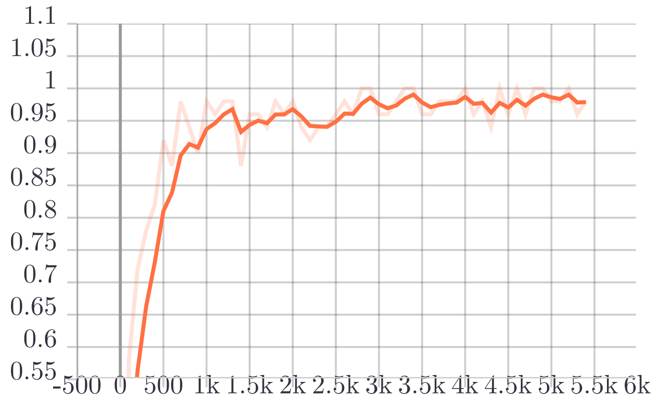


Figure 7: Train Accuracy

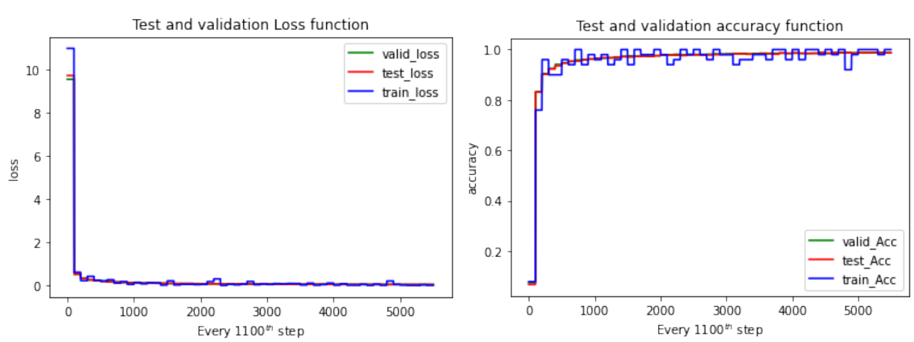
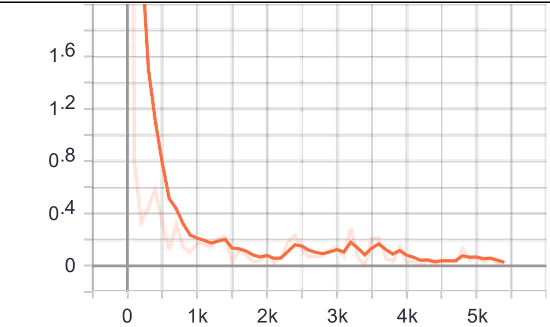
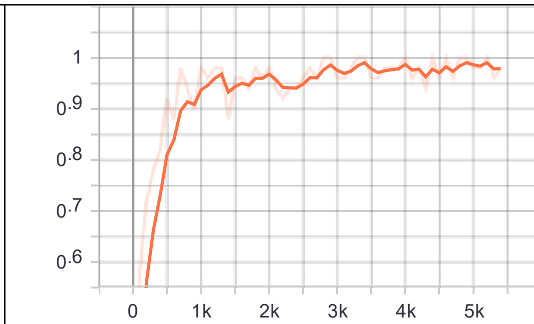


Figure 8: The loss and accuracy plots of training, validation and test

(b). The figures from Tensorboard:

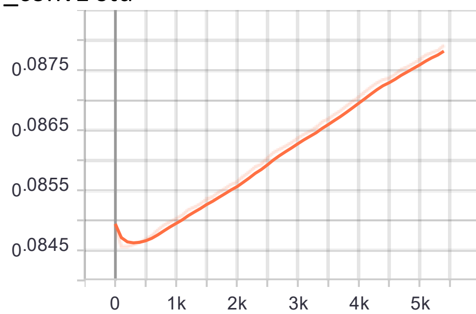


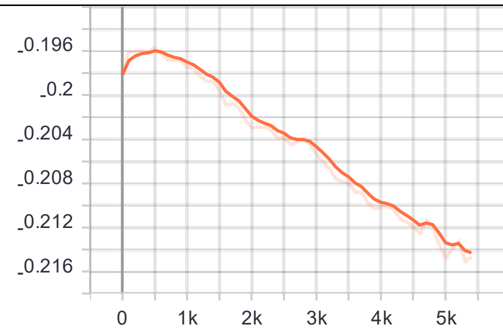
Train loss



Train accuracy

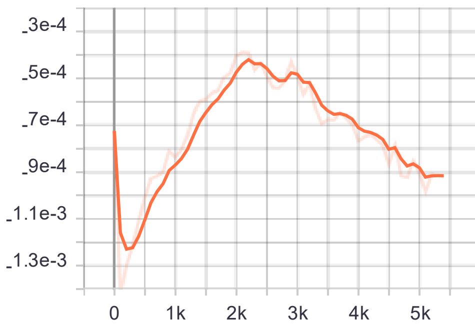
W_conv1 std



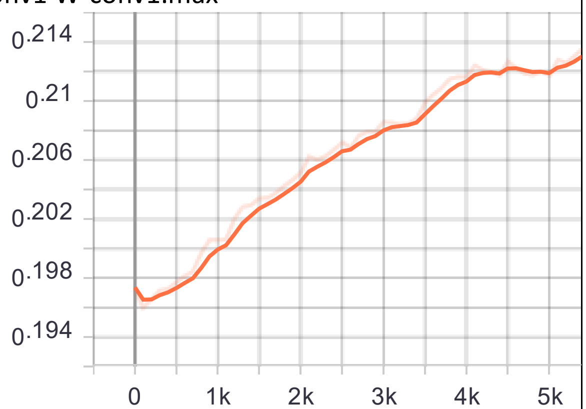


W_conv1 min

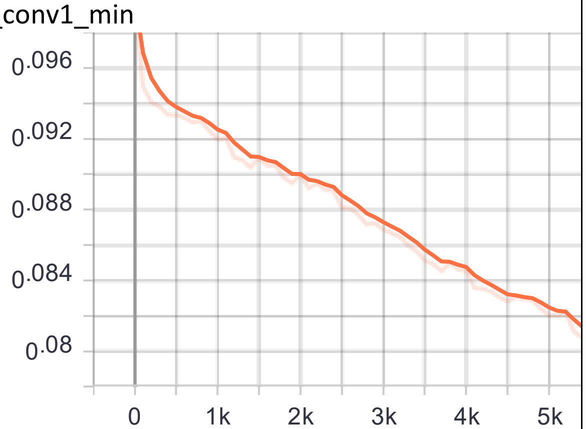
W_conv1 mean



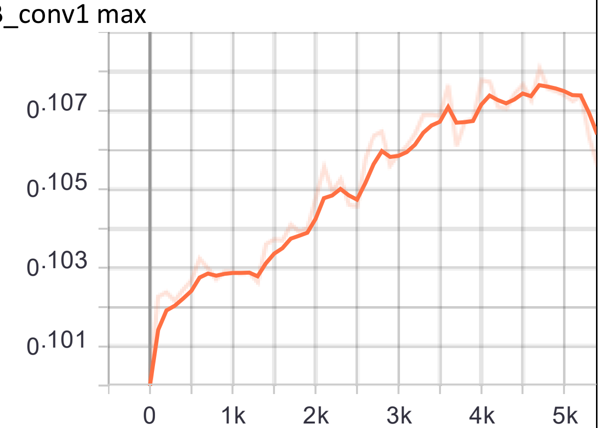
Conv1 W-conv1.max

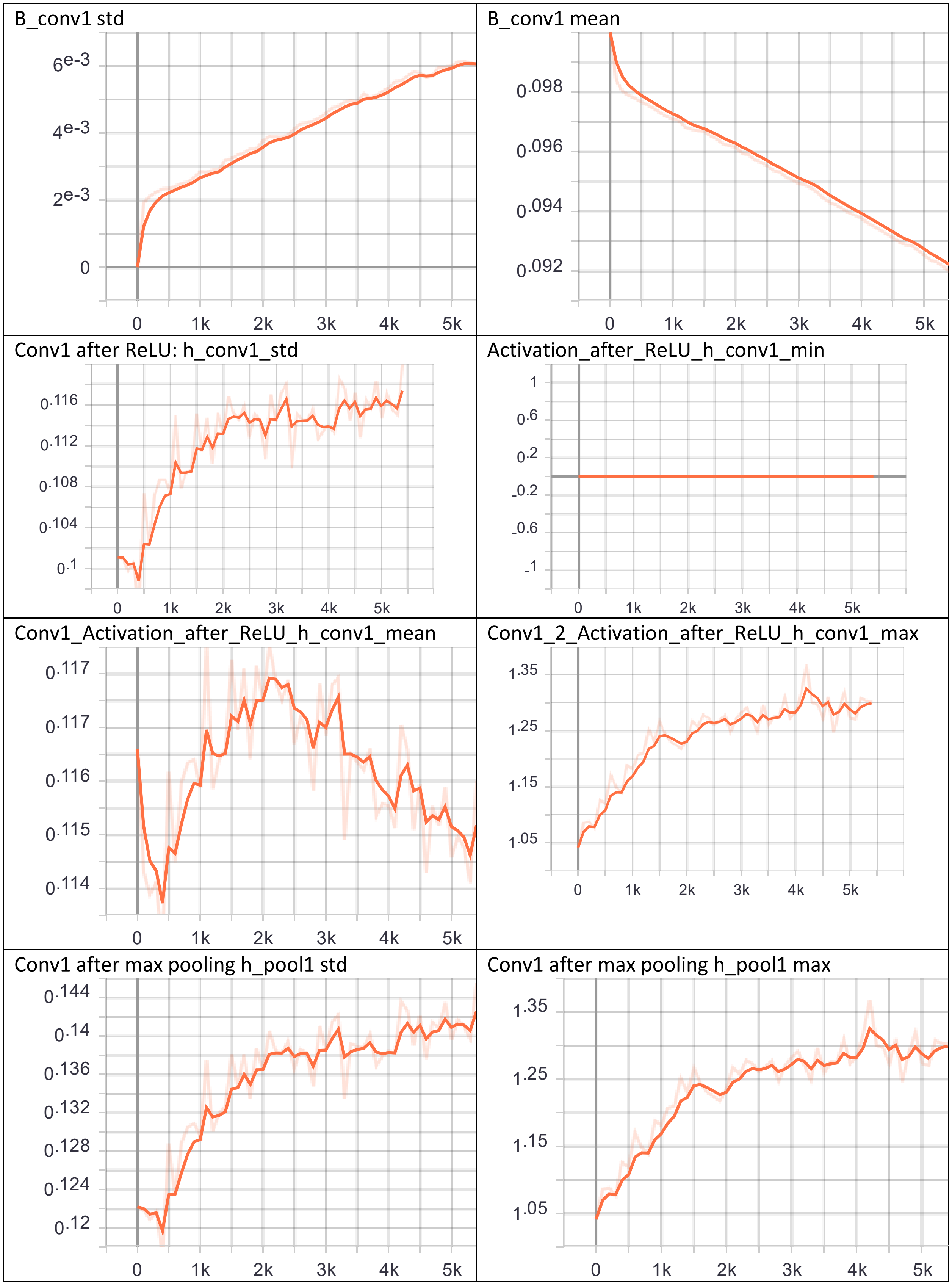


b_conv1_min

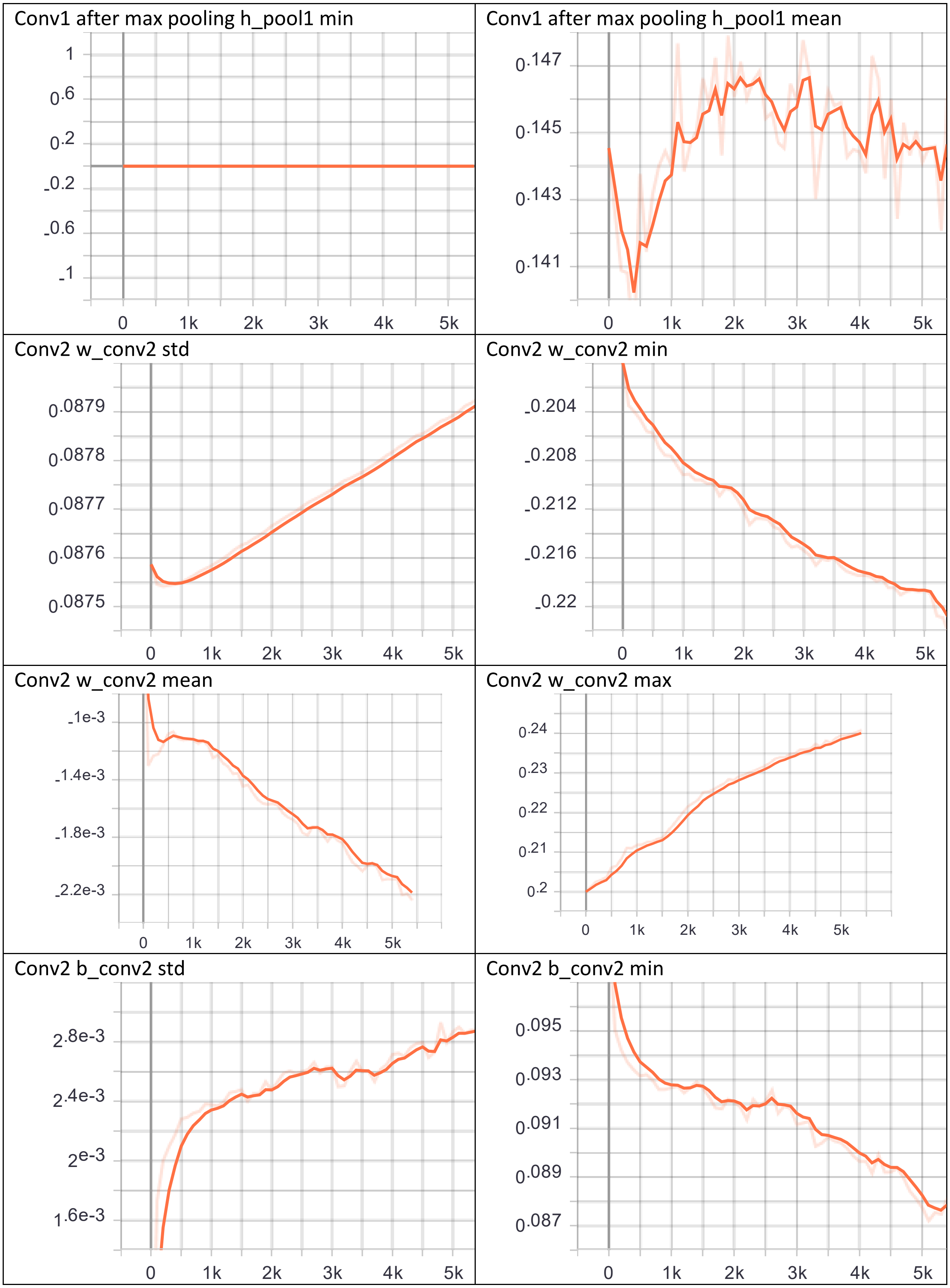


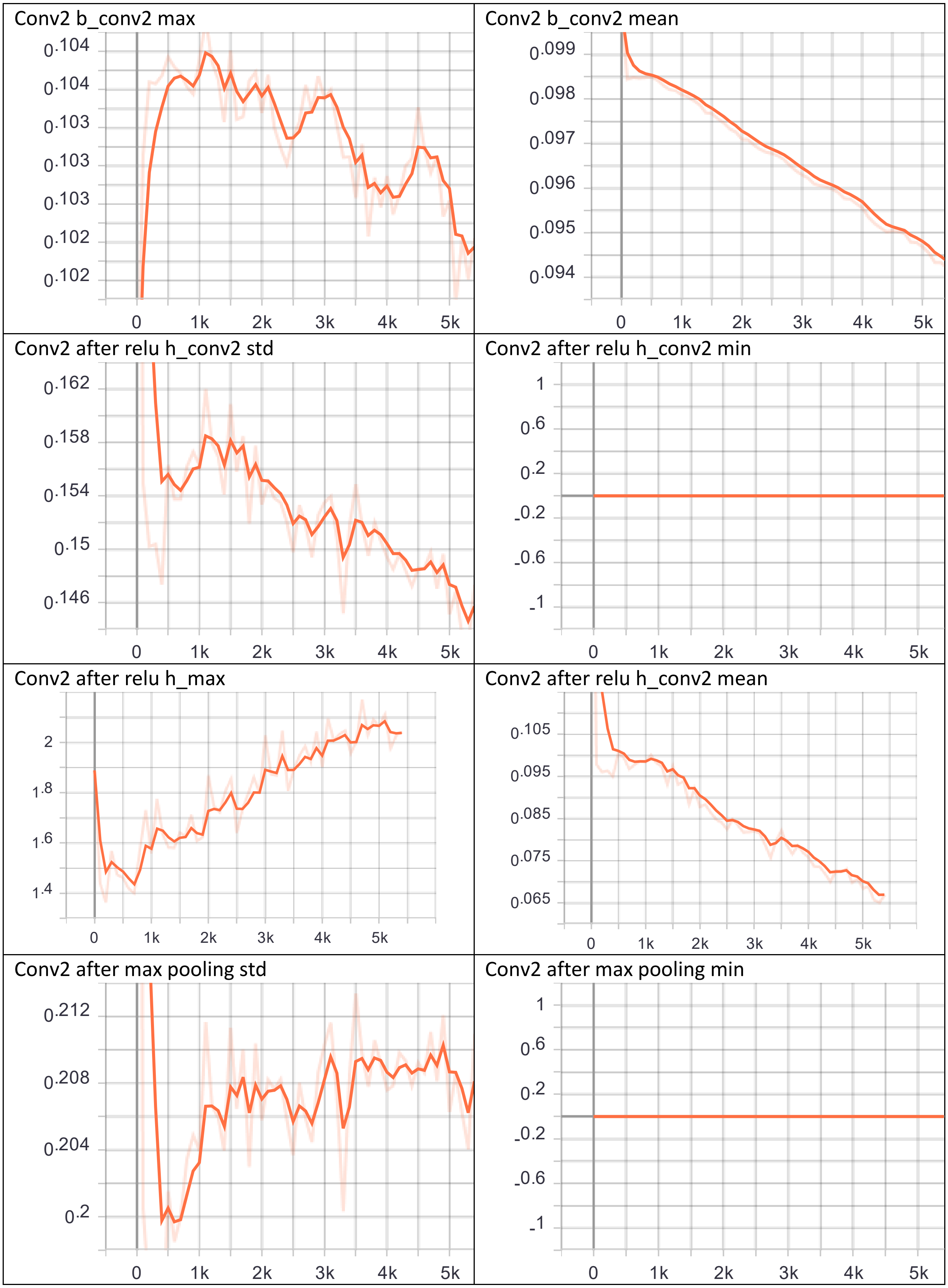
B_conv1 max

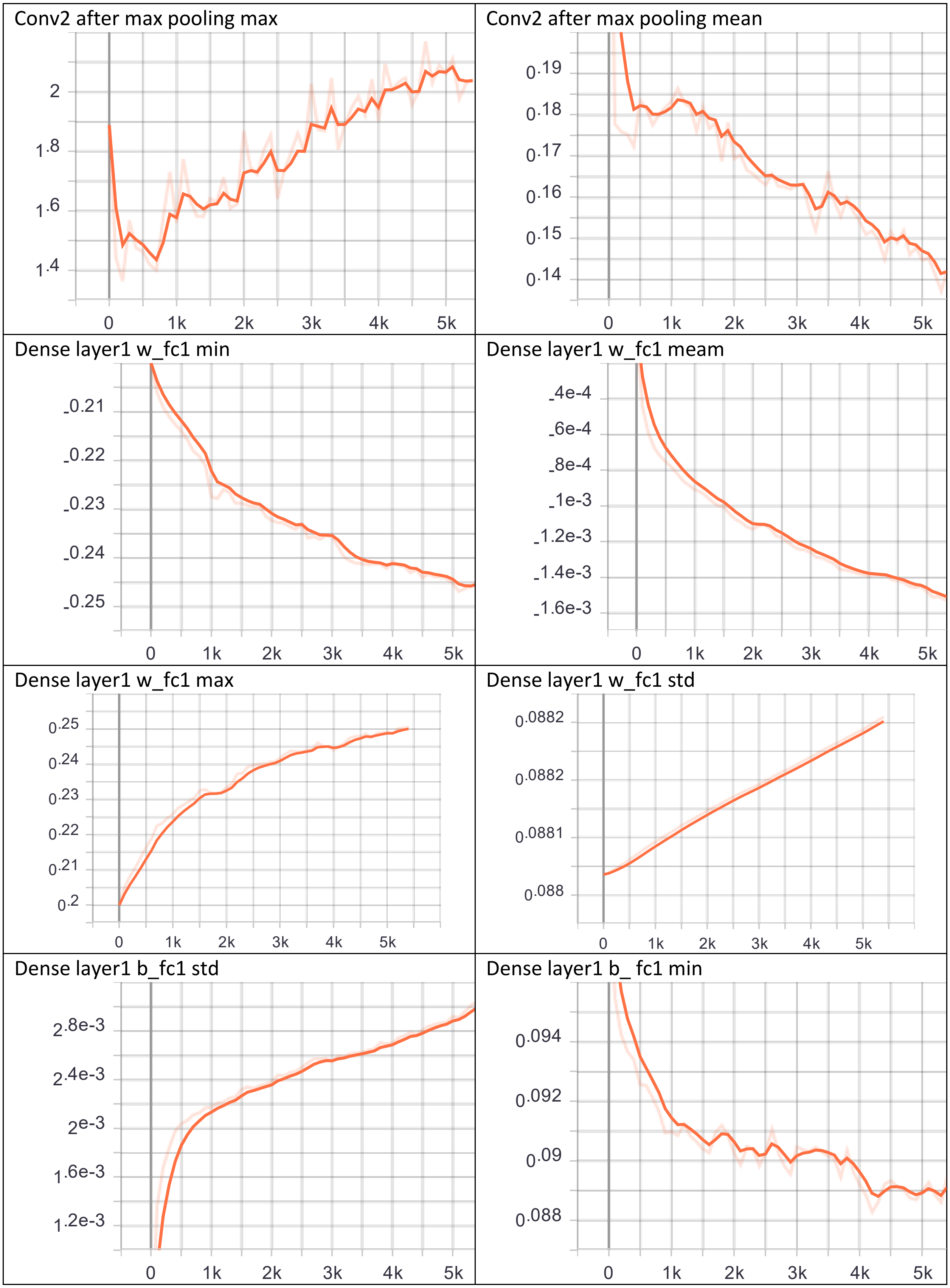


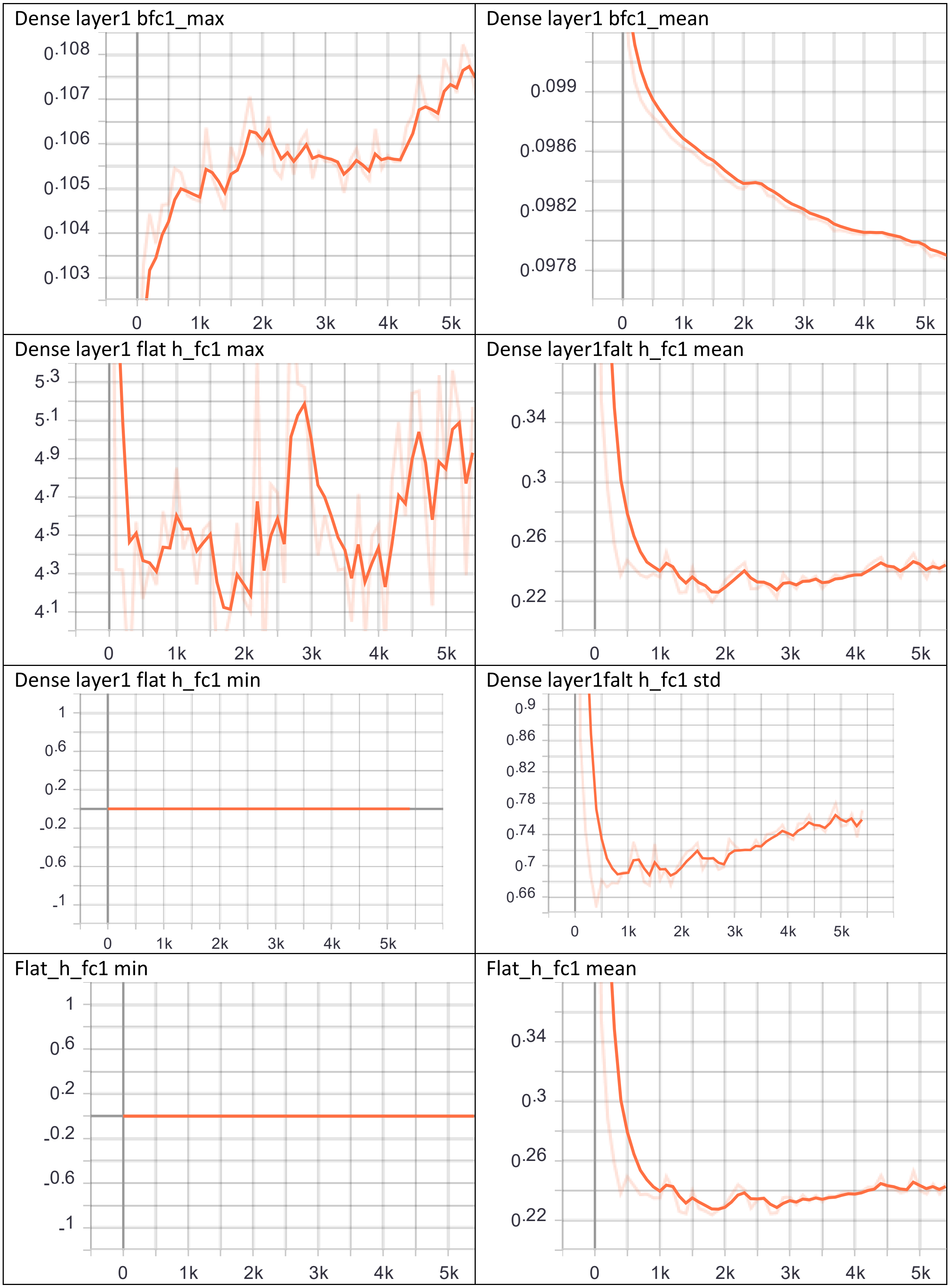


B_conv1 std
6e-3
4e-3
2e-3
0
0 1k 2k 3k 4k 5k
B_conv1 mean
0.098
0.096
0.094
0.092
0 1k 2k 3k 4k 5k
Conv1 after ReLU: h_conv1_std
0.116
0.112
0.108
0.104
0.1
0 1k 2k 3k 4k 5k
Activation_after_ReLU_h_conv1_min
1
0.6
0.2
-0.2
-0.6
-1
0 1k 2k 3k 4k 5k
Conv1_Activation_after_ReLU_h_conv1_mean
0.117
0.117
0.116
0.115
0.114
0 1k 2k 3k 4k 5k
Conv1_2_Activation_after_ReLU_h_conv1_max
1.35
1.25
1.15
1.05
0 1k 2k 3k 4k 5k
Conv1 after max pooling h_pool1 std
0.144
0.14
0.136
0.132
0.128
0.124
0.12
0 1k 2k 3k 4k 5k
Conv1 after max pooling h_pool1 max
1.35
1.25
1.15
1.05
0 1k 2k 3k 4k 5k

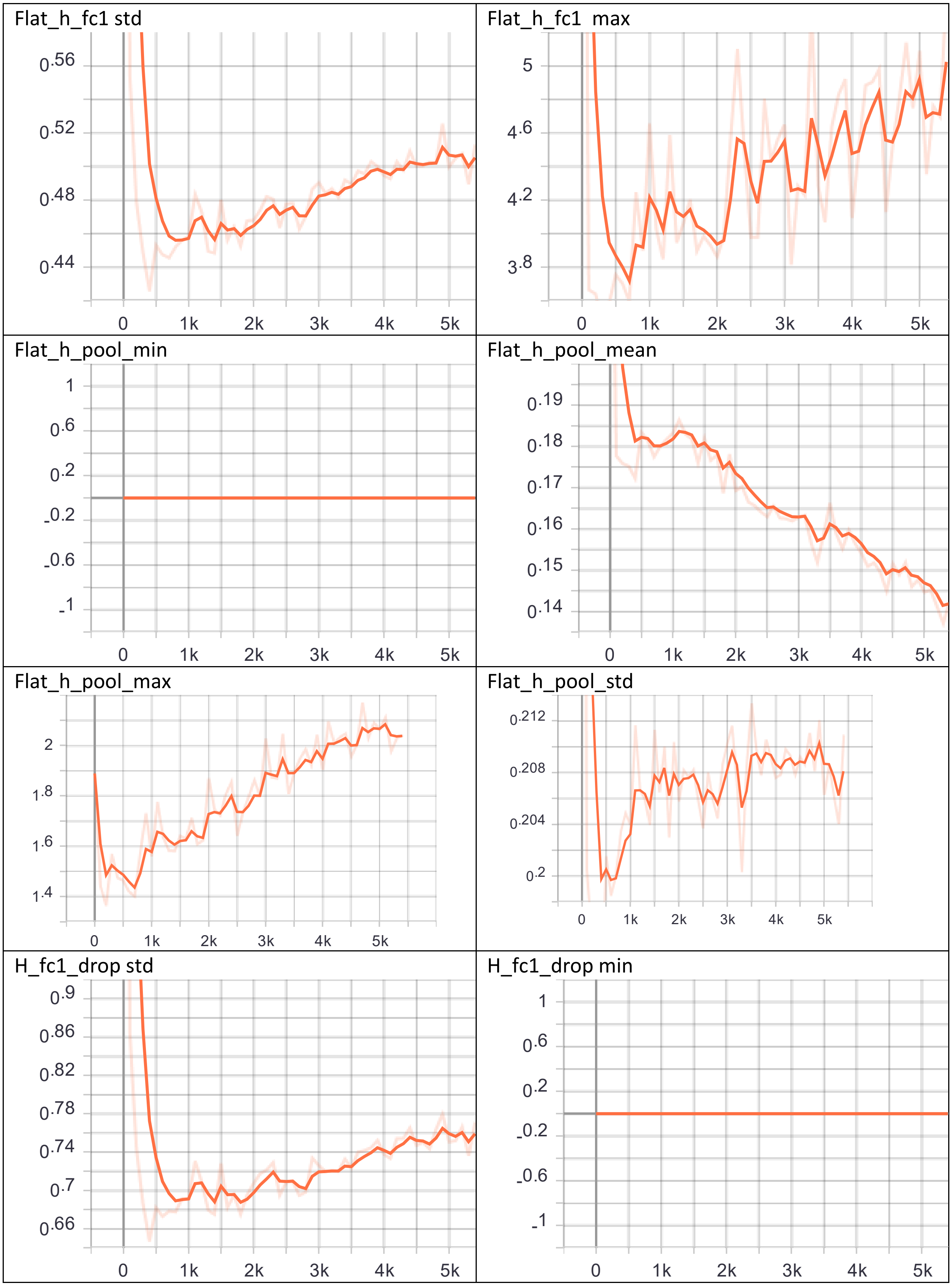

Conv1 after max pooling h_pool1 min
1
0.6
0.2
-0.2
-0.6
-1
0
1k
2k
3k
4k
5k
Conv1 after max pooling h_pool1 mean
0.147
0.145
0.143
0.141
0
1k
2k
3k
4k
5k
Conv2 w_conv2 std
0.0879
0.0878
0.0877
0.0876
0.0875
0
1k
2k
3k
4k
5k
Conv2 w_conv2 min
-0.204
-0.208
-0.212
-0.216
-0.22
0
1k
2k
3k
4k
5k
Conv2 w_conv2 mean
-1e-3
-1.4e-3
-1.8e-3
-2.2e-3
0
1k
2k
3k
4k
5k
Conv2 w_conv2 max
0.24
0.23
0.22
0.21
0.2
0
1k
2k
3k
4k
5k
Conv2 b_conv2 std
2.8e-3
2.4e-3
2e-3
1.6e-3
0
1k
2k
3k
4k
5k
Conv2 b_conv2 min
0.095
0.093
0.091
0.089
0.087
0
1k
2k
3k
4k
5k


Conv2 b_conv2 max
0.104
0.104
0.103
0.103
0.103
0.102
0.102
0 1k 2k 3k 4k 5k
Conv2 b_conv2 mean
0.099
0.098
0.097
0.096
0.095
0.094
0 1k 2k 3k 4k 5k
Conv2 after relu h_conv2 std
0.162
0.158
0.154
0.15
0.146
0 1k 2k 3k 4k 5k
Conv2 after relu h_conv2 min
1
0.6
0.2
-0.2
-0.6
-1
0 1k 2k 3k 4k 5k
Conv2 after relu h_max
2
1.8
1.6
1.4
0 1k 2k 3k 4k 5k
Conv2 after relu h_conv2 mean
0.105
0.095
0.085
0.075
0.065
0 1k 2k 3k 4k 5k
Conv2 after max pooling std
0.212
0.208
0.204
0.2
0 1k 2k 3k 4k 5k
Conv2 after max pooling min
1
0.6
0.2
-0.2
-0.6
-1
0 1k 2k 3k 4k 5k

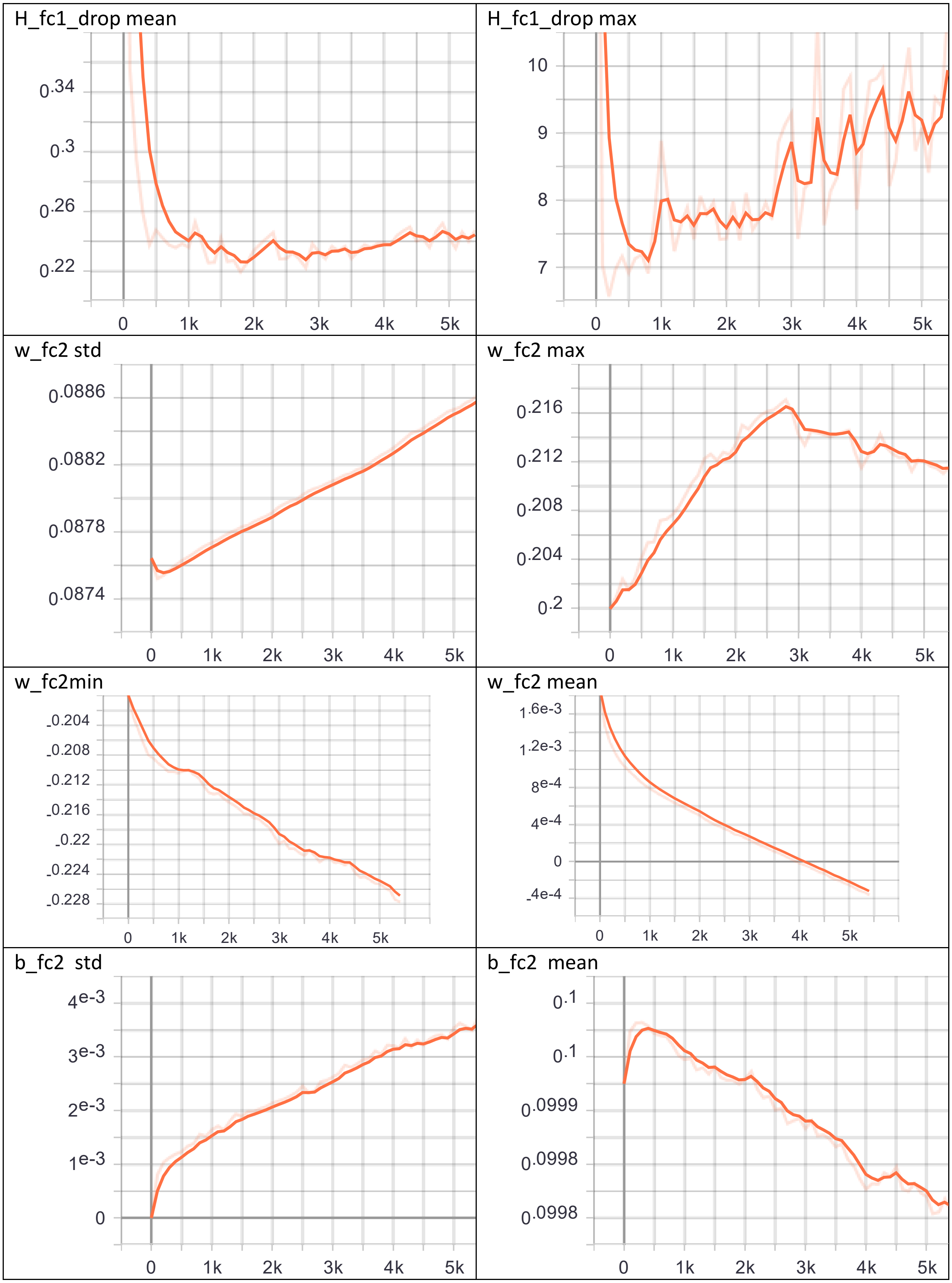

Conv2 after max pooling max
2
1.8
1.6
1.4
0
1k
2k
3k
4k
5k
Conv2 after max pooling mean
0.19
0.18
0.17
0.16
0.15
0.14
0
1k
2k
3k
4k
5k
Dense layer1 w_fc1 min
-0.21
-0.22
-0.23
-0.24
-0.25
0
1k
2k
3k
4k
5k
Dense layer1 w_fc1 meam
-4e-4
-6e-4
-8e-4
-1e-3
-1.2e-3
-1.4e-3
-1.6e-3
0
1k
2k
3k
4k
5k
Dense layer1 w_fc1 max
0.25
0.24
0.23
0.22
0.21
0.2
0
1k
2k
3k
4k
5k
Dense layer1 w_fc1 std
0.0882
0.0882
0.0881
0.088
0
1k
2k
3k
4k
5k
Dense layer1 b_fc1 std
2.8e-3
2.4e-3
2e-3
1.6e-3
1.2e-3
0
1k
2k
3k
4k
5k
Dense layer1 b_ fc1 min
0.094
0.092
0.09
0.088
0
1k
2k
3k
4k
5k



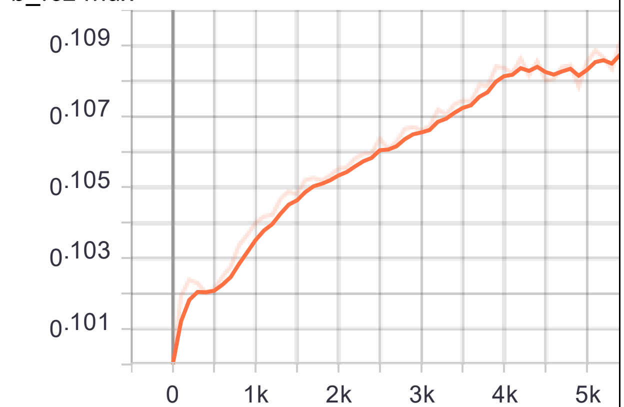
Dense layer1 bfc1_max
0.108
0.107
0.106
0.105
0.104
0.103
0 1k 2k 3k 4k 5k
Dense layer1 bfc1_mean
0.099
0.0986
0.0982
0.0978
0 1k 2k 3k 4k 5k
Dense layer1 flat h_fc1 max
5.3
5.1
4.9
4.7
4.5
4.3
4.1
0 1k 2k 3k 4k 5k
Dense layer1falt h_fc1 mean
0.34
0.3
0.26
0.22
0 1k 2k 3k 4k 5k
Dense layer1 flat h_fc1 min
1
0.6
0.2
-0.2
-0.6
-1
0 1k 2k 3k 4k 5k
Dense layer1falt h_fc1 std
0.9
0.86
0.82
0.78
0.74
0.7
0.66
0 1k 2k 3k 4k 5k
Flat_h_fc1 min
1
0.6
0.2
-0.2
-0.6
-1
0 1k 2k 3k 4k 5k
Flat_h_fc1 mean
0.34
0.3
0.26
0.22
0 1k 2k 3k 4k 5k



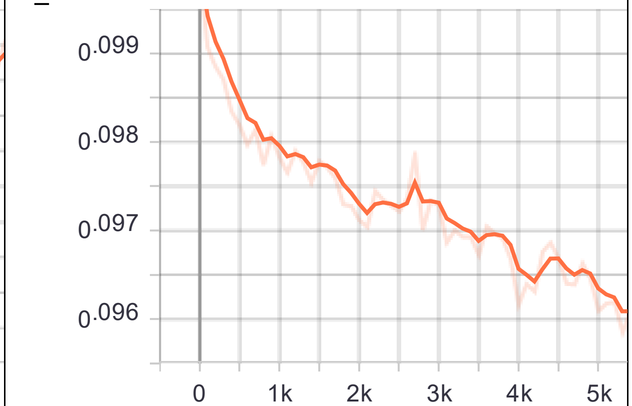
Flat_h_fc1 std
0.56
0.52
0.48
0.44
0
1k
2k
3k
4k
5k
Flat_h_fc1 max
5
4.6
4.2
3.8
0
1k
2k
3k
4k
5k
Flat_h_pool_min
1
0.6
0.2
-0.2
-0.6
-1
0
1k
2k
3k
4k
5k
Flat_h_pool_mean
0.19
0.18
0.17
0.16
0.15
0.14
0
1k
2k
3k
4k
5k
Flat_h_pool_max
2
1.8
1.6
1.4
0
1k
2k
3k
4k
5k
Flat_h_pool_std
0.212
0.208
0.204
0.2
0
1k
2k
3k
4k
5k
H_fc1_drop std
0.9
0.86
0.82
0.78
0.74
0.7
0.66
0
1k
2k
3k
4k
5k
H_fc1_drop min
1
0.6
0.2
-0.2
-0.6
-1
0
1k
2k
3k
4k
5k


H_fc1_drop mean
0.34
0.3
0.26
0.22
0
1k
2k
3k
4k
5k
H_fc1_drop max
10
9
8
7
0
1k
2k
3k
4k
5k
w_fc2 std
0.0886
0.0882
0.0878
0.0874
0
1k
2k
3k
4k
5k
w_fc2 max
0.216
0.212
0.208
0.204
0.2
0
1k
2k
3k
4k
5k
w_fc2min
-0.204
-0.208
-0.212
-0.216
-0.22
-0.224
-0.228
0
1k
2k
3k
4k
5k
w_fc2 mean
1.6e-3
1.2e-3
8e-4
4e-4
0
-4e-4
0
1k
2k
3k
4k
5k
b_fc2 std
4e-3
3e-3
2e-3
1e-3
0
0
1k
2k
3k
4k
5k
b_fc2 mean
0.1
0.1
0.0999
0.0998
0.0998
0
1k
2k
3k
4k
5k

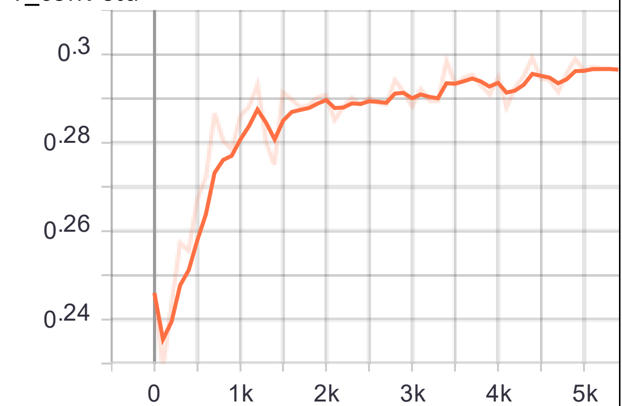
b_fc2 max



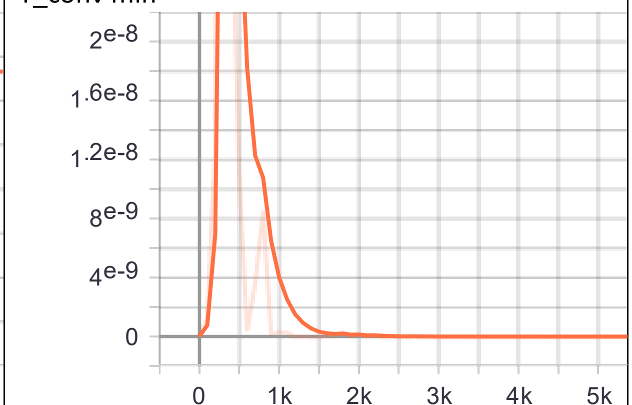
b_fc2 min



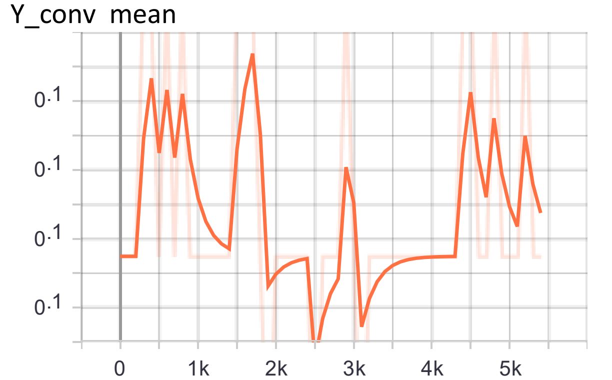
Y_conv std



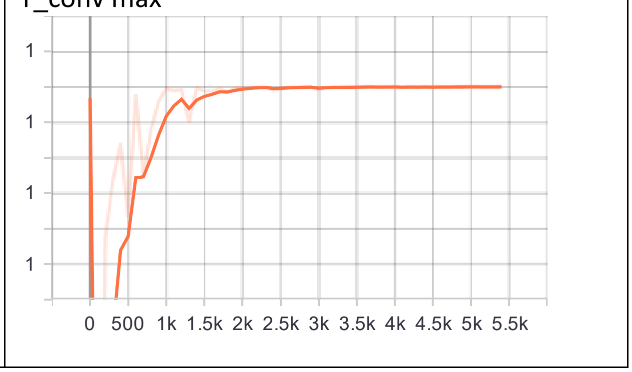
Y_conv min



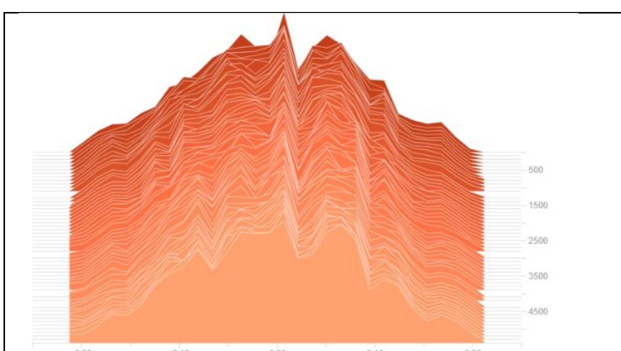
Y_conv mean



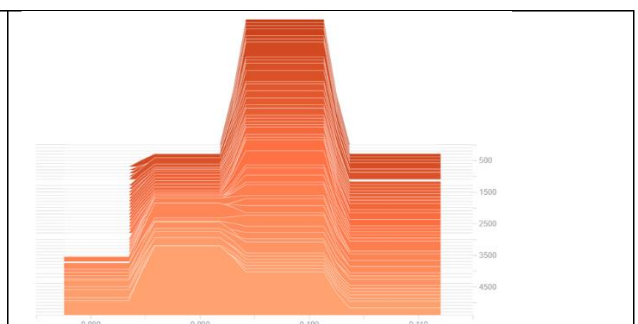
Y_conv max



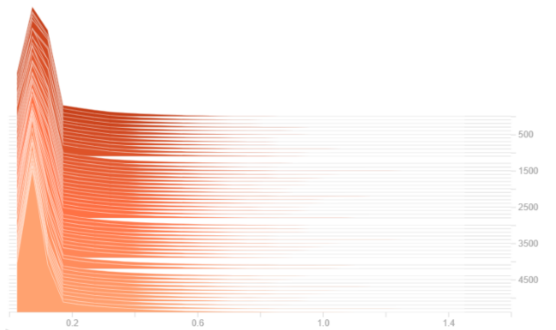
Histogram Plots:



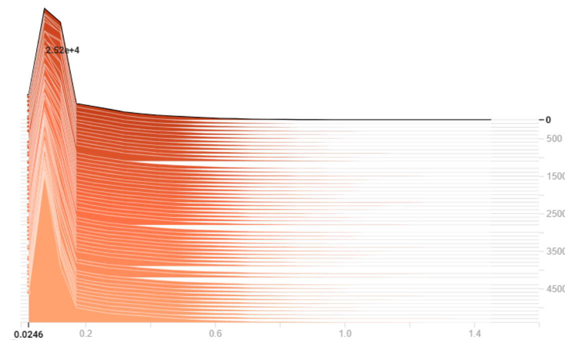
W_conv1



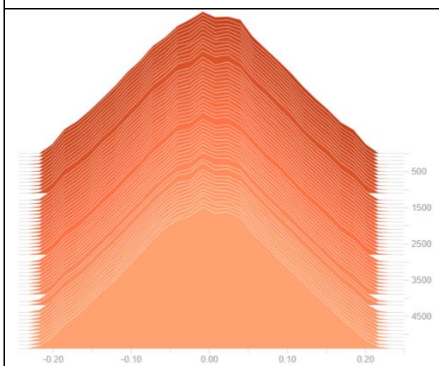
b_conv1



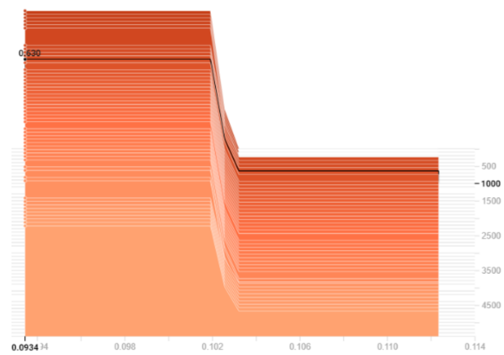
After relu h_conv1



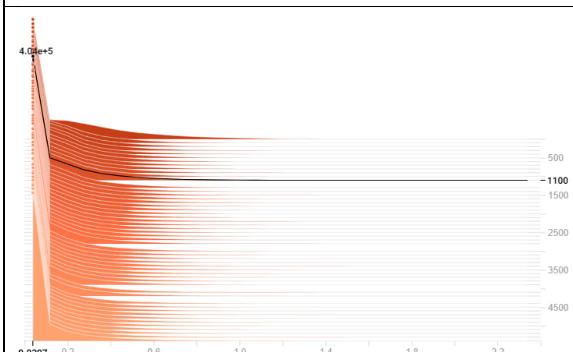
Max pool1



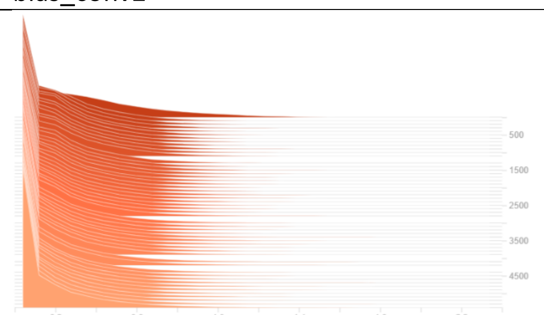
W_conv2



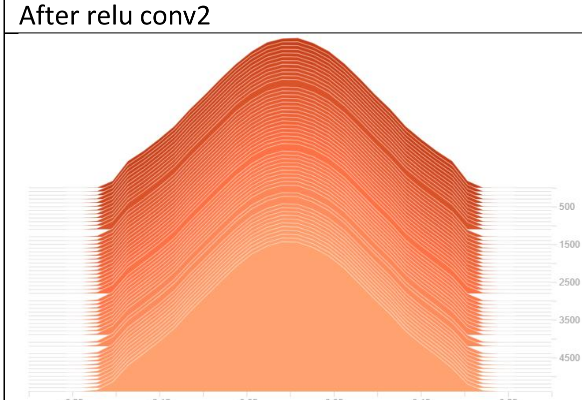
bias_conv2



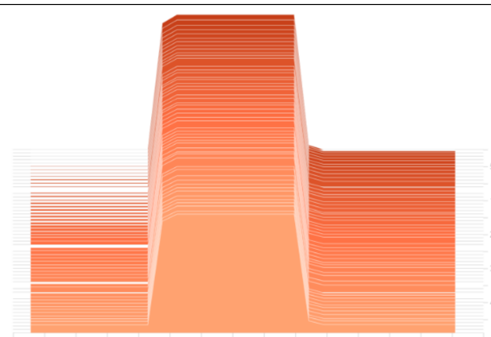
After relu conv2



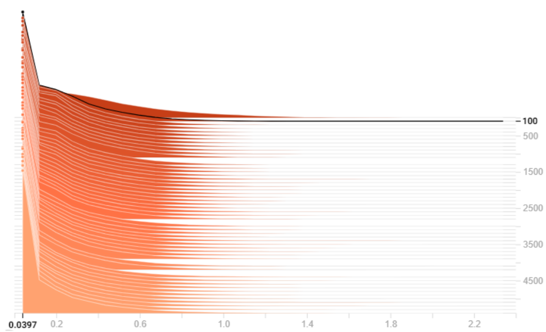
Max pool conv2



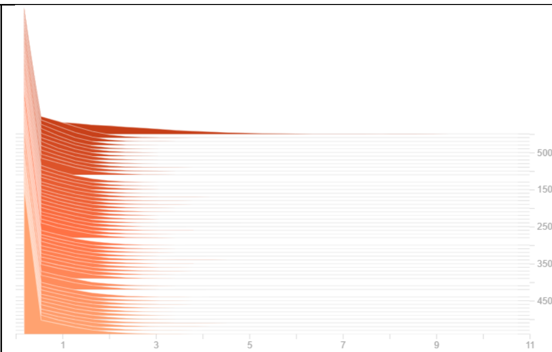
Dense layer weight



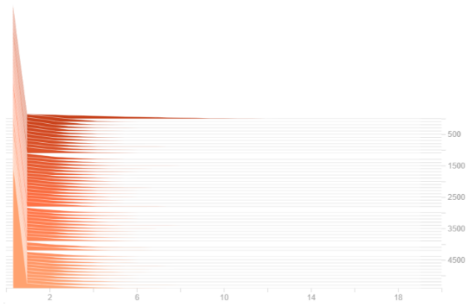
Dense layer biase



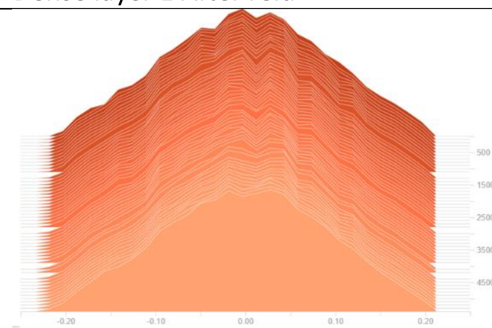
Dense layer h_pool2 flat



Dense layer 1 After relu

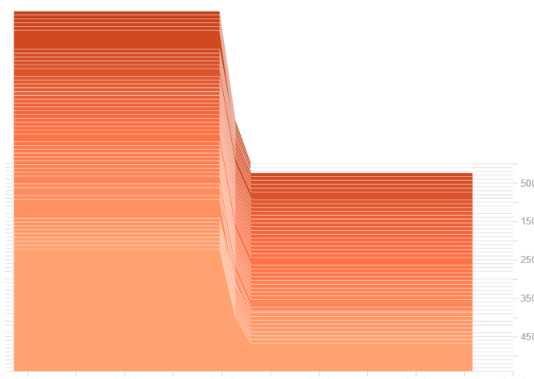


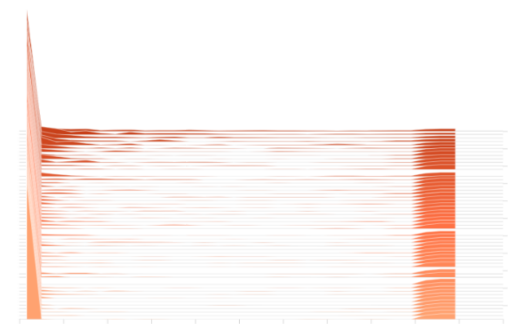
Dropout layer fc1_drop



Output layer weight

Output layer bias





Output layer Y_conv

(c)

For netwrok architecture for this part of the problem is:

conv1(5 - 5 - 1 - 32) - leaky-ReLU - maxpool(2 - 2) - conv2(5 - 5 - 32 - 64) - leaky-ReLU - maxpool(2 - 2) - fc(1024) - leaky-ReLU - DropOut(0.5) - Softmax(10).

I used Xavier initialization technique and Monemtum optimizer to train this network. The figure 9 shows the training, test and validation loss and accuracy curve for this network.
The test accuracy after at was approximately 98.23% and it the training takes 121.732510 second to finish, where the network training in part (a) takes 64.156681 second to finish and the accuracy is 98.73.

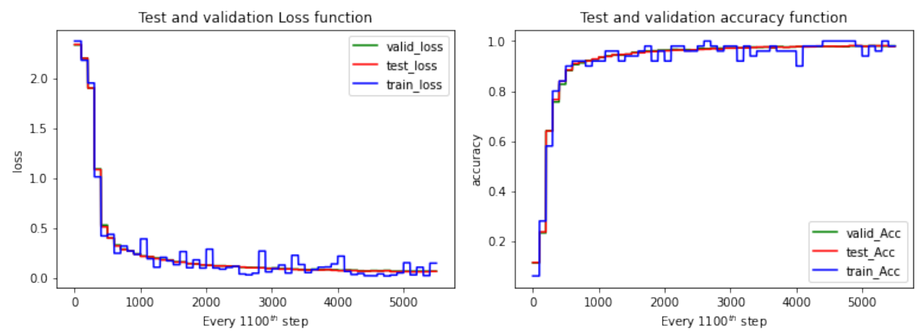
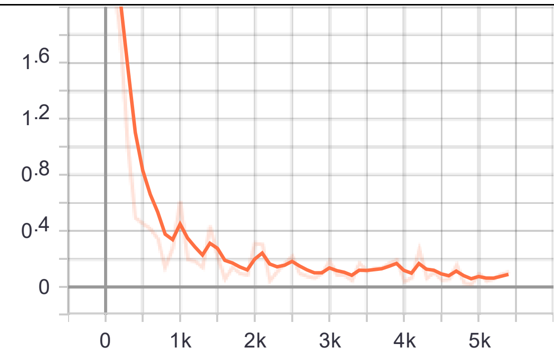
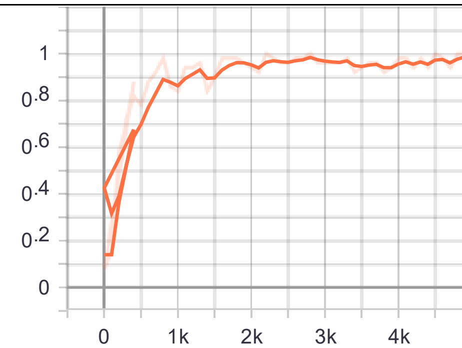


Figure 9: The loss and accuracy plots of training, validation and test

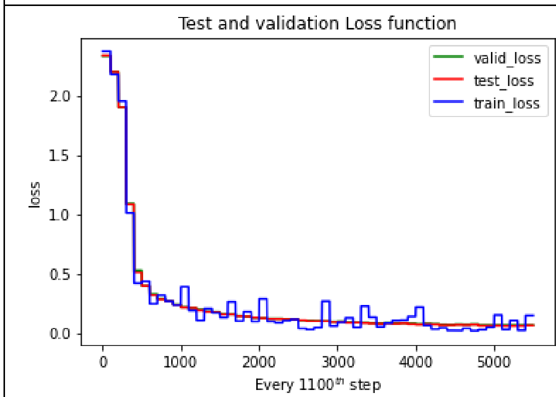
Part C:



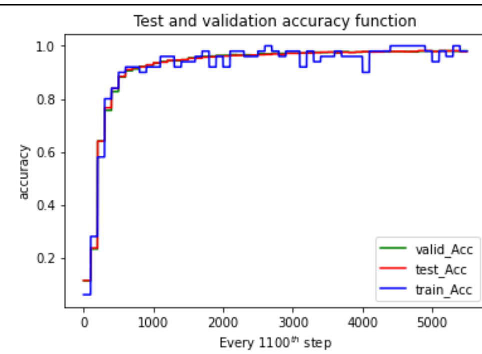
Train loss



Train accuracy

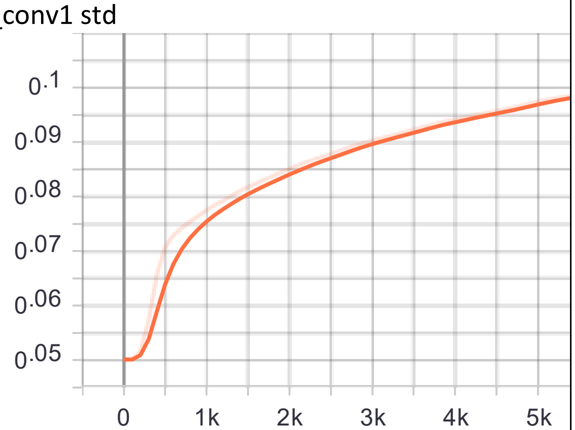


Train, test and validation loss

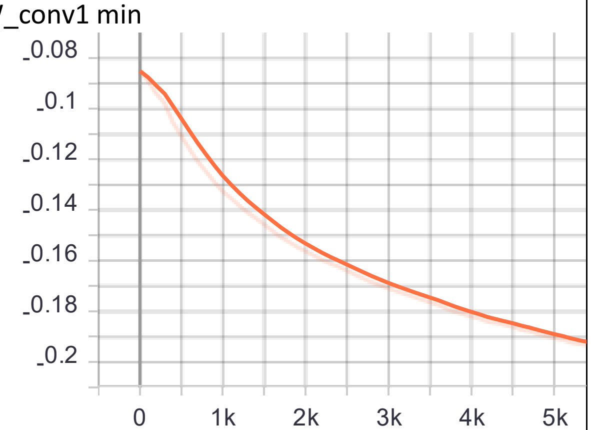


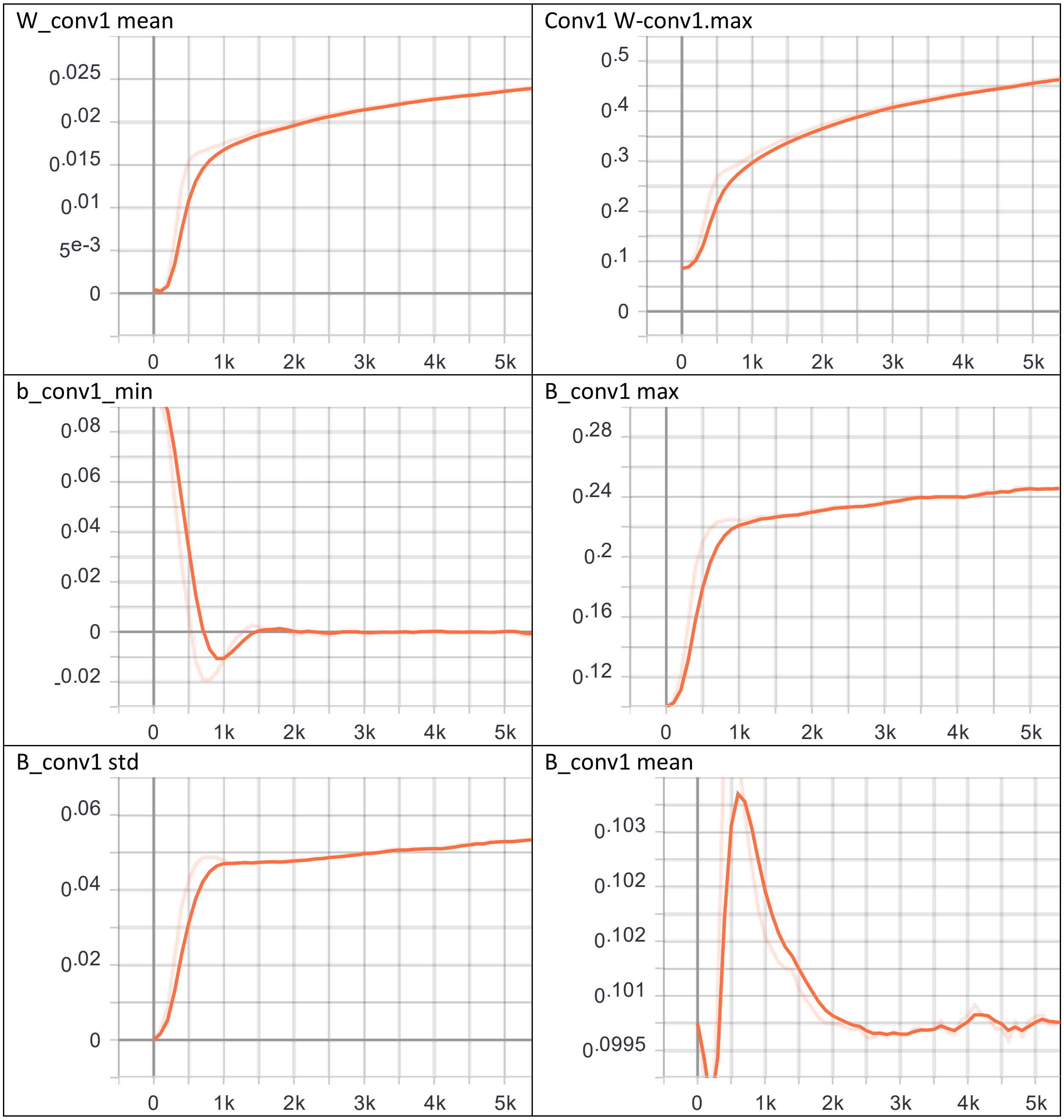
Train, test and validation accuracy

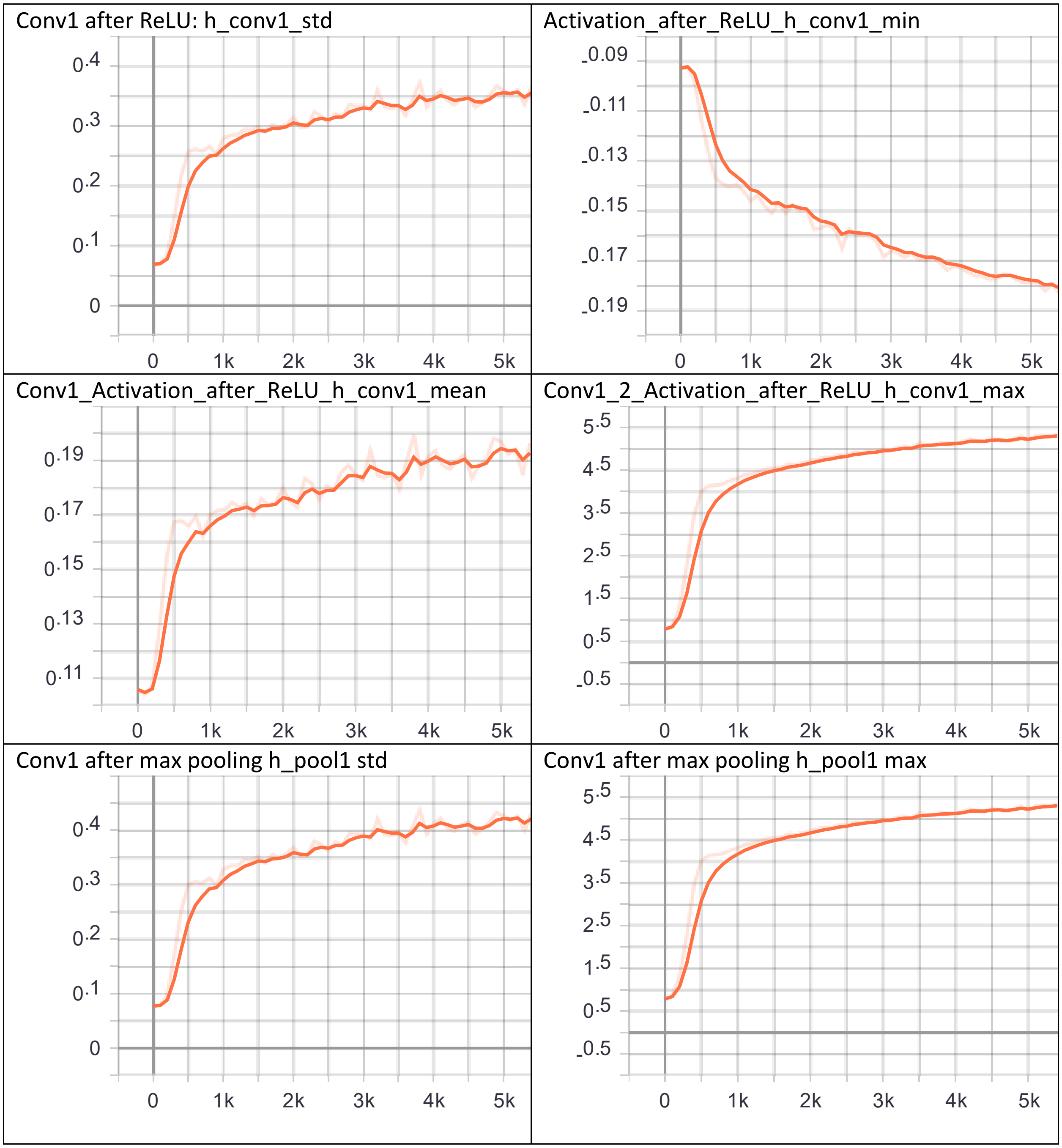
W_conv1 std

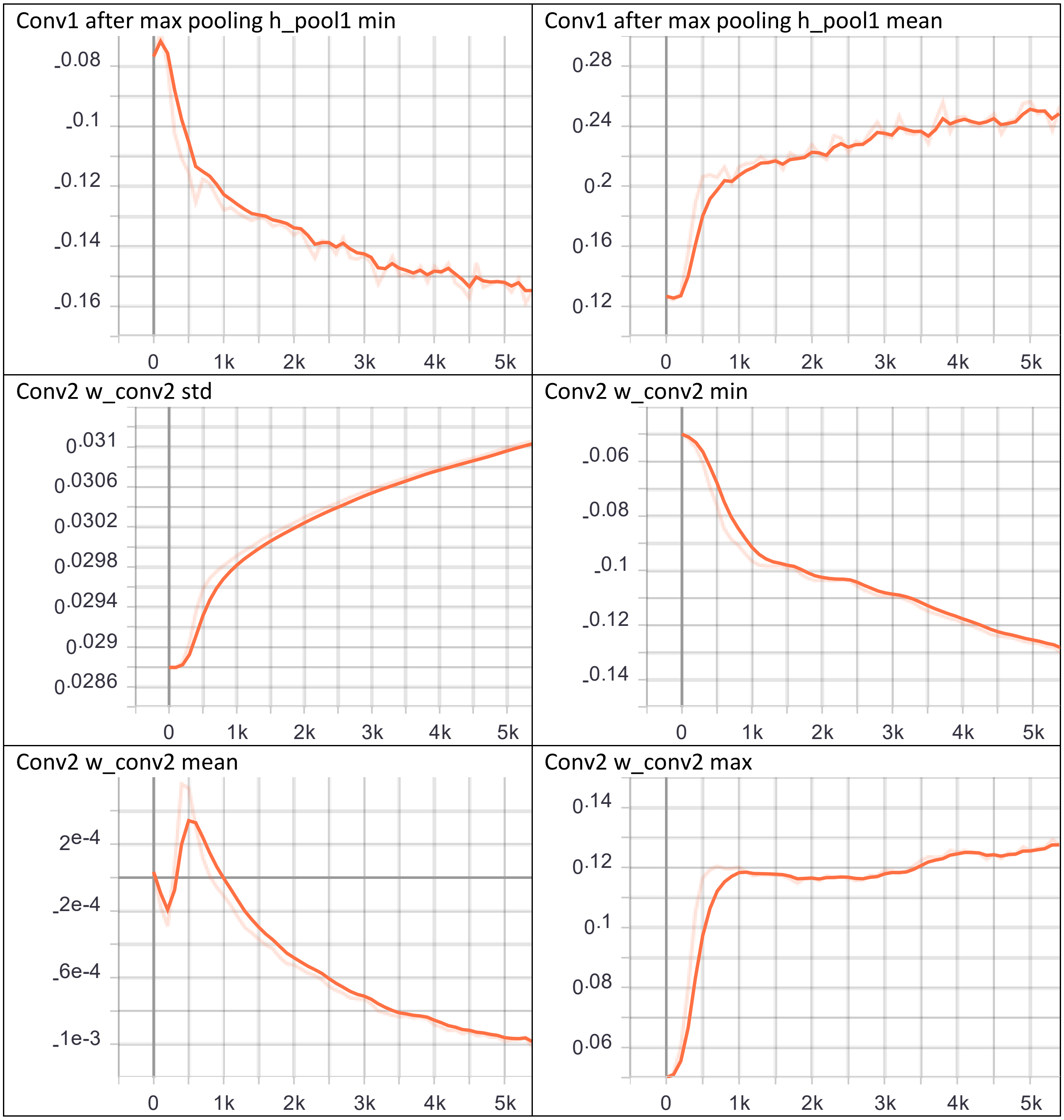


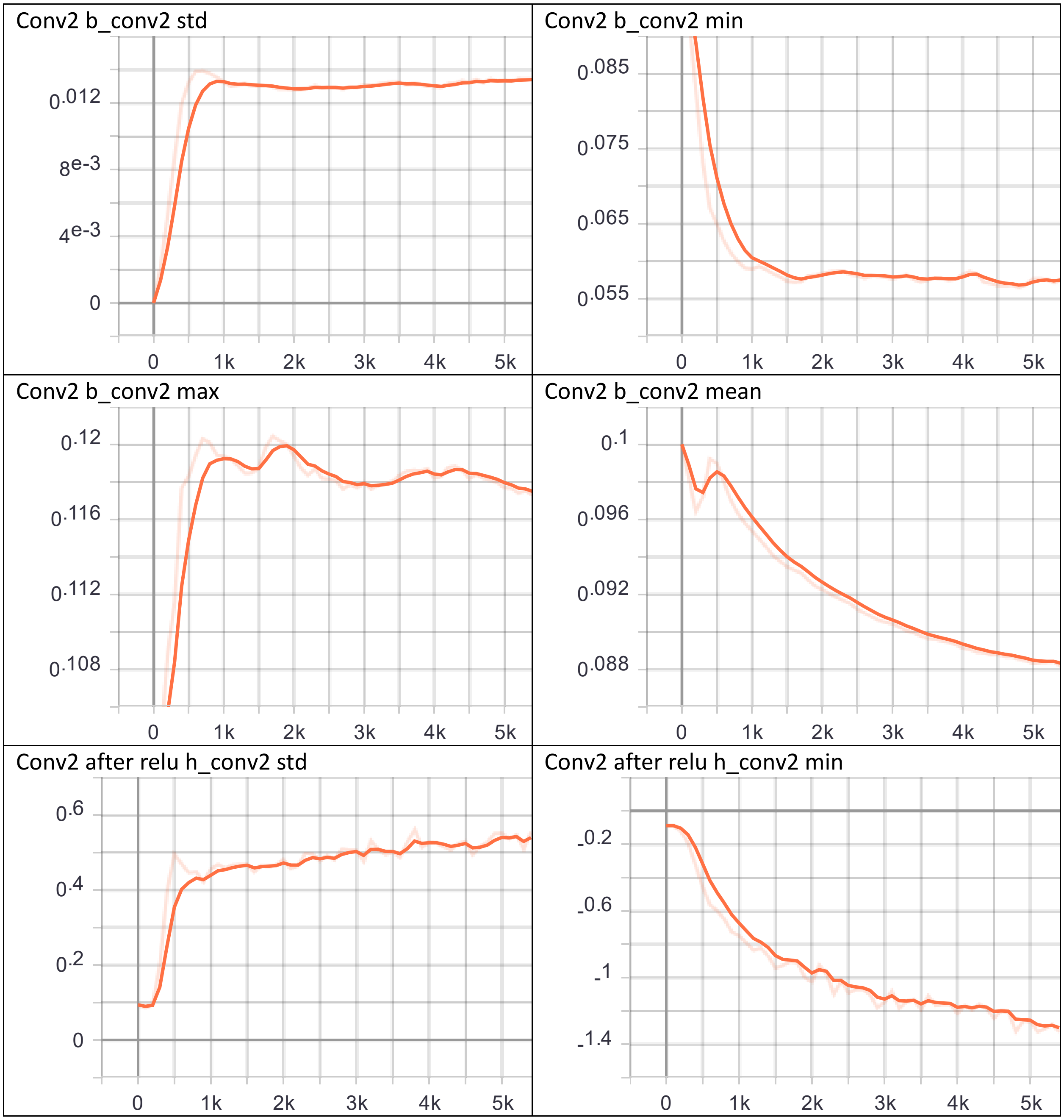
W_conv1 min

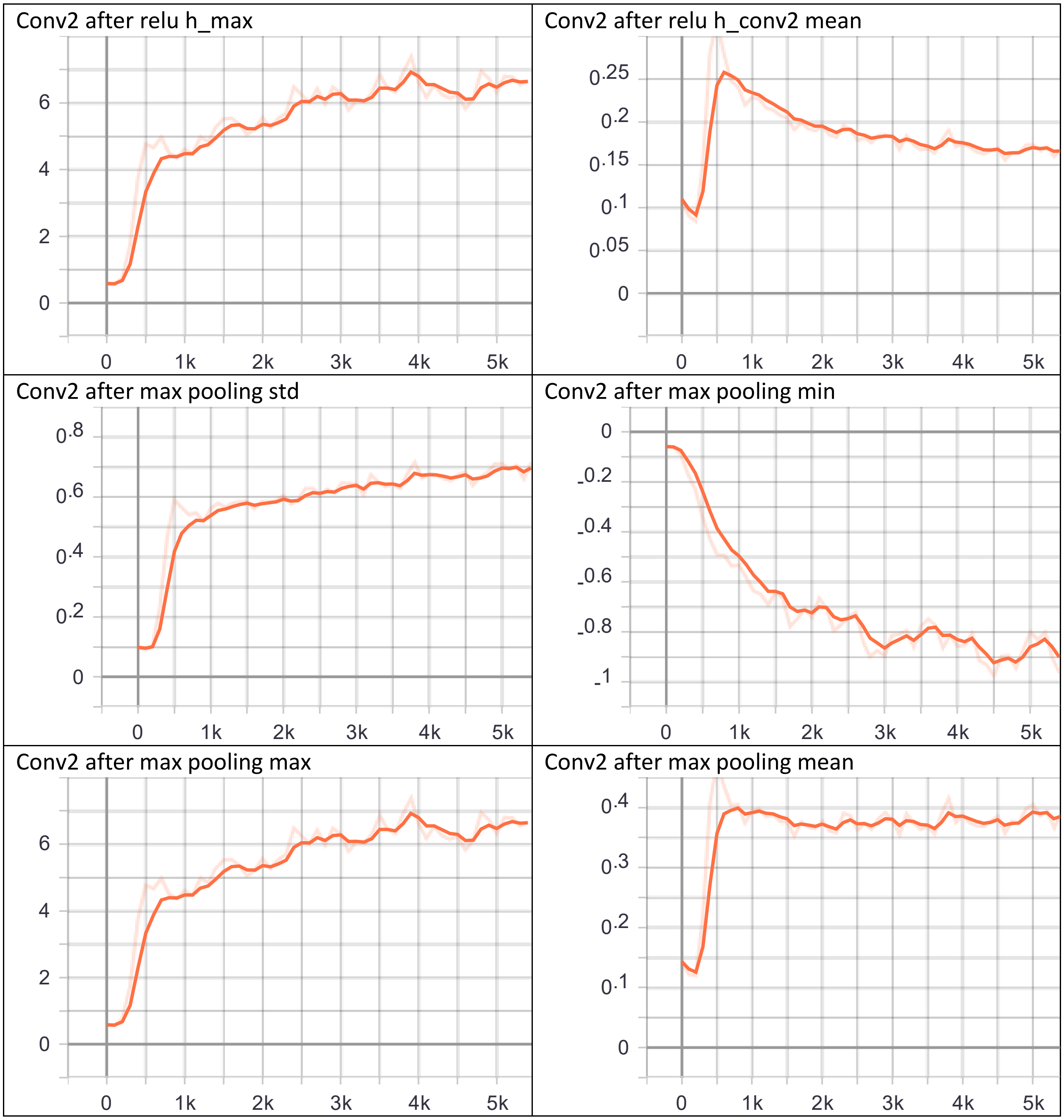


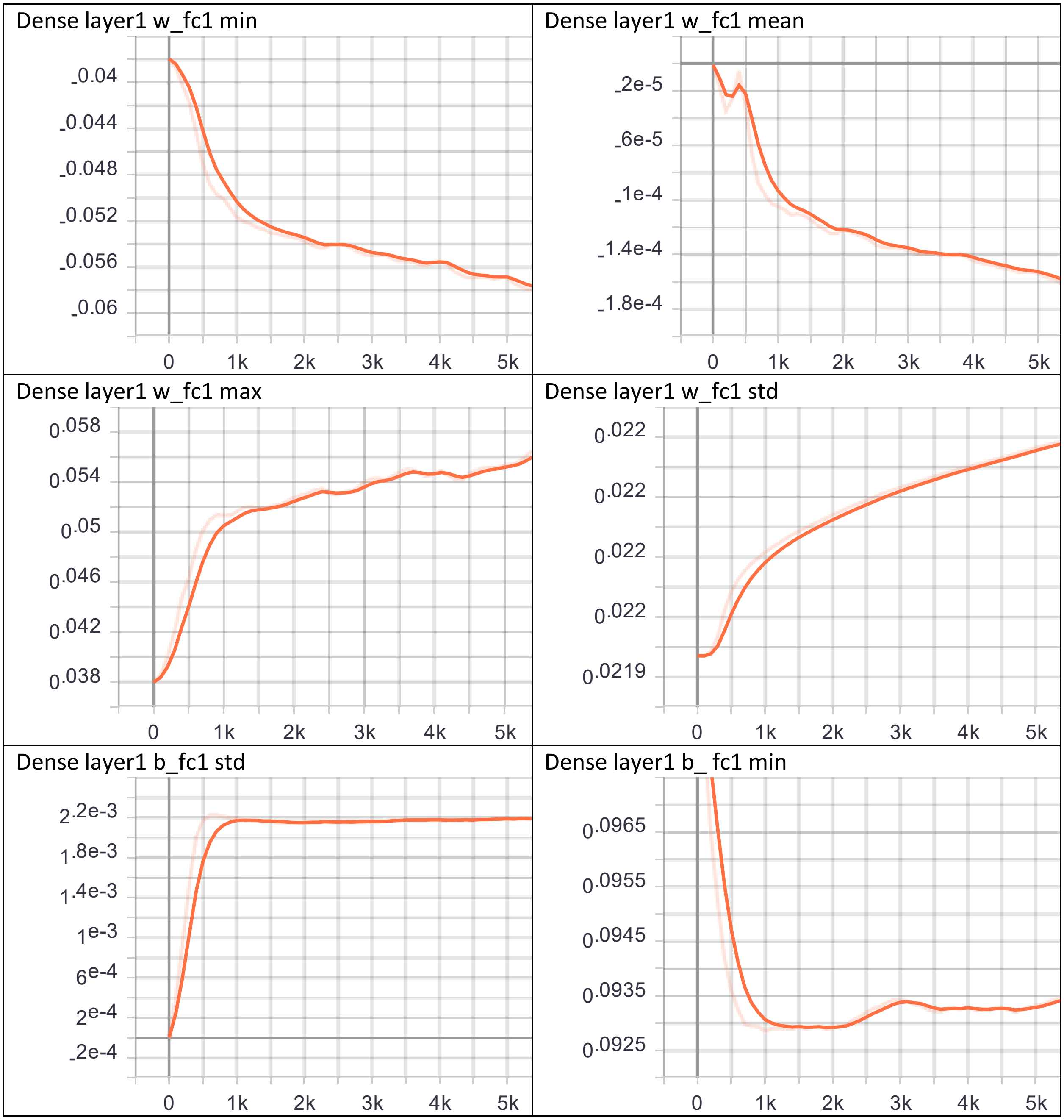

W_conv1 mean
0.025
0.02
0.015
0.01
5e-3
0
0
1k
2k
3k
4k
5k
Conv1 W-conv1.max
0.5
0.4
0.3
0.2
0.1
0
0
1k
2k
3k
4k
5k
b_conv1_min
0.08
0.06
0.04
0.02
0
-0.02
0
1k
2k
3k
4k
5k
B_conv1 max
0.28
0.24
0.2
0.16
0.12
0
1k
2k
3k
4k
5k
B_conv1 std
0.06
0.04
0.02
0
0
1k
2k
3k
4k
5k
B_conv1 mean
0.103
0.102
0.102
0.101
0.0995
0
1k
2k
3k
4k
5k

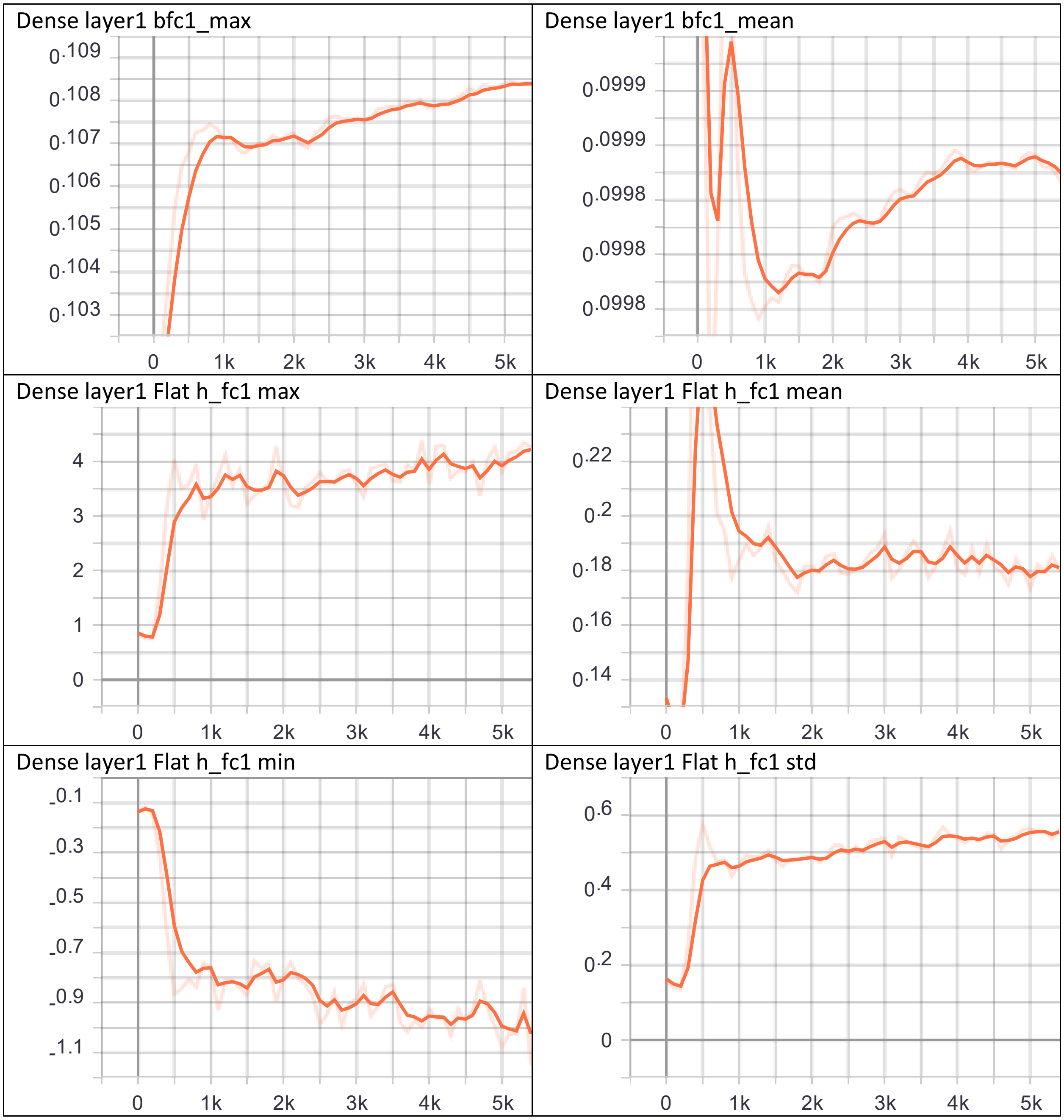

Conv1 after ReLU: h_conv1_std
0.4
0.3
0.2
0.1
0
0
1k
2k
3k
4k
5k
Activation_after_ReLU_h_conv1_min
-0.09
-0.11
-0.13
-0.15
-0.17
-0.19
0
1k
2k
3k
4k
5k
Conv1_Activation_after_ReLU_h_conv1_mean
0.19
0.17
0.15
0.13
0.11
0
1k
2k
3k
4k
5k
Conv1_2_Activation_after_ReLU_h_conv1_max
5.5
4.5
3.5
2.5
1.5
0.5
-0.5
0
1k
2k
3k
4k
5k
Conv1 after max pooling h_pool1 std
0.4
0.3
0.2
0.1
0
0
1k
2k
3k
4k
5k
Conv1 after max pooling h_pool1 max
5.5
4.5
3.5
2.5
1.5
0.5
-0.5
0
1k
2k
3k
4k
5k


Conv1 after max pooling h_pool1 min
-0.08
-0.1
-0.12
-0.14
-0.16
0
1k
2k
3k
4k
5k
Conv1 after max pooling h_pool1 mean
0.28
0.24
0.2
0.16
0.12
0
1k
2k
3k
4k
5k
Conv2 w_conv2 std
0.031
0.0306
0.0302
0.0298
0.0294
0.029
0.0286
0
1k
2k
3k
4k
5k
Conv2 w_conv2 min
-0.06
-0.08
-0.1
-0.12
-0.14
0
1k
2k
3k
4k
5k
Conv2 w_conv2 mean
2e-4
-2e-4
-6e-4
-1e-3
0
1k
2k
3k
4k
5k
Conv2 w_conv2 max
0.14
0.12
0.1
0.08
0.06
0
1k
2k
3k
4k
5k

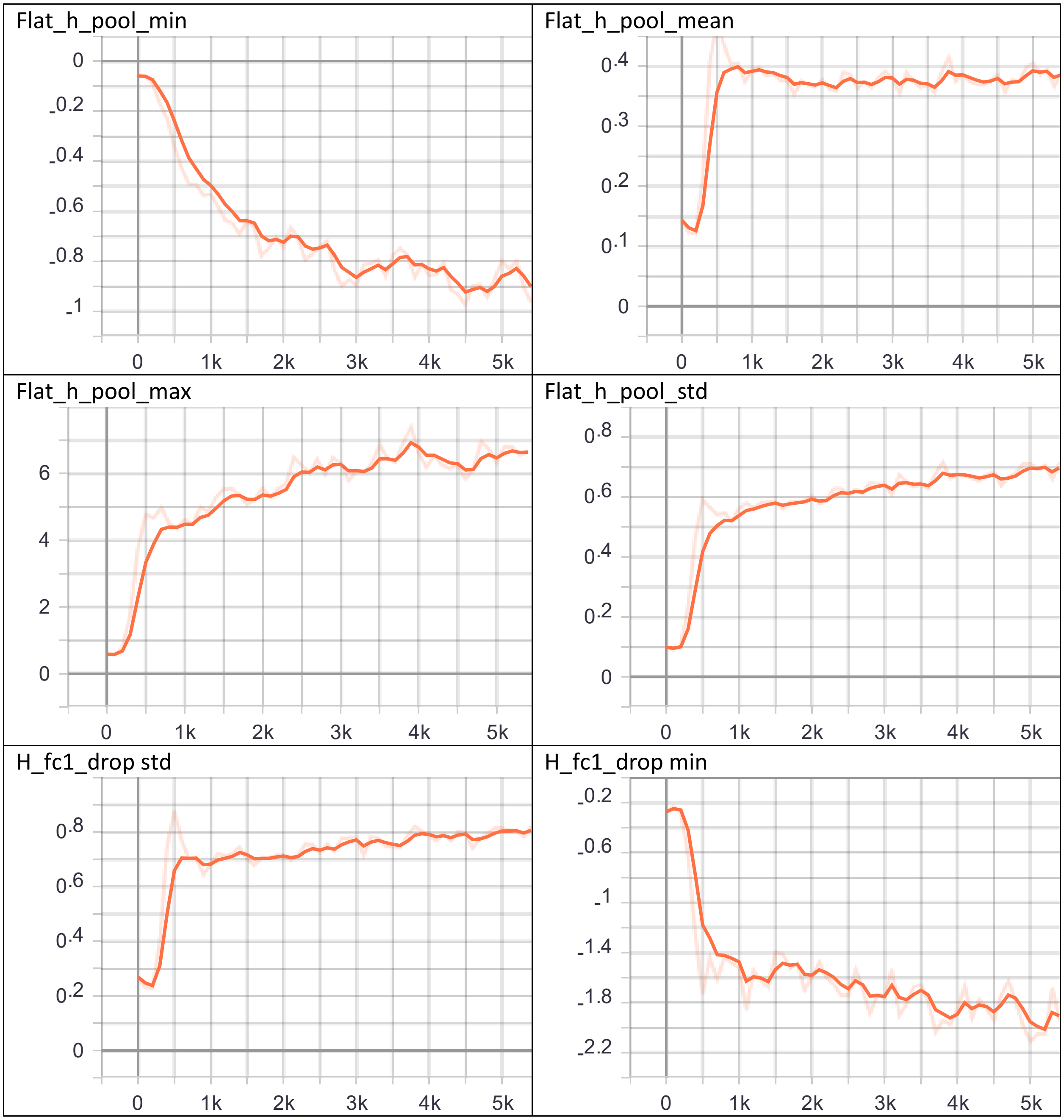

Conv2 b_conv2 std
0.012
8e-3
4e-3
0
0
1k
2k
3k
4k
5k
Conv2 b_conv2 min
0.085
0.075
0.065
0.055
Conv2 b_conv2 max
0.12
0.116
0.112
0.108
Conv2 b_conv2 mean
0.1
0.096
0.092
0.088
Conv2 after relu h_conv2 std
0.6
0.4
0.2
Conv2 after relu h_conv2 min
-0.2
-0.6
-1
-1.4


Conv2 after relu h_max
6
4
2
0
0
1k
2k
3k
4k
5k
Conv2 after relu h_conv2 mean
0.25
0.2
0.15
0.1
0.05
0
0
1k
2k
3k
4k
5k
Conv2 after max pooling std
0.8
0.6
0.4
0.2
0
0
1k
2k
3k
4k
5k
Conv2 after max pooling min
0
-0.2
-0.4
-0.6
-0.8
-1
0
1k
2k
3k
4k
5k
Conv2 after max pooling max
6
4
2
0
0
1k
2k
3k
4k
5k
Conv2 after max pooling mean
0.4
0.3
0.2
0.1
0
0
1k
2k
3k
4k
5k

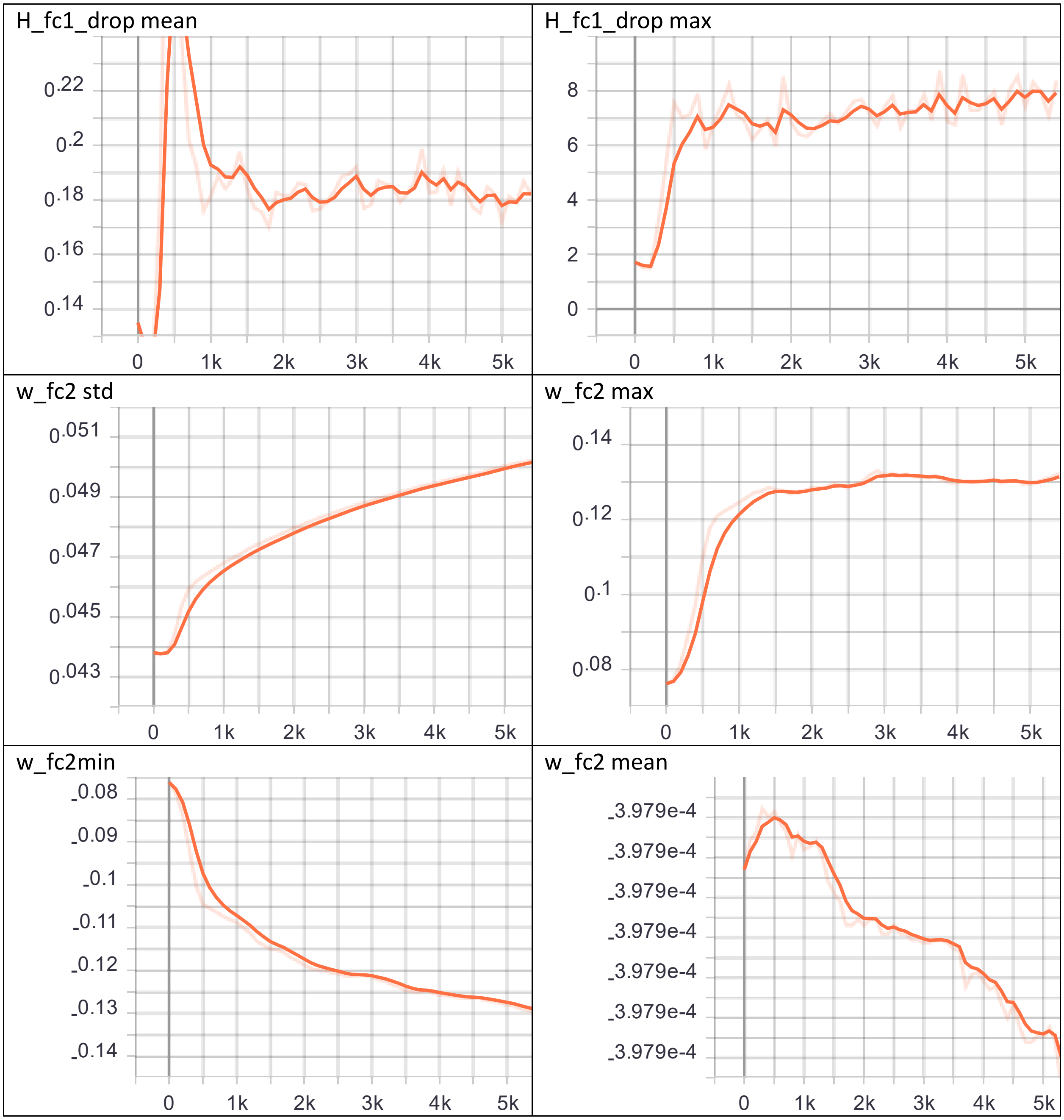

Dense layer1 w_fc1 min
-0.04
-0.044
-0.048
-0.052
-0.056
-0.06
0
1k
2k
3k
4k
5k
Dense layer1 w_fc1 mean
-2e-5
-6e-5
-1e-4
-1.4e-4
-1.8e-4
Dense layer1 w_fc1 max
0.058
0.054
0.05
0.046
0.042
0.038
Dense layer1 w_fc1 std
0.022
0.022
0.022
0.022
0.0219
Dense layer1 b_fc1 std
2.2e-3
1.8e-3
1.4e-3
1e-3
6e-4
2e-4
-2e-4
Dense layer1 b_ fc1 min
0.0965
0.0955
0.0945
0.0935
0.0925

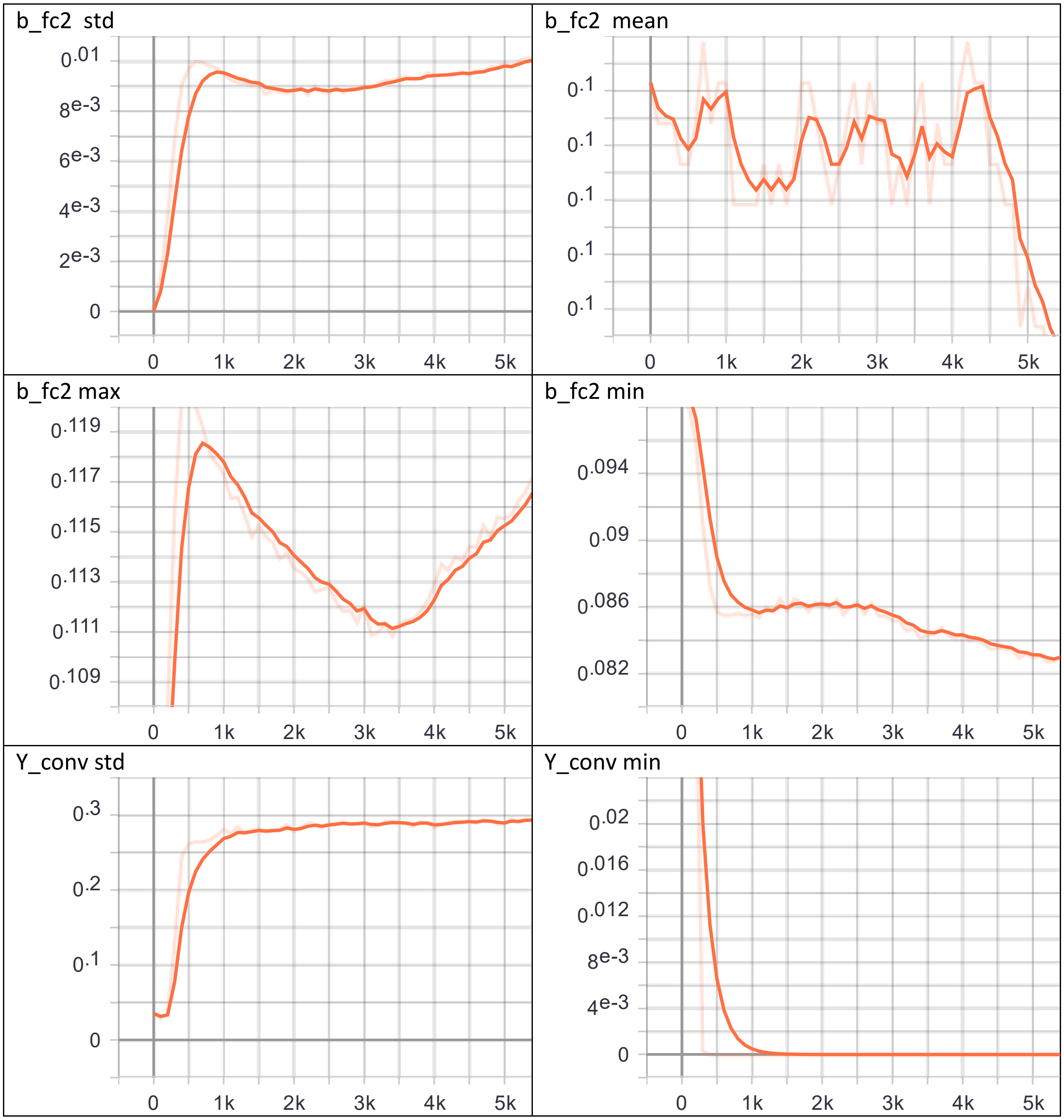


Dense layer1 bfc1_max
0.109
0.108
0.107
0.106
0.105
0.104
0.103
0
1k
2k
3k
4k
5k
Dense layer1 bfc1_mean
0.0999
0.0999
0.0998
0.0998
0.0998
0
1k
2k
3k
4k
5k
Dense layer1 Flat h_fc1 max
4
3
2
1
0
0
1k
2k
3k
4k
5k
Dense layer1 Flat h_fc1 mean
0.22
0.2
0.18
0.16
0.14
0
1k
2k
3k
4k
5k
Dense layer1 Flat h_fc1 min
-0.1
-0.3
-0.5
-0.7
-0.9
-1.1
0
1k
2k
3k
4k
5k
Dense layer1 Flat h_fc1 std
0.6
0.4
0.2
0
0
1k
2k
3k
4k
5k

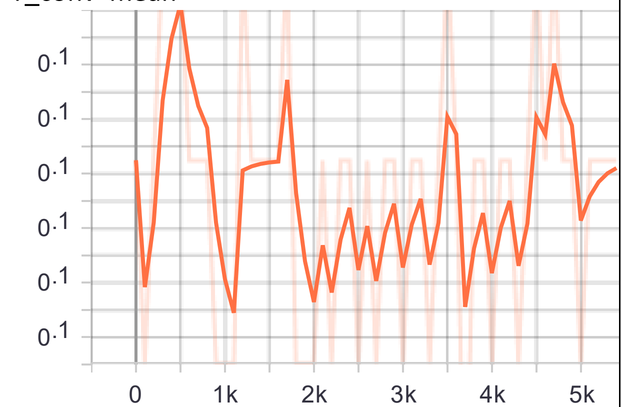


Flat_h_pool_min
0
-0.2
-0.4
-0.6
-0.8
-1
0
1k
2k
3k
4k
5k
Flat_h_pool_mean
0.4
0.3
0.2
0.1
0
0
1k
2k
3k
4k
5k
Flat_h_pool_max
6
4
2
0
0
1k
2k
3k
4k
5k
Flat_h_pool_std
0.8
0.6
0.4
0.2
0
0
1k
2k
3k
4k
5k
H_fc1_drop std
0.8
0.6
0.4
0.2
0
0
1k
2k
3k
4k
5k
H_fc1_drop min
-0.2
-0.6
-1
-1.4
-1.8
-2.2
0
1k
2k
3k
4k
5k

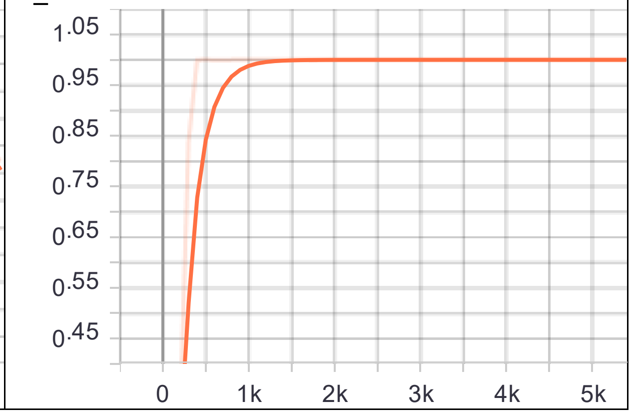

H_fc1_drop mean
H_fc1_drop max
w_fc2 std
w_fc2 max
w_fc2min
w_fc2 mean


b_fc2 std
0.01
8e-3
6e-3
4e-3
2e-3
0
0
1k
2k
3k
4k
5k
b_fc2 mean
0.1
0.1
0.1
0.1
0.1
b_fc2 max
0.119
0.117
0.115
0.113
0.111
0.109
b_fc2 min
0.094
0.09
0.086
0.082
Y_conv std
0.3
0.2
0.1
0
Y_conv min
0.02
0.016
0.012
8e-3
4e-3
0

Y_conv mean

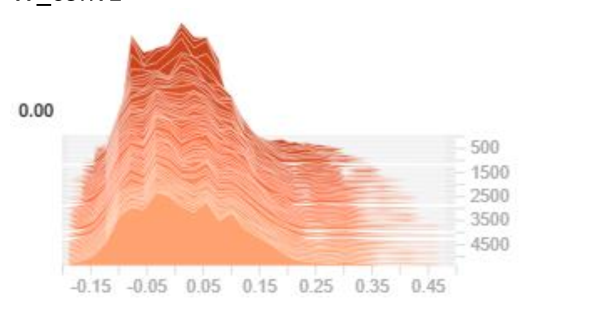


Y_conv max

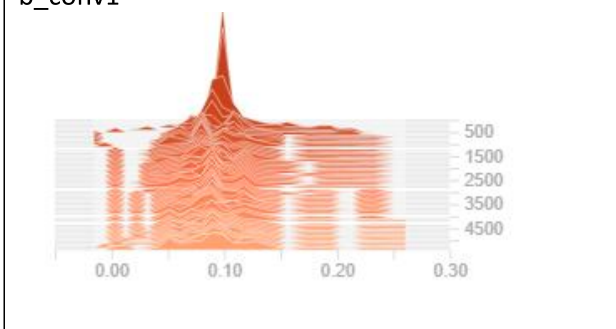


Histogram Plots:

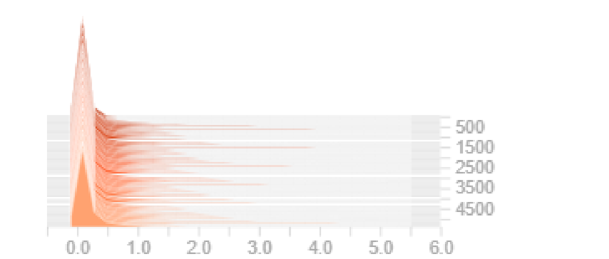
W_conv1



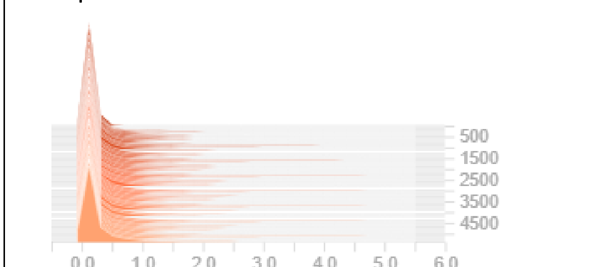
b_conv1



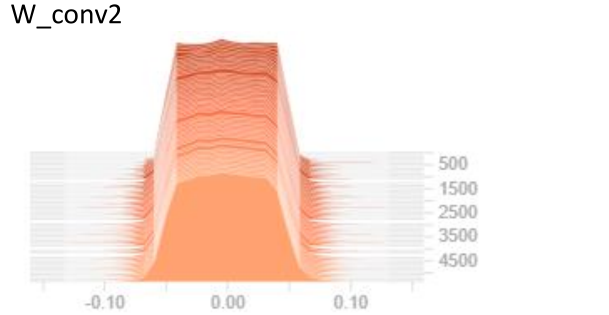
After relu h_conv1



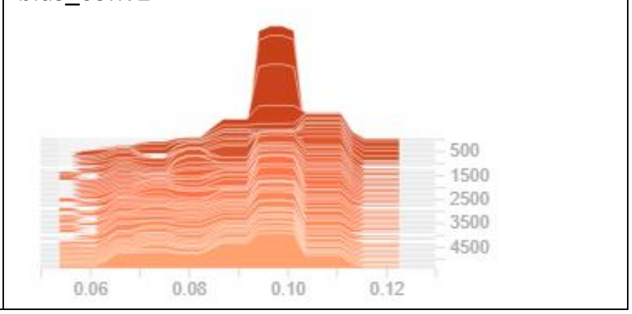
Max pool1



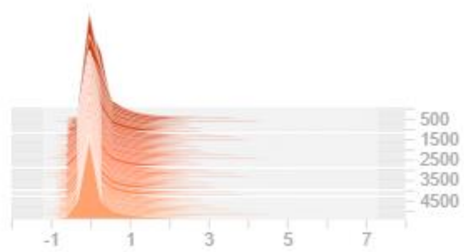
W_conv2

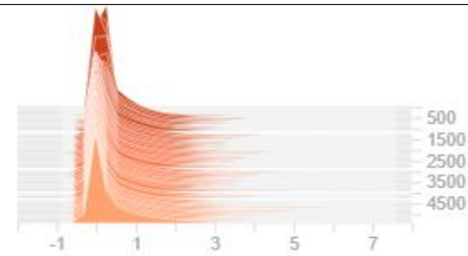


bias_conv2



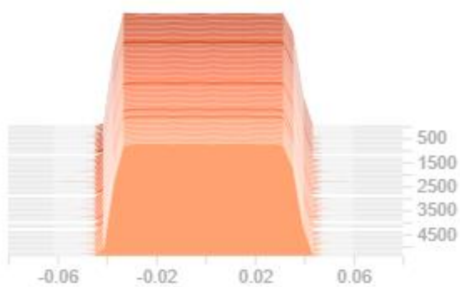
After relu conv2



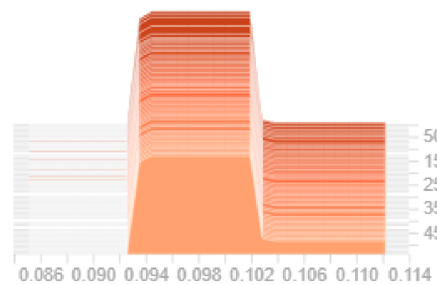


Max pool conv2

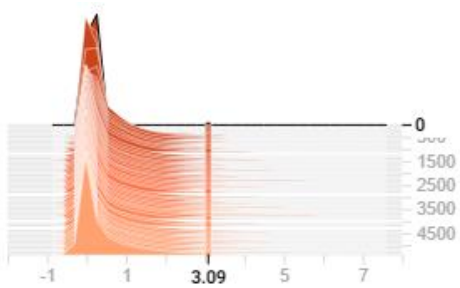
Dense layer weight



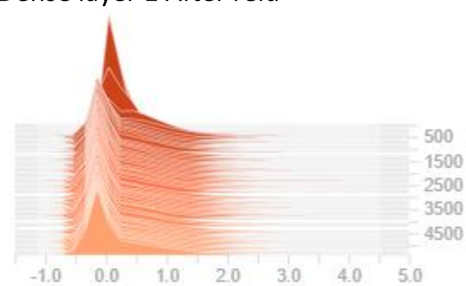
Dense layer biase

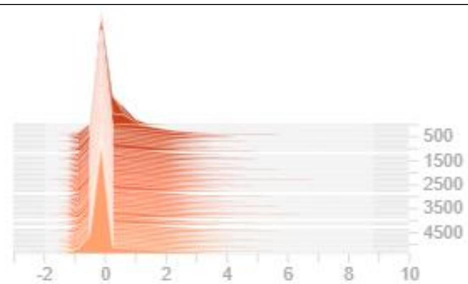


Dense layer h_pool2 flat



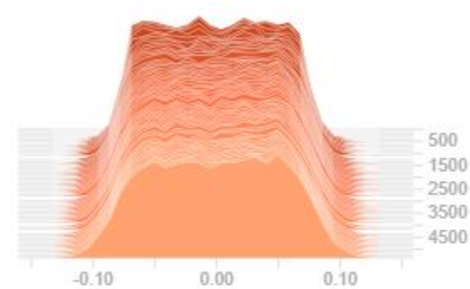
Dense layer 1 After relu



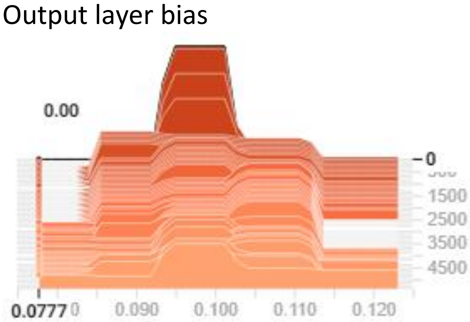


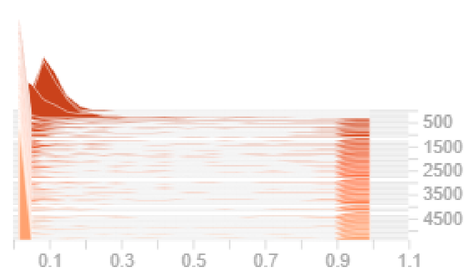
Dropout layer h_fc1_drop

Output layer weight



Output layer bias





Output layer Y_conv

2.1 Resources I used to work on this HW:

1. Introduction to Deep Learning Lectures
2. Multi-Layer Neural Network
3. How the backpropagation algorithm works
4. Mysteries of Neural Networks Part III
5. Neural Networks Tutorial – A Pathway to Deep Learning