Deep Learning: Homework #2

Due on November 4, 2019 at 11:55pm

Professor Emad Fatemizadeh

Amirhossein Yousefi

97206984

Practical Excercise 1

Part1:

Initializing neural networks weights yields two flaws.**Firstly** as all weights start with same initial values,they end up with same gradients and so they do same thing which leads to decreasing the capacity and ability of the neural network as a global approximator function.**Secondly** as neural networks tend to stuck in local minima, in order to find the best of them, it is better to have different starting points but zero initializing prevent us from doing that.Actually zero initializing hinder the network as it cause symmetry in network which as mentioned above decline the the capacity of network in complexity space.After implementing network you can see below the computational graph of the network.

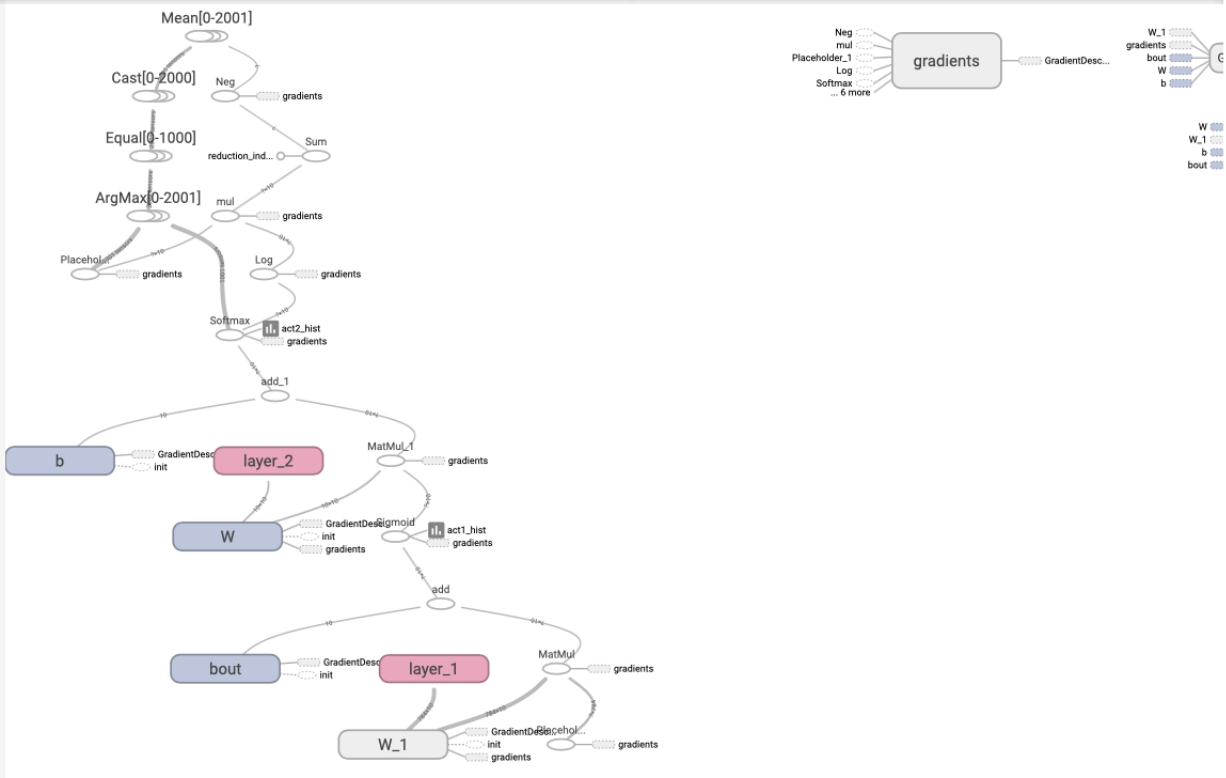
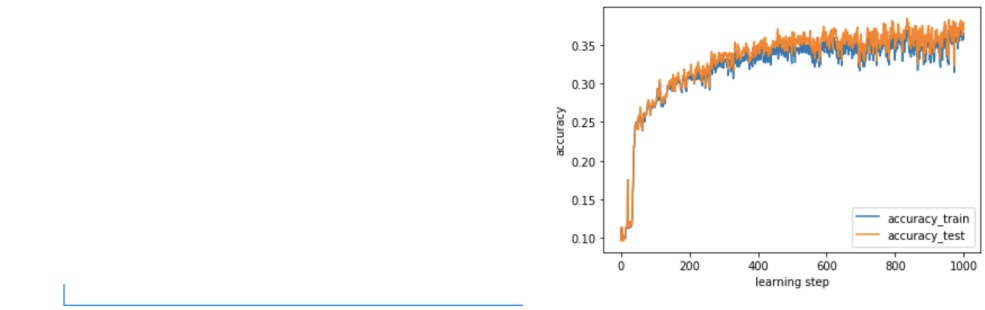


Figure 1: Computational graph

In the following network implemented by different initial values for weights and test error and train error is reported.What is inferred from below figure is that as we decrease the variance of weight,we reach better accuracy and loss.Firstly accuracy for zero initializing is plotted below.Then for initializing by different variance of gaussian distribution as initial weights diagram is plotted.

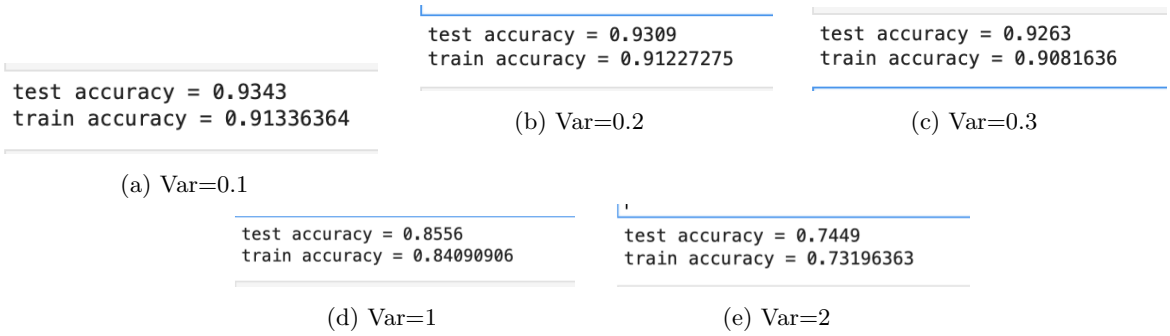


(a) accuracy

(b) accuracy

Figure 2: Zero init results

After that results for different value of variances are plotted.It is obvious that the less the variance the better their corresponding accuracy are.



(a) Var=0.1

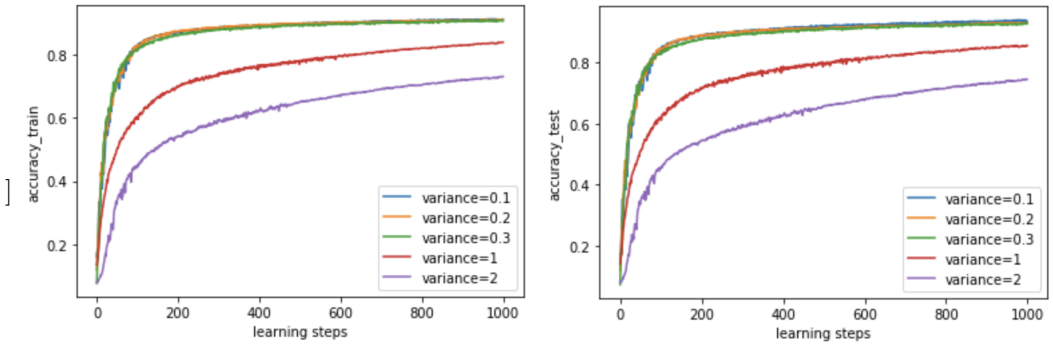
(b) Var=0.2

(c) Var=0.3

(d) Var=1

(e) Var=2

Figure 3: Results



(a) train accuracy

(b) test accuracy

Figure 4: Results

As mentioned best accuracy has reached when **variance=0.1** and as we increase variance both test and train accuracy decreased.

Problem 1 continued on next page...

Part 2&3:

In this part former results is monitored by **tensorboard** include weights, activation output , bias and train and test loss during learning steps.**Vanishing gradients** cause difficulty in knowing which direction should parameters go in order to improve the cost function, while **exploding gradients** make learning unstable.In other words vanishing gradients cause flaw in process of back propagation and learning process.More generally In deep networks the gradients tend to get smaller as we keep on moving backward in the network so neurons in the earlier layers learn very slowly compared to the neurons in the latter layers in the hierarchy and as earlier neurons are responsible to learn and detect simple patterns or in other words are building blocks of the network,they play important role in learning procedure.So generally it's better that in deep networks relu will be used as an activation function.So whenever very small changes or no change happen in histogram of weight we can conclude that they vanished.Vanishing Gradients is particularly problematic in with sigmoid activation function as when the input of sigmoid function is high or low the derivative is small that causes vanishing gradient which occurs when initializing is poor.For example initial weights are large negative or positive that saturates the input to sigmoid and pushes the derivatives into small value.As we used sigmoid in first hidden layer so we may face vanishing in that layer.For example in figure7 for histogram of weights in layer1 we faced with vanishing gradient as no change happened during training procedure.It is obvious from figure9 that activation function saturated as its reached to 1.In saturation we should note that it happens whenever we stuck in point zero value or one value in sigmoid and -1 and $+1$ value in tanh activation function where gradient vanished.Saturation also happened in figure13 and 16 for layer1 activation function.Vanishing also occurred in figure18 and 20 for weight of layer1.

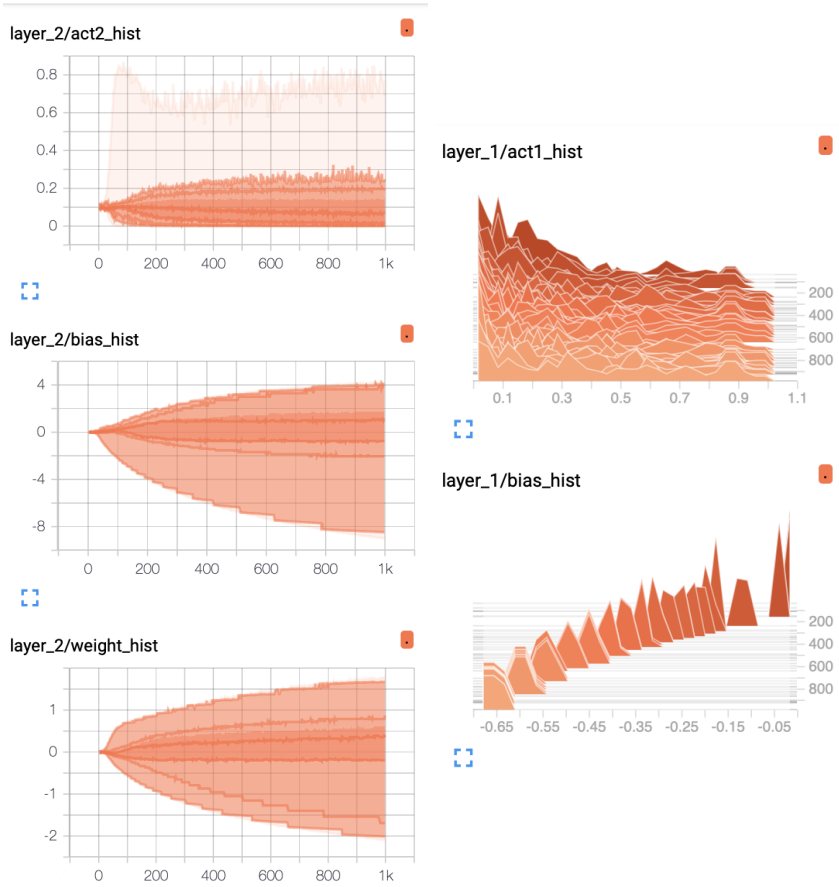


Figure 5: Results for zero initializing

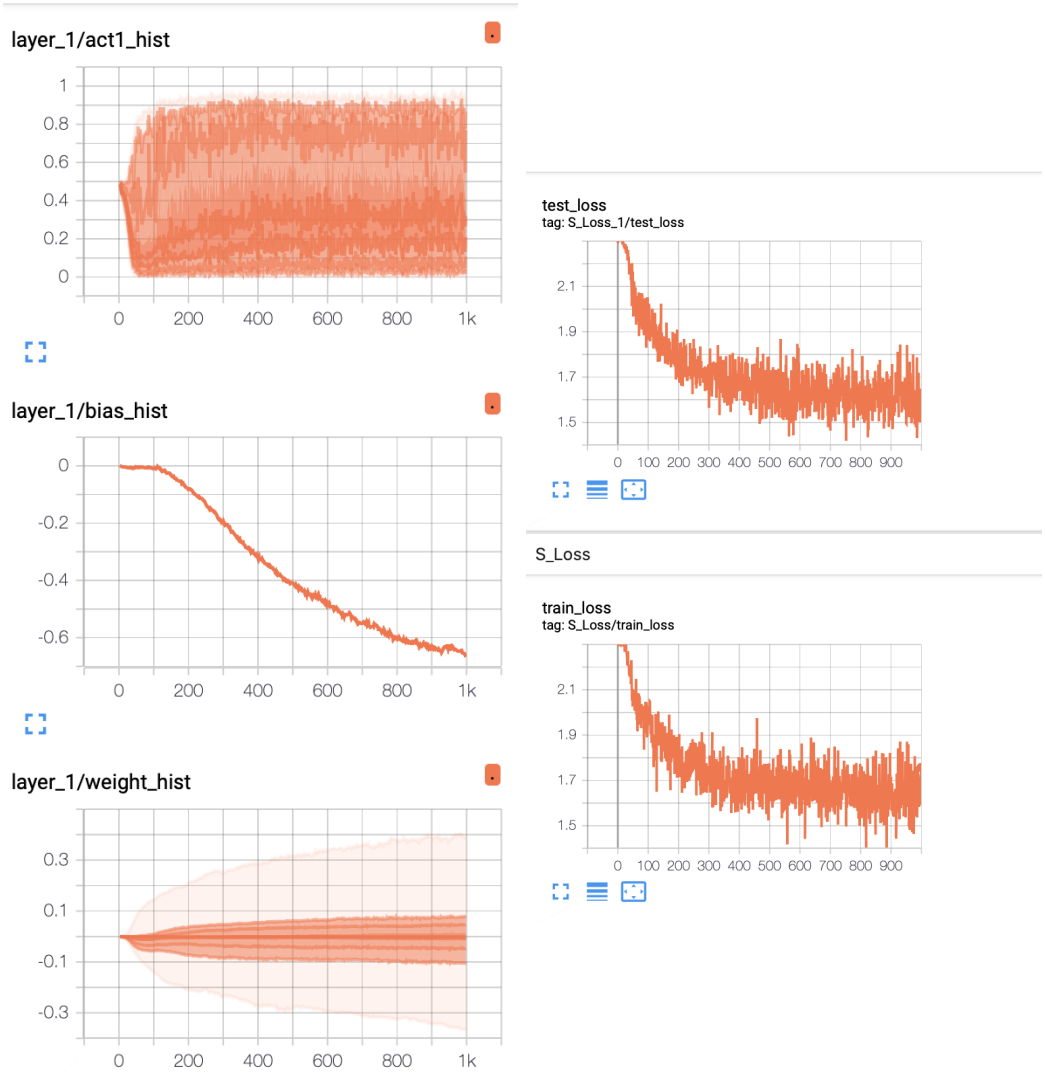


Figure 6: Results for zero initializing

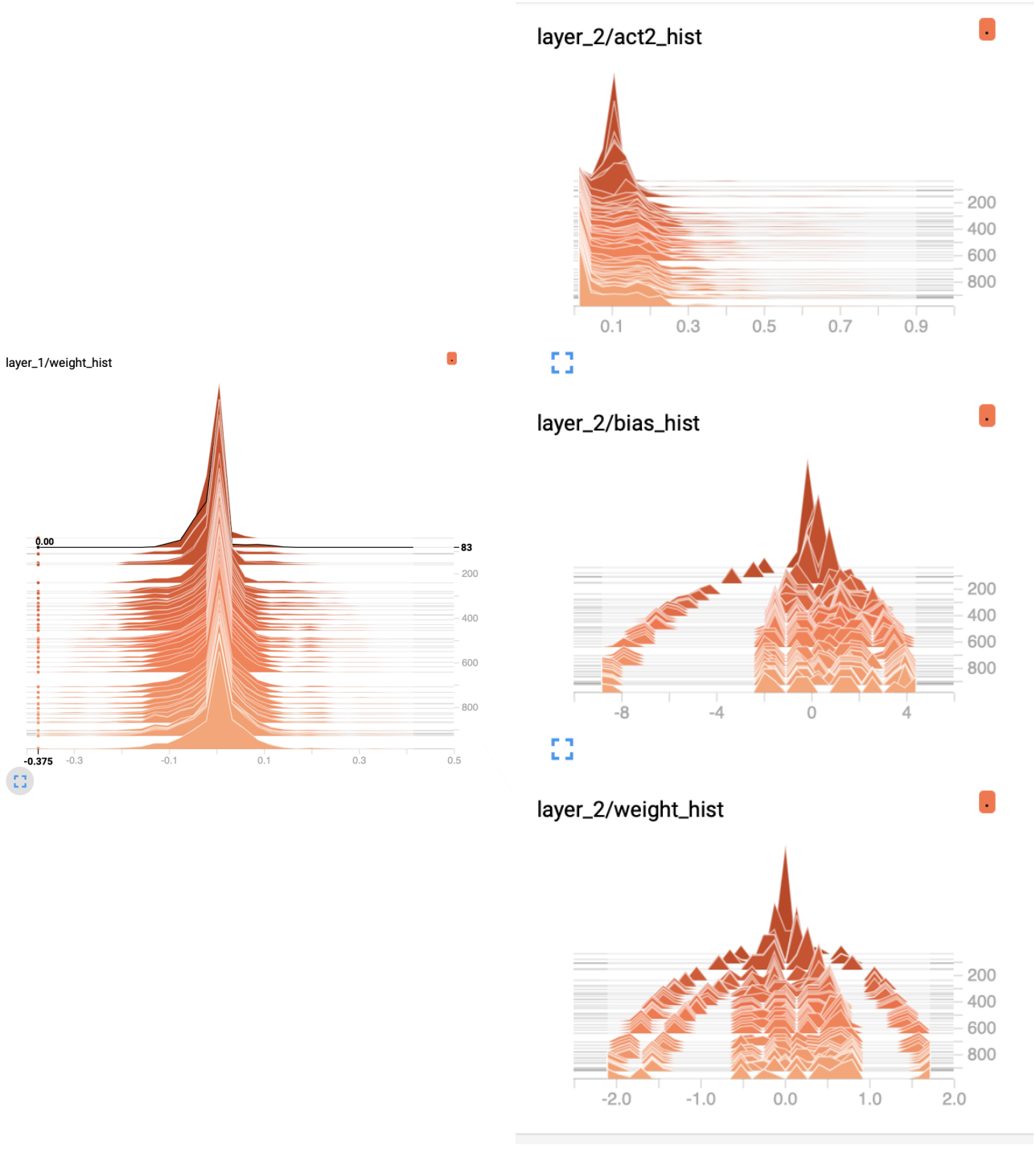


Figure 7: Results for zero initializing

Problem 1 continued on next page. . .

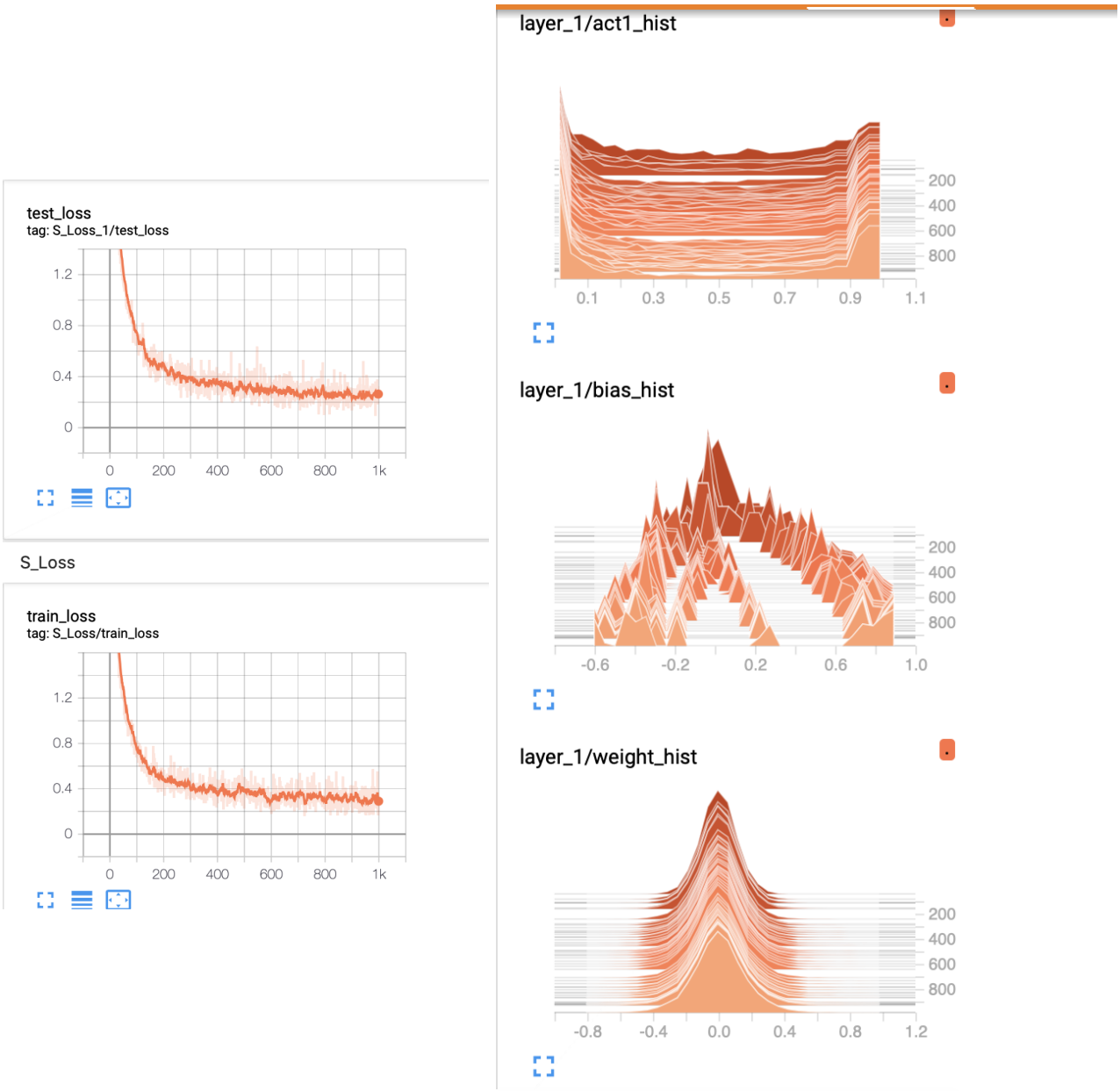


Figure 8: Results for Var=0.1

Problem 1 continued on next page. . .

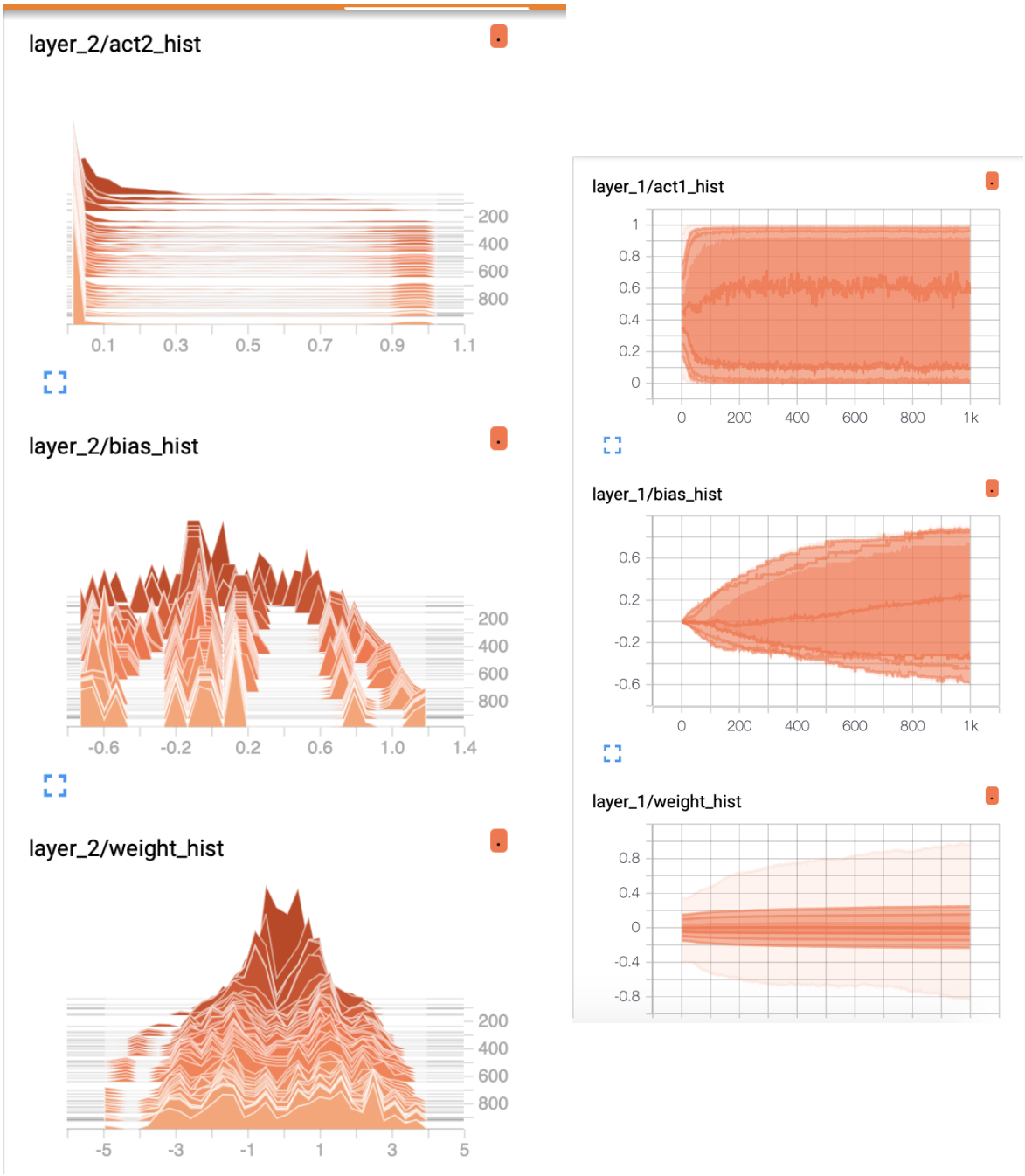


Figure 9: Results for Var=0.1

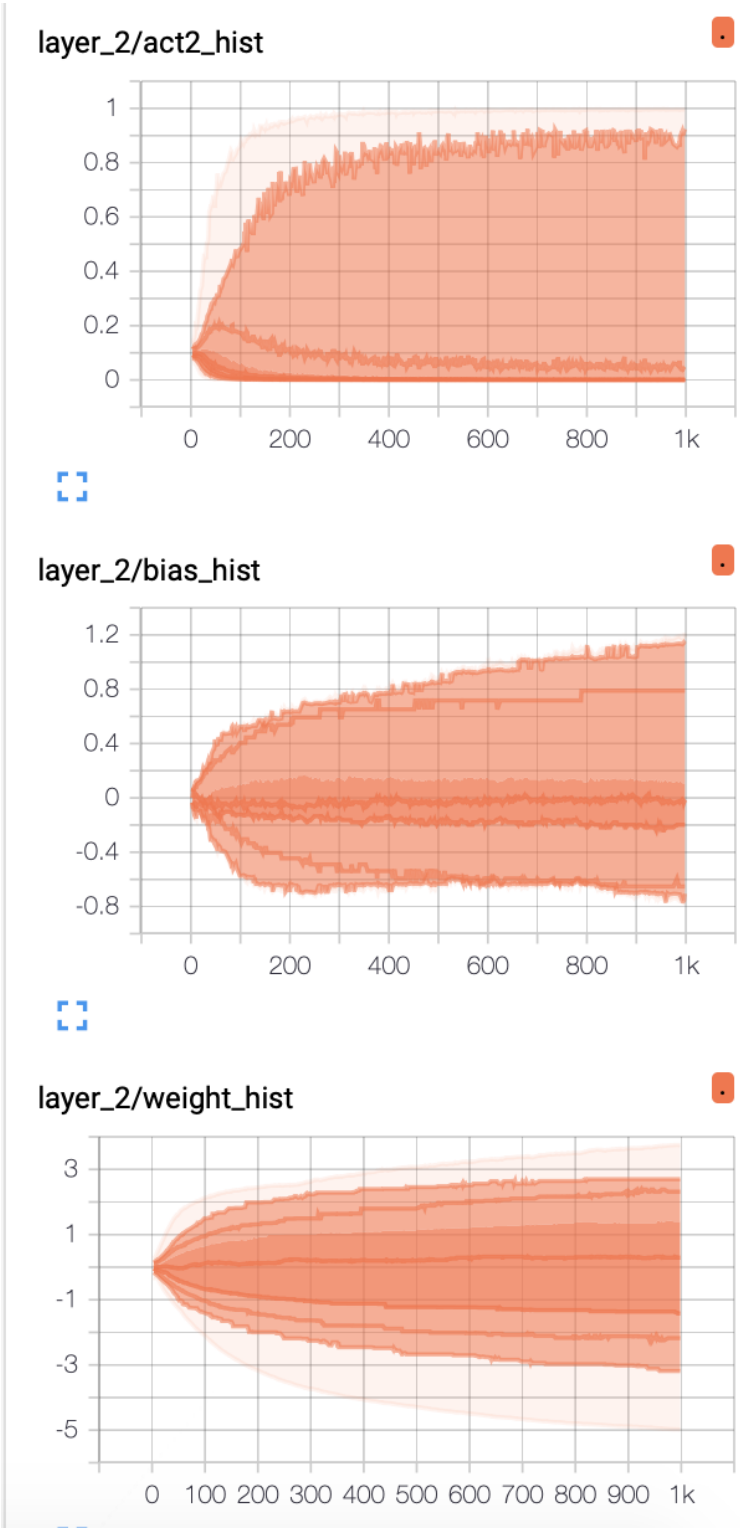


Figure 10: Results for Var=0.1

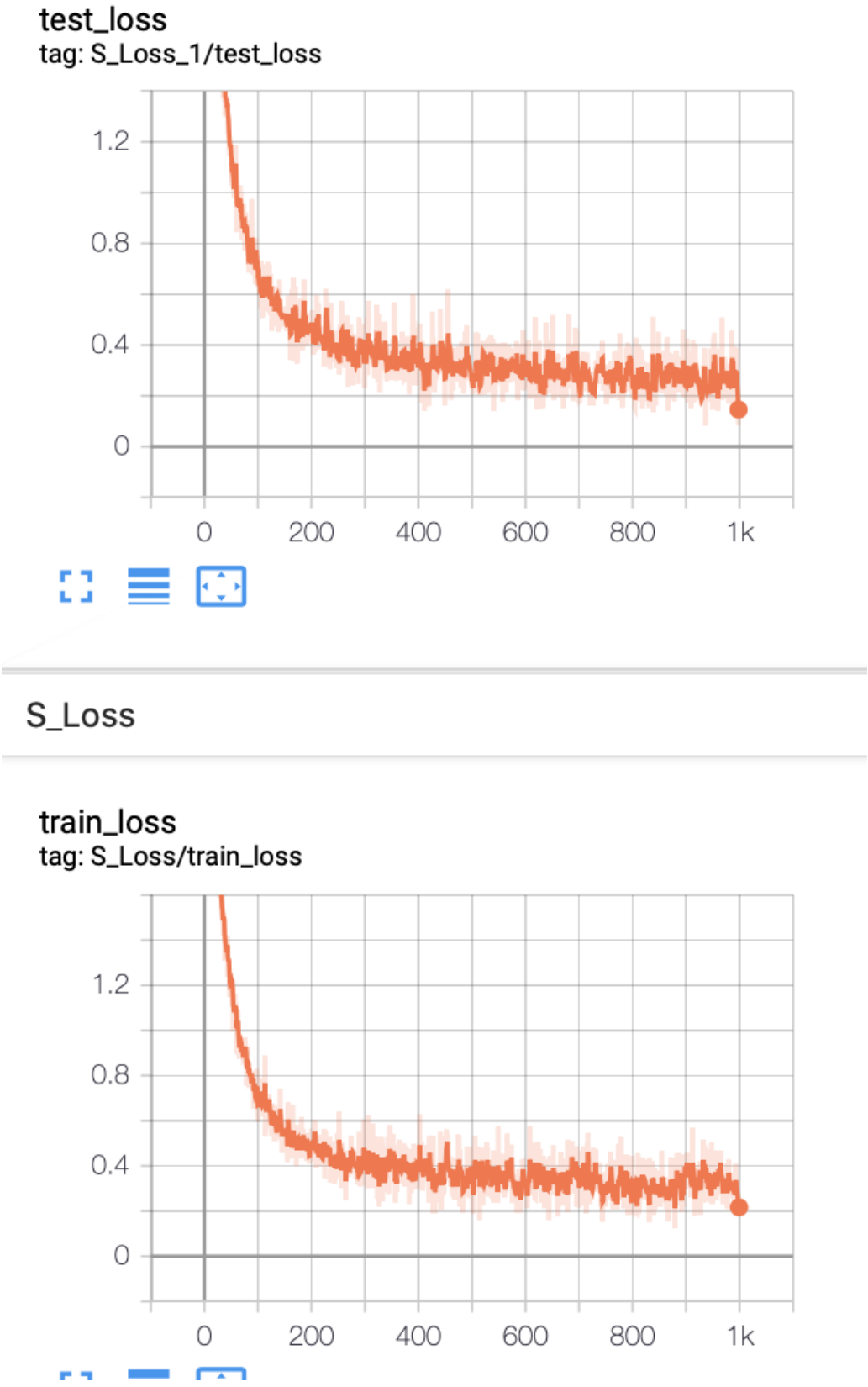


Figure 11: Results for Var=0.2

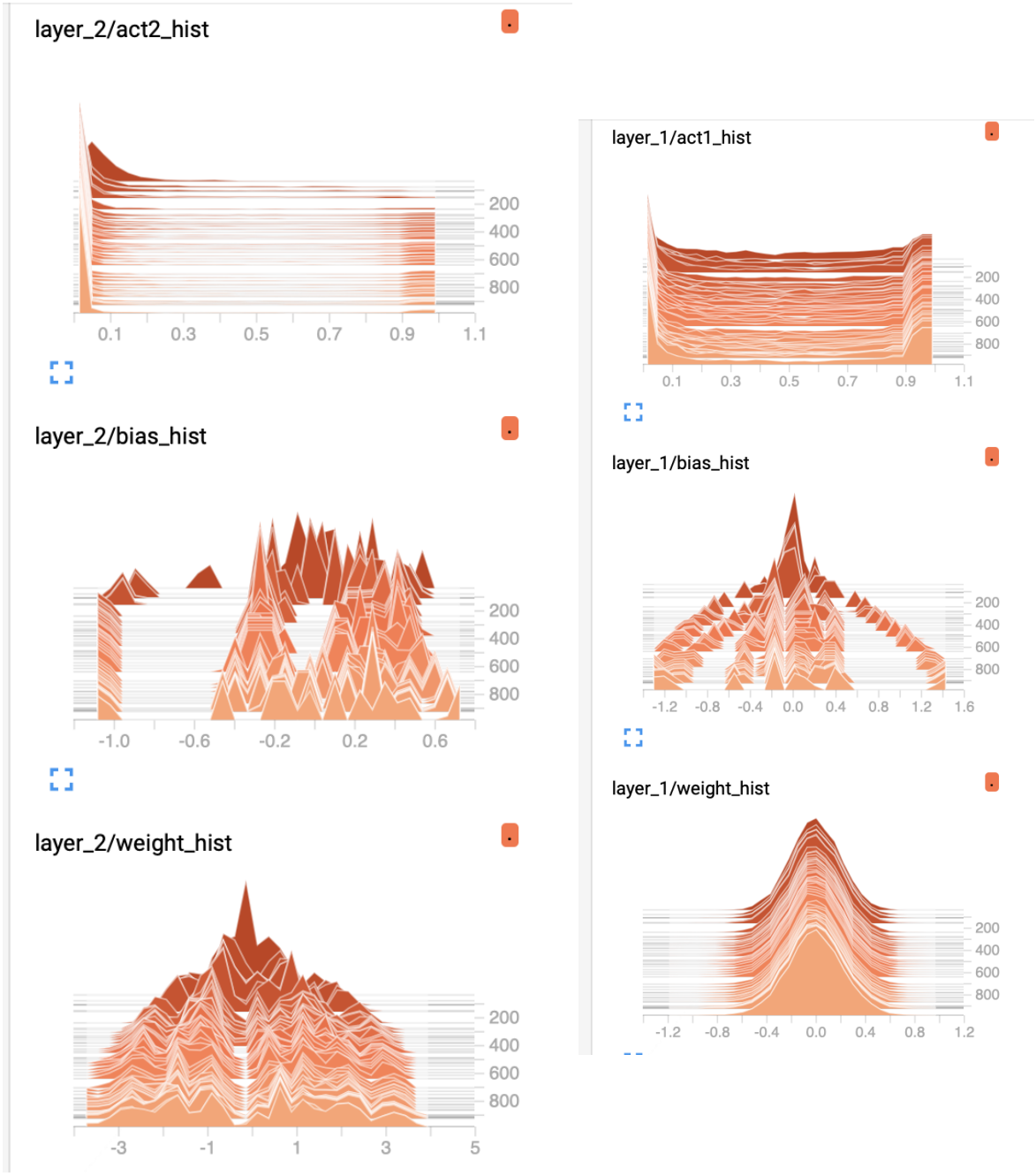


Figure 12: Results for Var=0.2

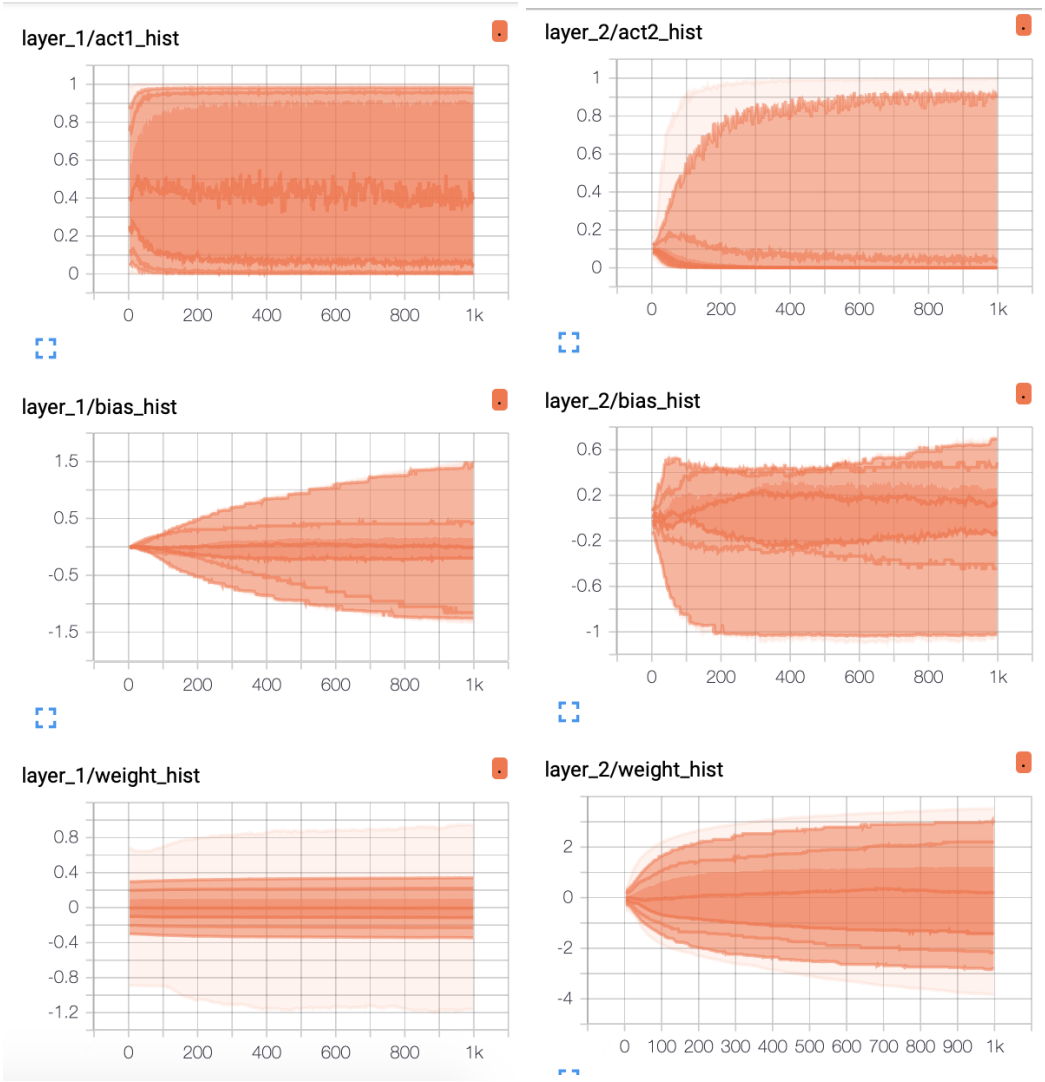


Figure 13: Results for Var=0.2

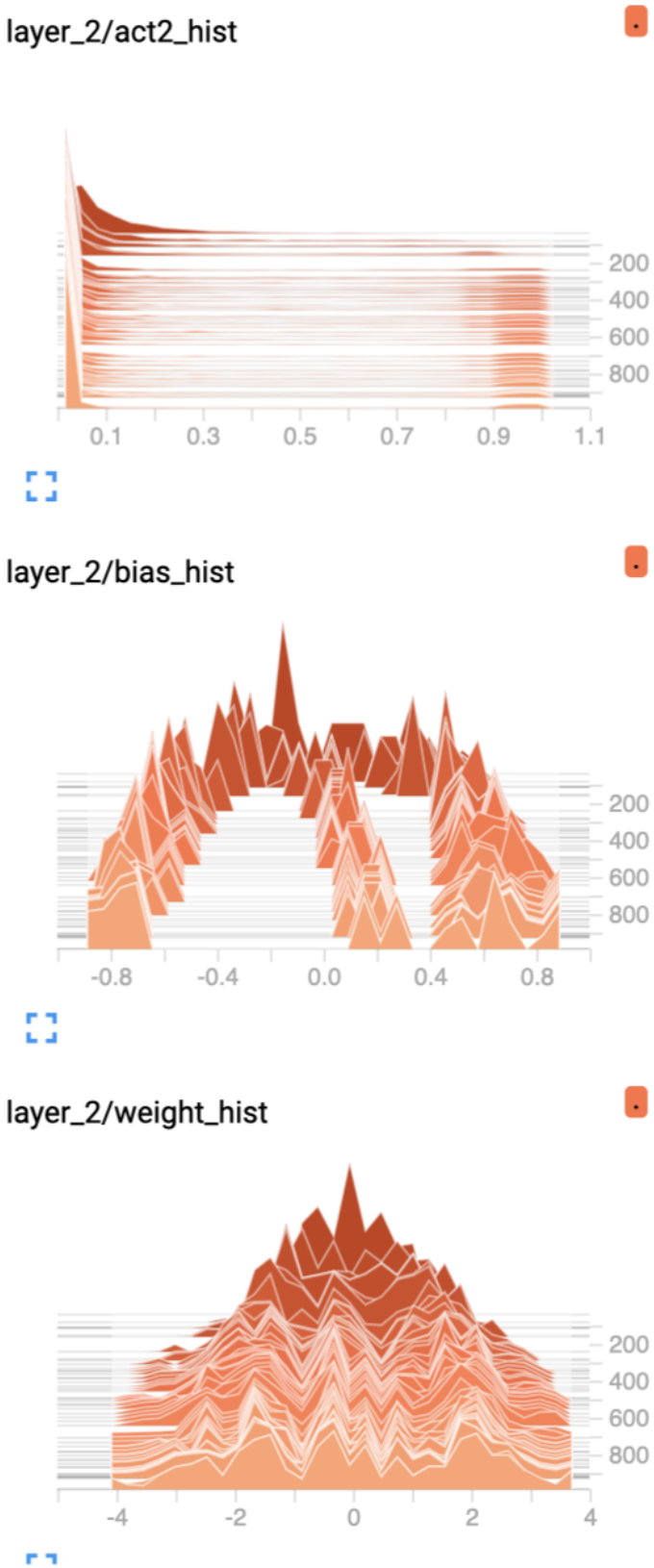


Figure 14: Results for Var=0.3

Problem 1 continued on next page...

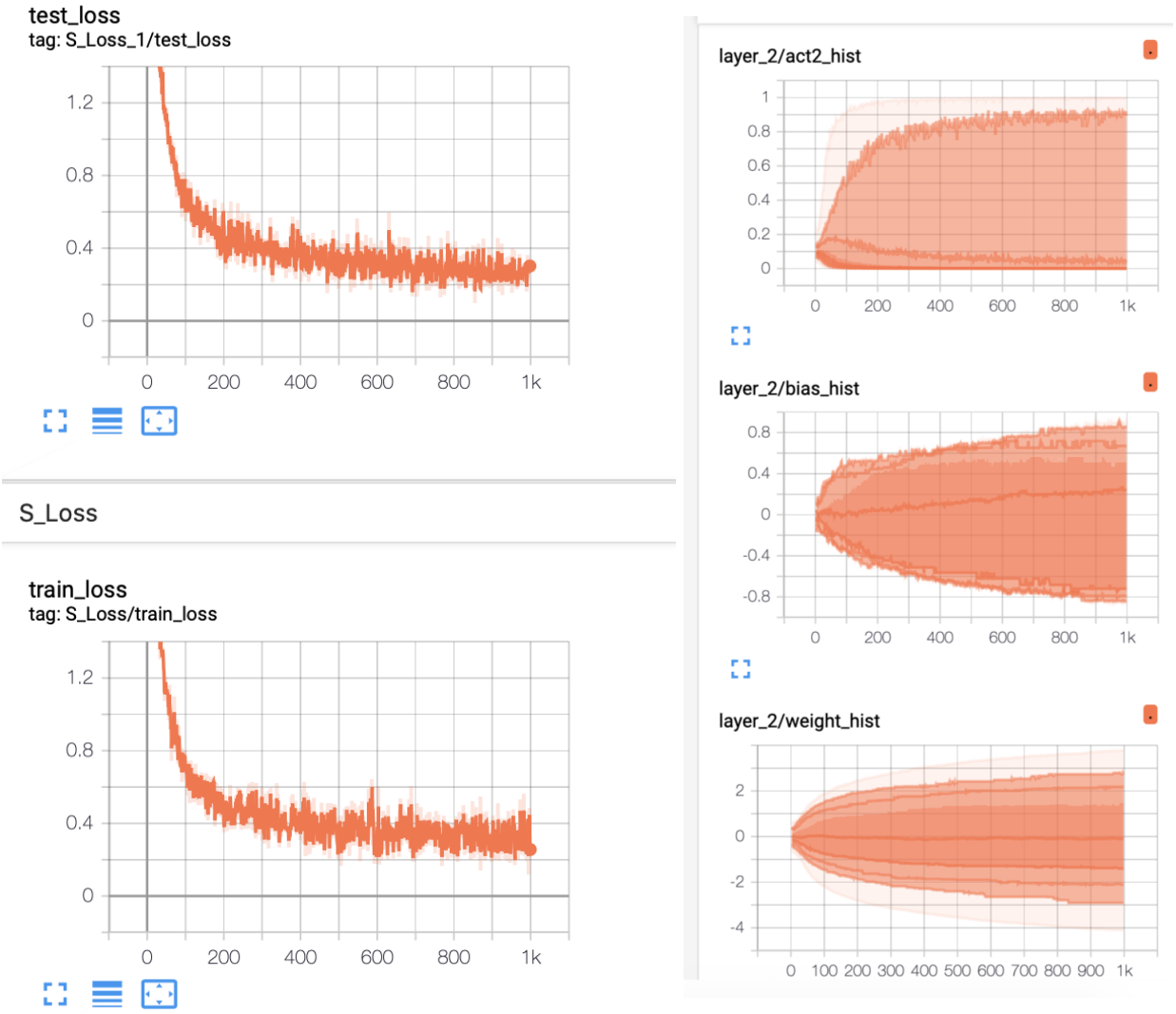


Figure 15: Results for Var=0.3

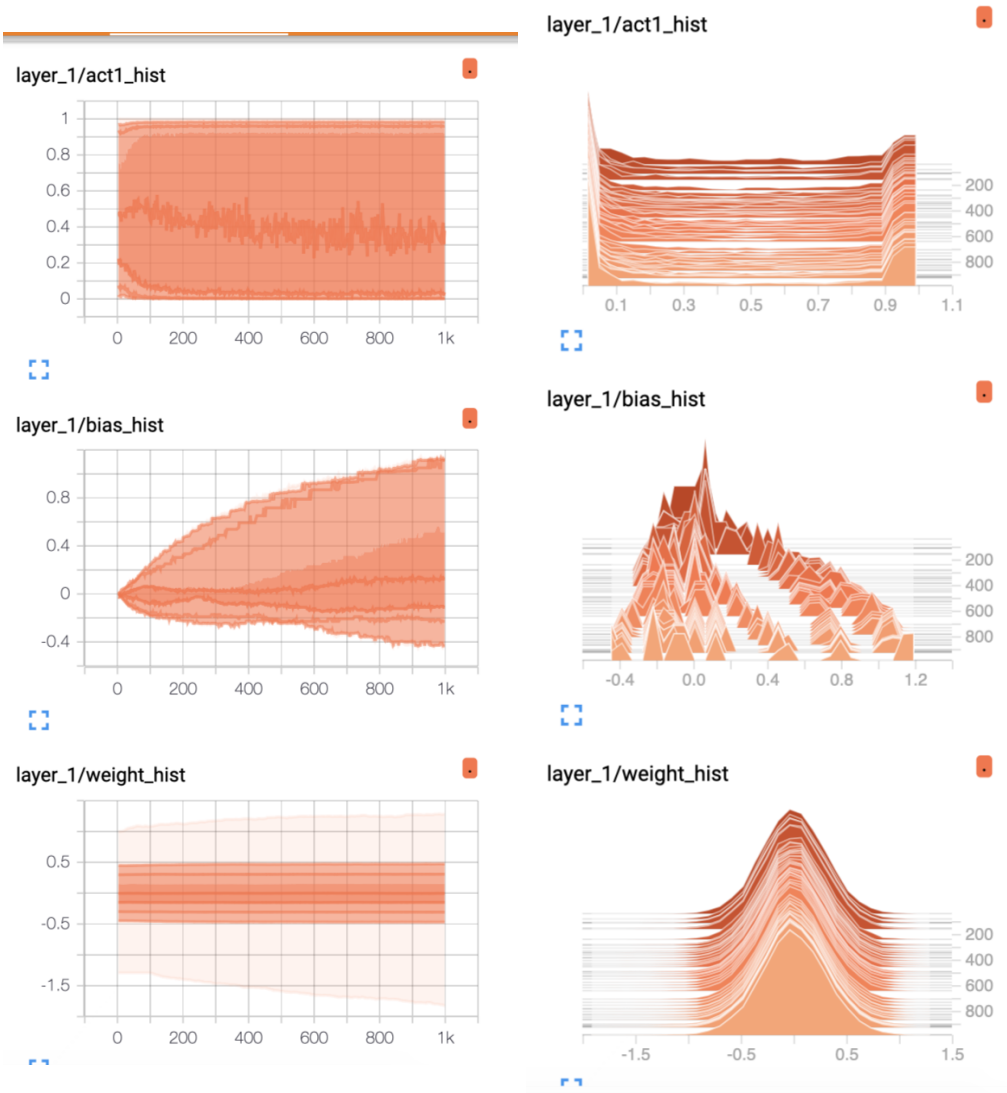


Figure 16: Results for Var=0.3

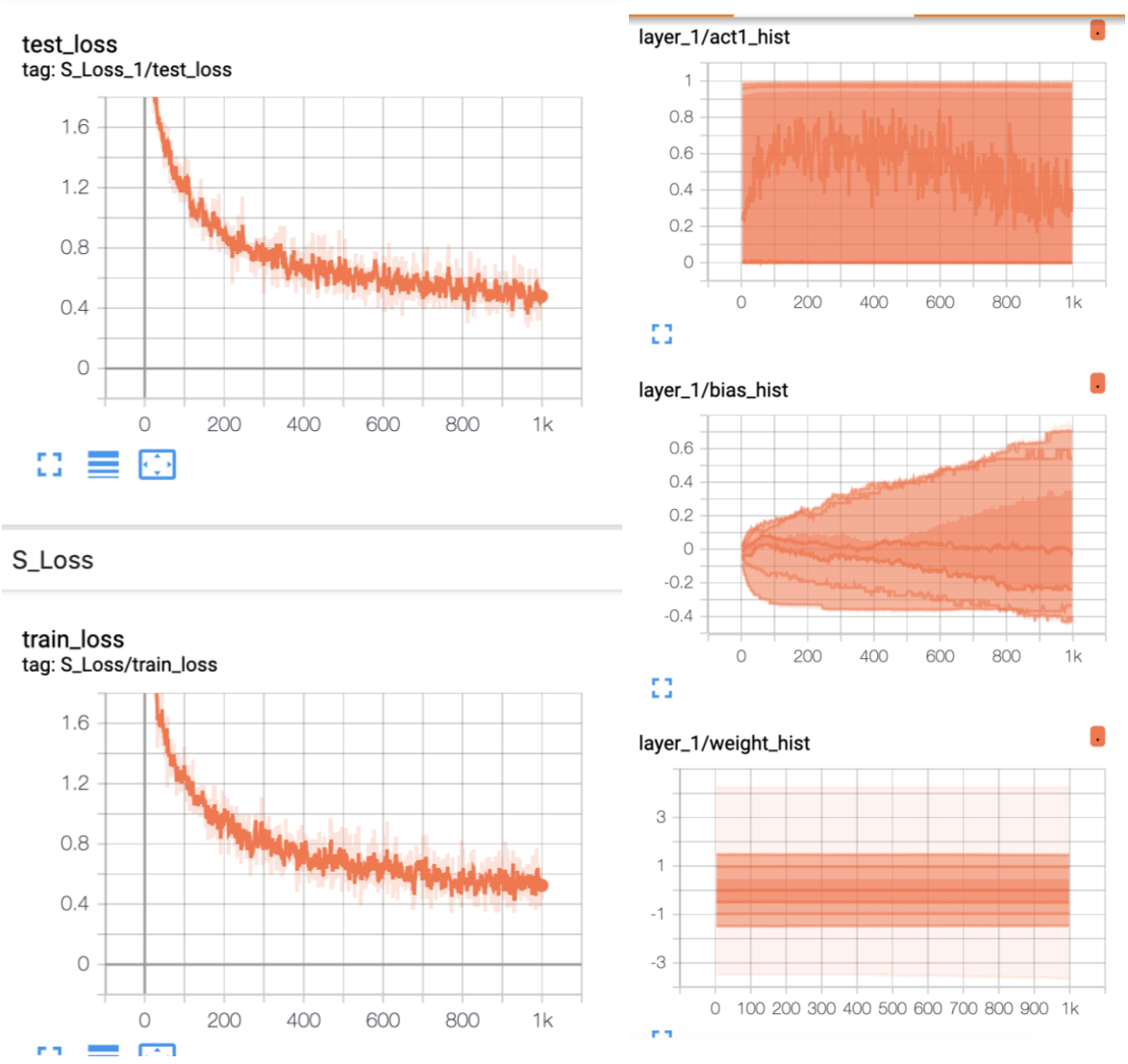


Figure 17: Results for Var=1

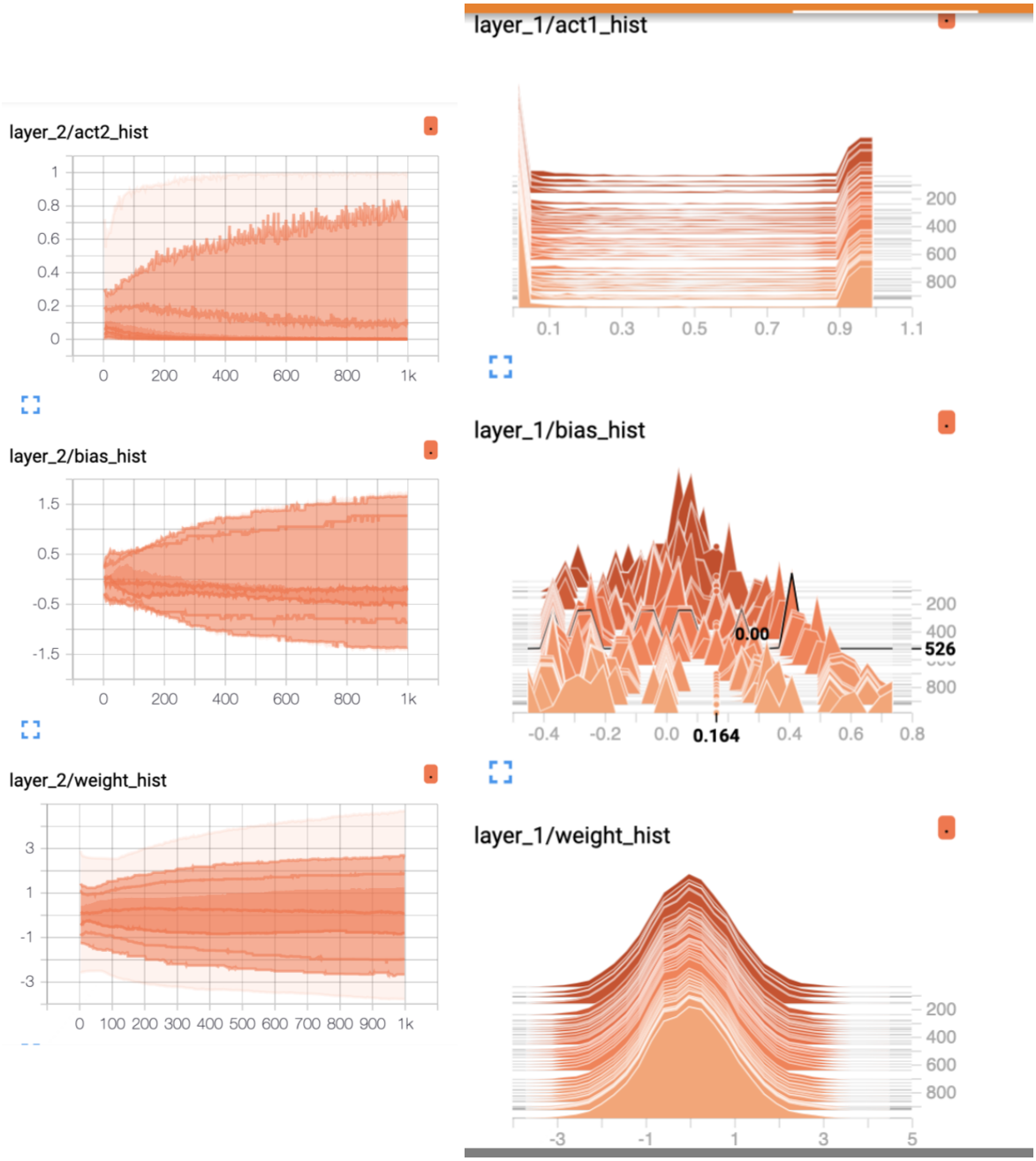


Figure 18: Results for Var=1

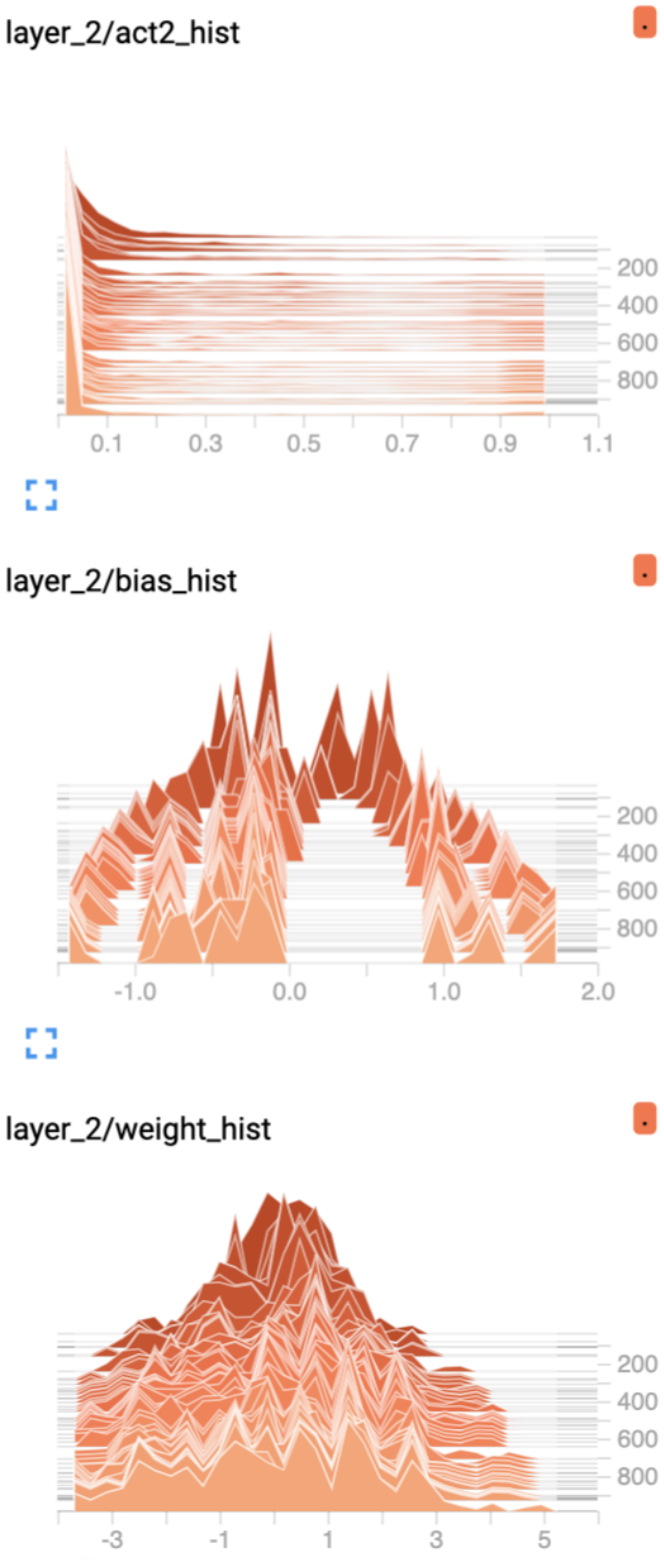


Figure 19: Results for Var=1

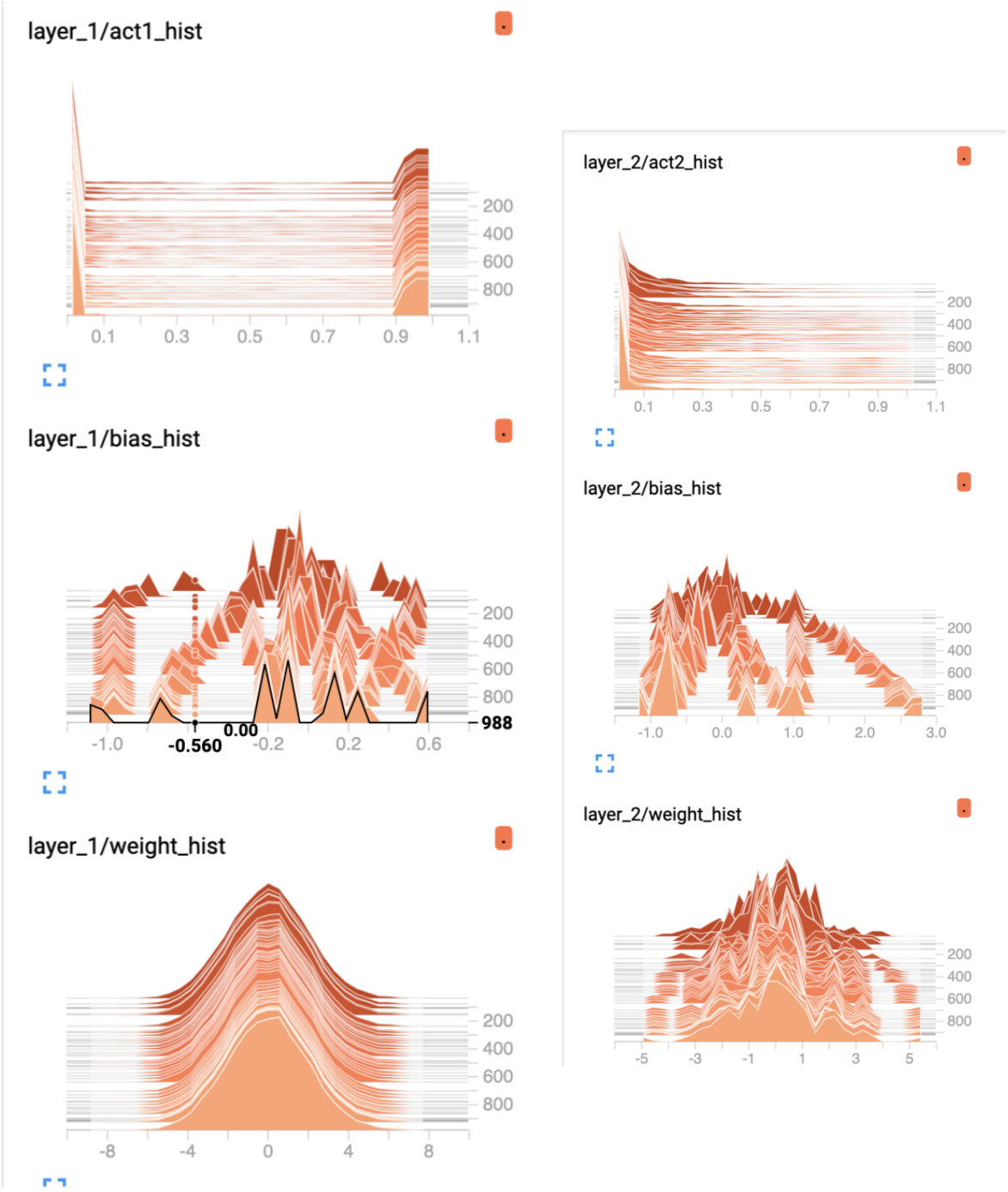


Figure 20: Results for Var=2

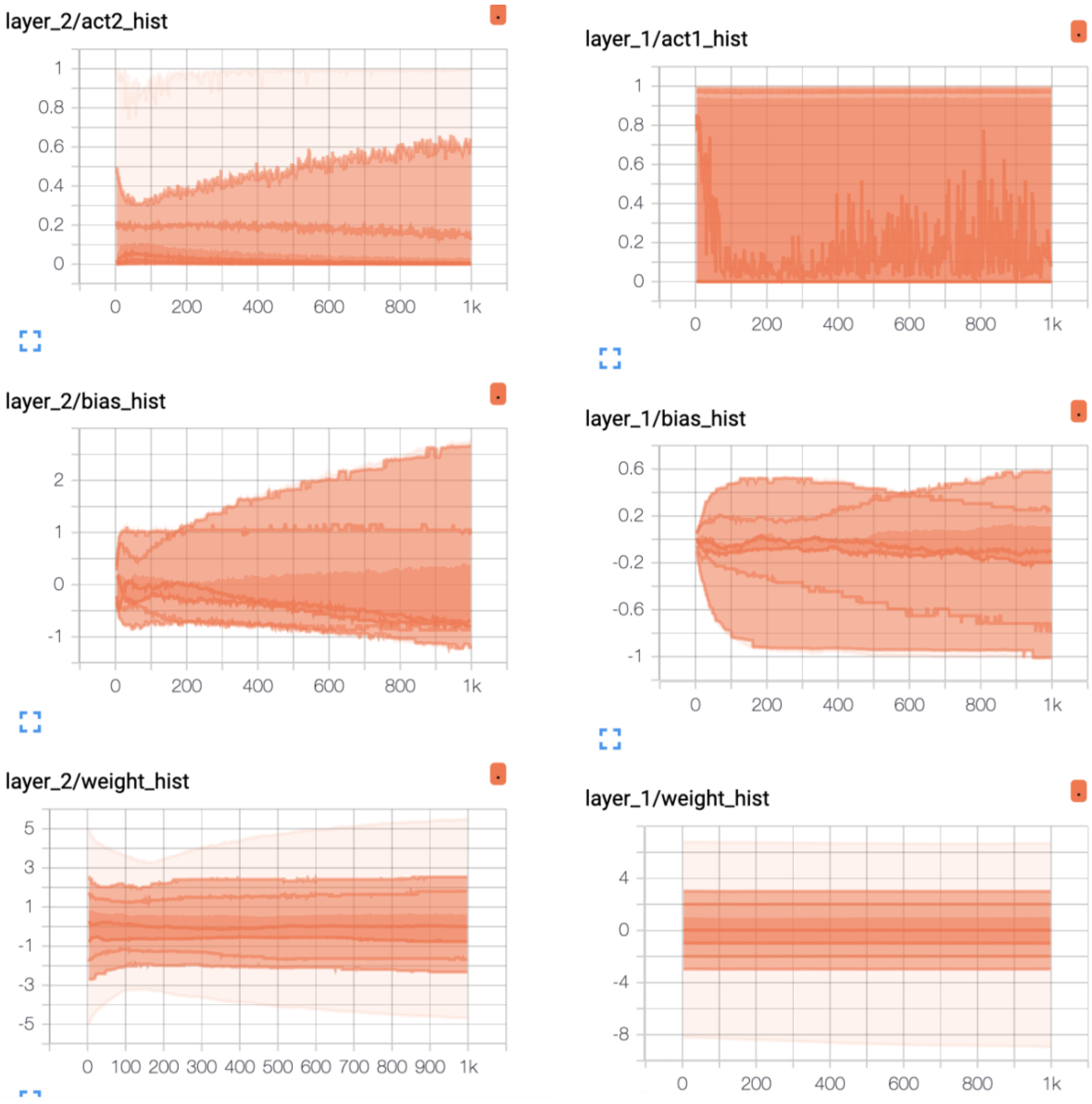


Figure 21: Results for Var=2

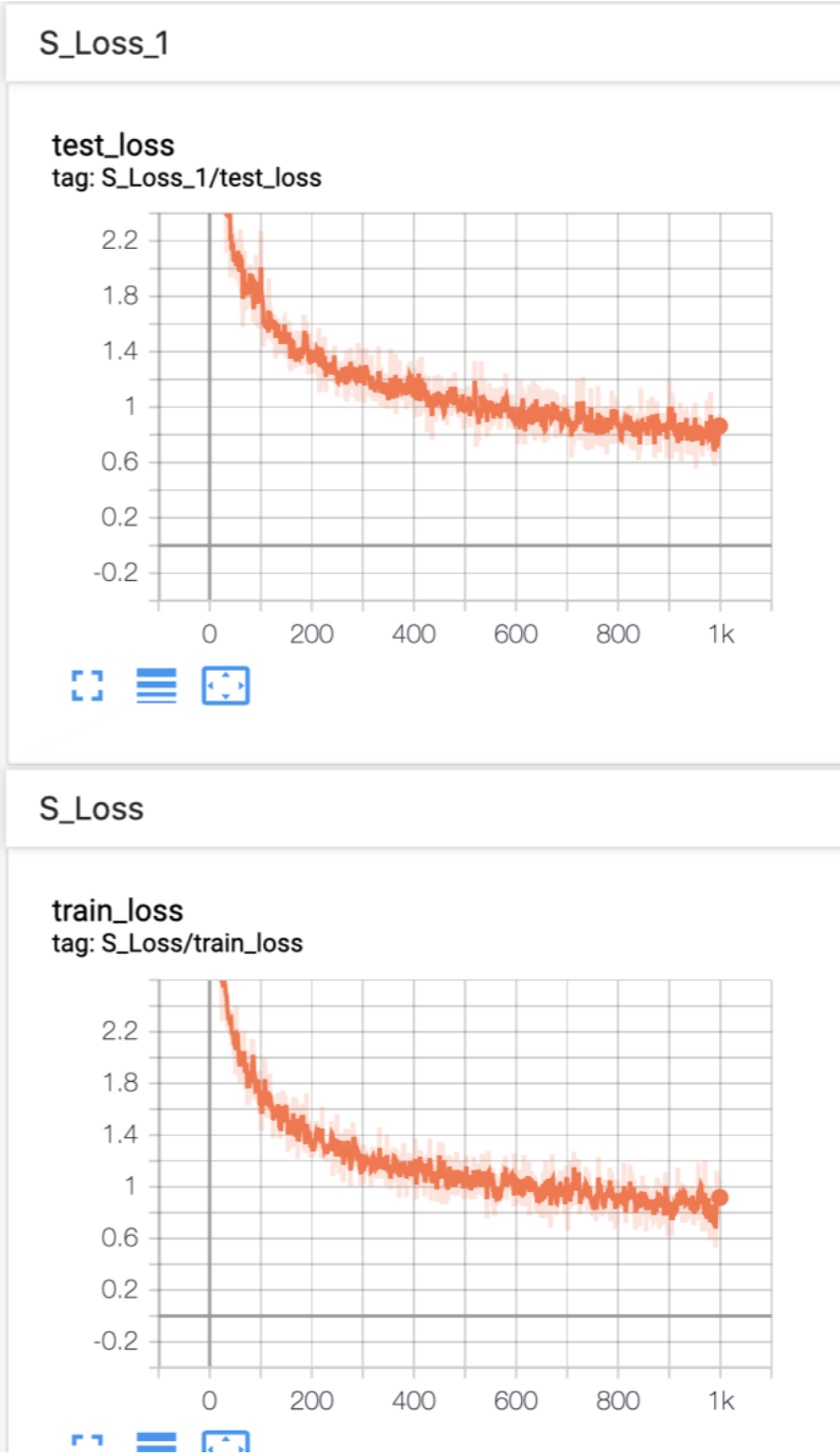
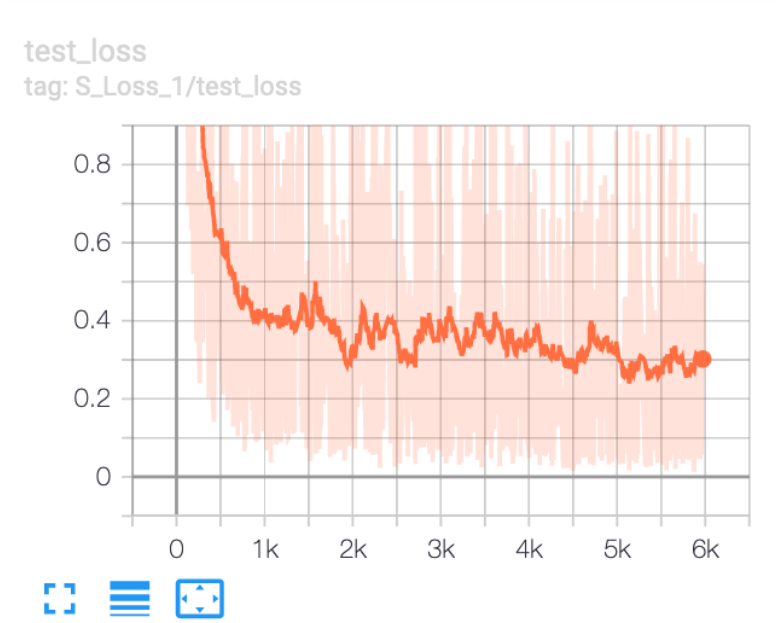


Figure 22: Results for Var=2

Part 4:

Stochastic gradient descent is one of the stochastic methods in optimization that solve the problem of occupying huge memory for big data and also computation load for calculating gradient by batching the data so the expectation of this method is gradient descent.Full batch suffers from occupation of memory and computation load but we sure that in each step we always approach to optimum point and as we see all data and the estimation is more accurate as we increased the quantity of samples which leads to reduction in variance of estimation and we approached to the true value.As we decrease the batch size ,the estimation is more noisy or in other word we have redundancy.Small batches also offer a regularizing effect since they add noise to the process of learning so as we have high variance in estimating the gradient,it's better to have small learning rate in order to prevent sudden change.**Generalization error** is often best for $batchsize = 1$. One drawback of mini batch is the total runtime or learning steps that is needed in the process of learning since in mini batch it takes time that model see all data but in full batch as network see all data,we need less runtime or learning step or some times we may move in the opposite direction of optimum point for mini batch as we may had a bad estimation of the gradient.It's clear from below figures that as batch size increased ,loss fluctuate less as the estimation is less noisy and more accurate so the sudden changes also decreased and also less step is needed to train the network.

Figure 23: Results for batch size=1




```
test loss is 0.026715297 step= 5998
test loss is 0.32983297 step= 5999
test accuracy = 0.9104
train accuracy = 0.9117636
```

S_Loss

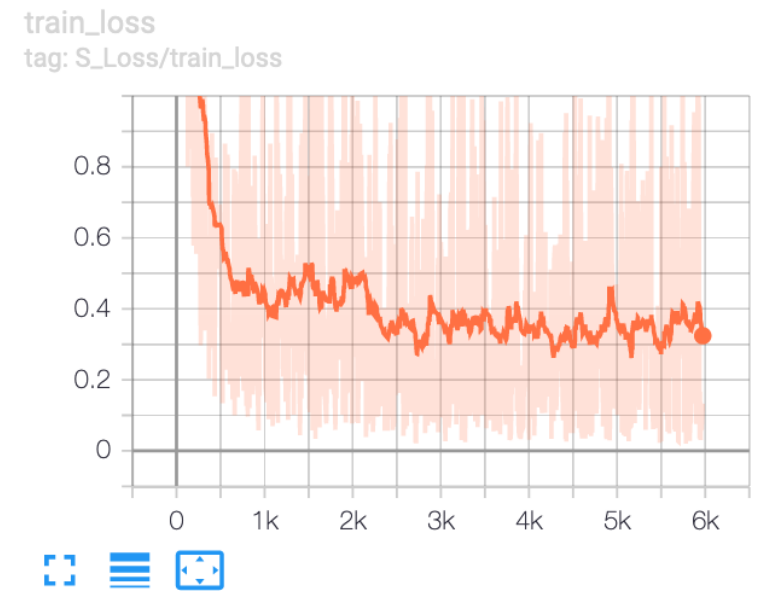
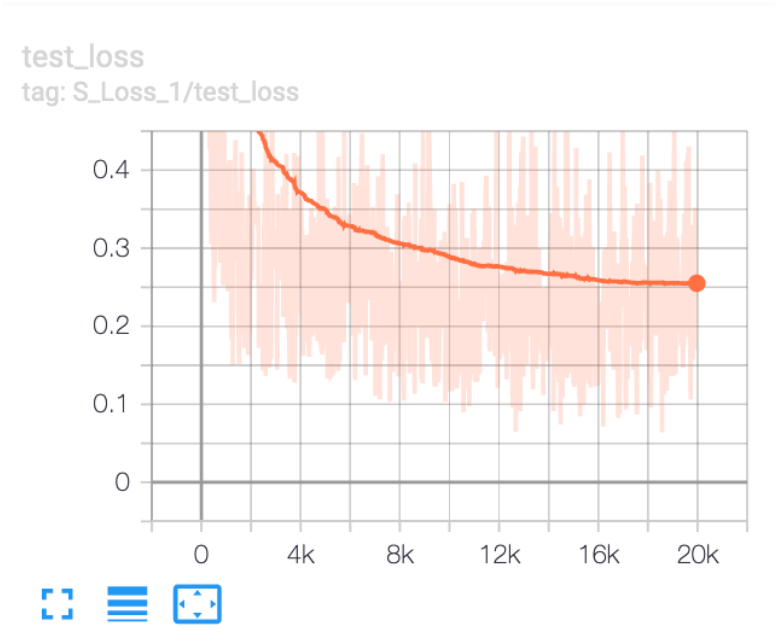


Figure 24: Results for batch size=10

Figure 25: Results for batch size=50




```
test loss is 0.16634327 step= 19994
test loss is 0.32917732 step= 19995
test loss is 0.37900642 step= 19996
test loss is 0.23781954 step= 19997
test loss is 0.27300686 step= 19998
test loss is 0.18335791 step= 19999
test accuracy = 0.9296
train accuracy = 0.94225454
```

S_Loss

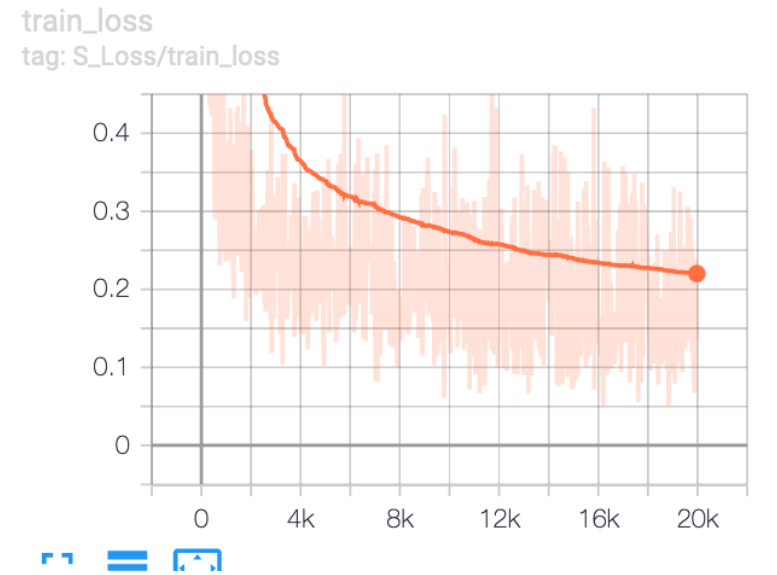
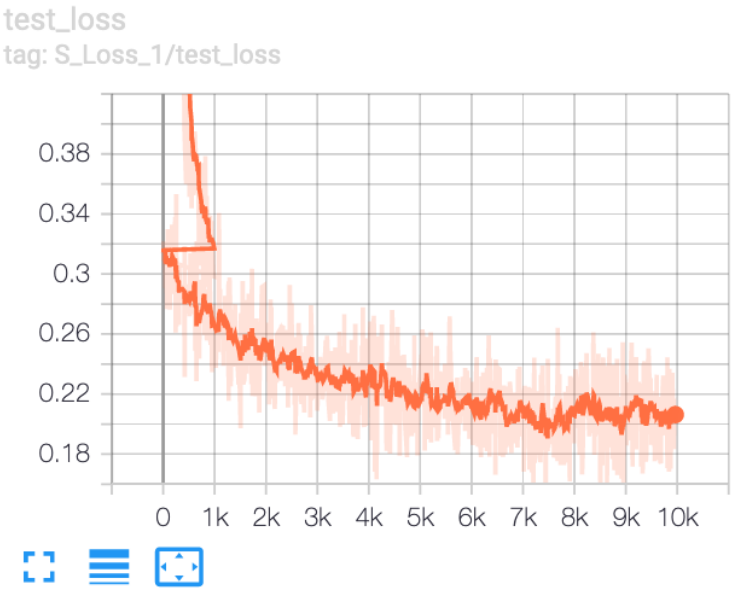


Figure 26: Results for batch size=100




```
test loss is 0.17873728 step= 9994
test loss is 0.21535185 step= 9995
test loss is 0.23105453 step= 9996
test loss is 0.21110499 step= 9997
test loss is 0.2304419 step= 9998
test loss is 0.18471295 step= 9999
test accuracy = 0.9403
train accuracy = 0.9495091
```

S_Loss

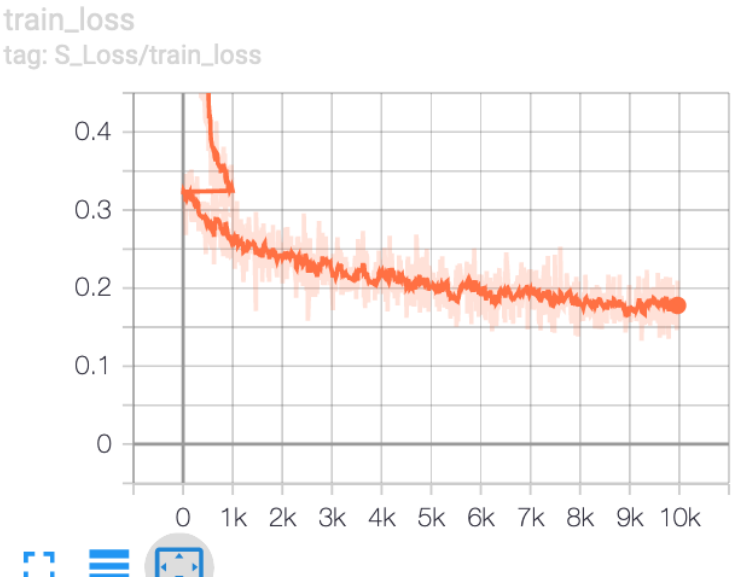


Figure 27: Results for batch size=1000

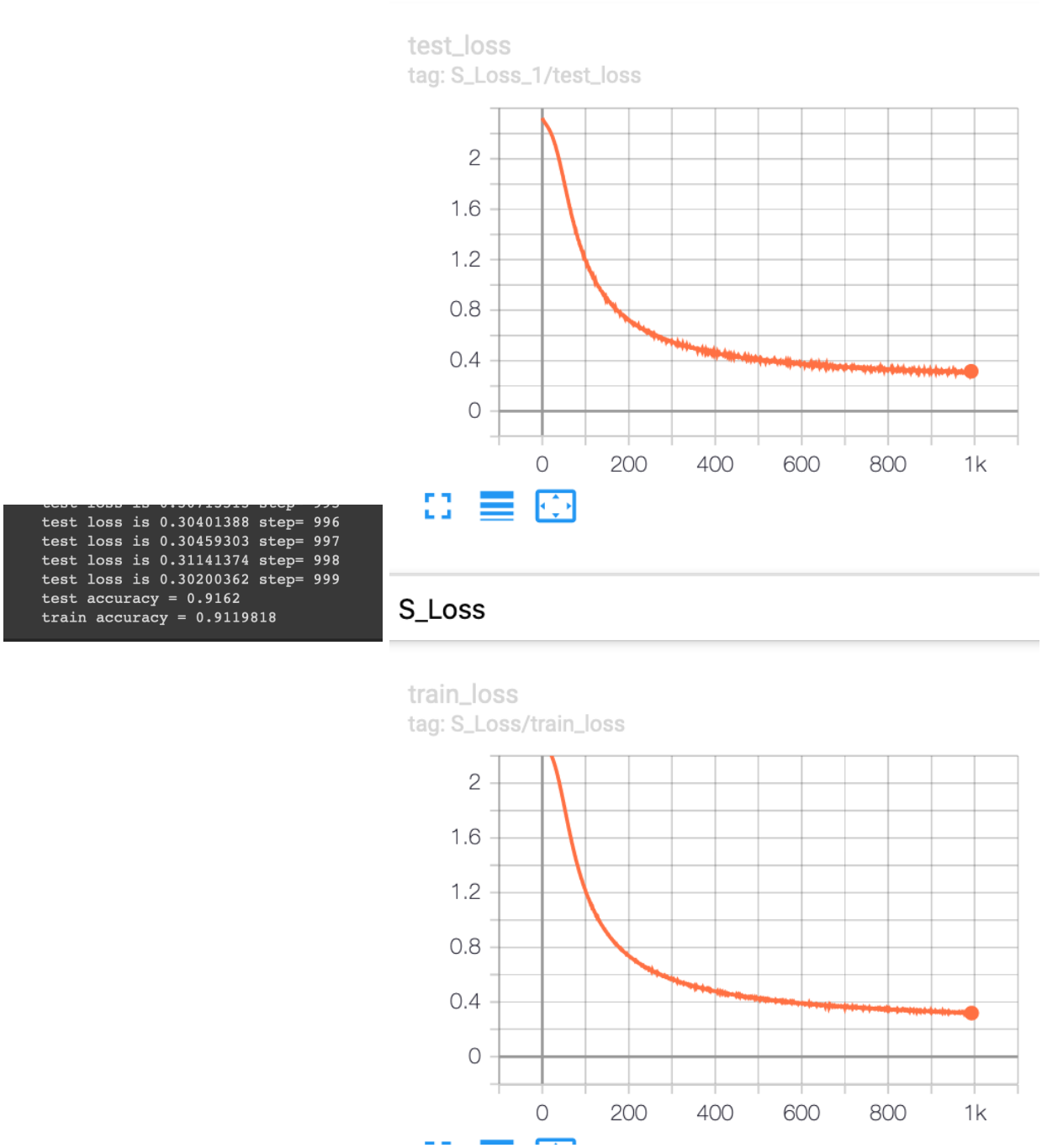


Figure 28: Results for batch size=55000(full batch)

At the end best accuracy achieved for batch size=1000 according to learning steps that we assumed that such that learning process is done completely.

Part 5:

Increasing learning rate causes faster change in weights in other words we emphasize on gradient term rather than former value of weights.Conversely if we decrease the learning rate we do not allow sudden and great change in weights in each iteration.Convergence condition is to decrease learning rate in each step to satisfy below condition. $\sum learning\ rate = \infty$ and $\sum (learning\ rate)^2 < \infty$. In the following cost for different learning rates has been plotted. We can see from the figures that small learning rate causes slowness in the process of learning as we depend our updates to their former value rather than to gradient term which prevent weights from big changes in each step but our step is less noisy and also we need more learning step in order to reach to the optimum point.Conversely high learning rate cause more fluctuation and may lead us to diverge or passing the optimum point.In this task higher learning rate causes less training steps to train the network or in other word higher learning rate leads to higher rate of convergence as we can see in figure 30 for both train and test.

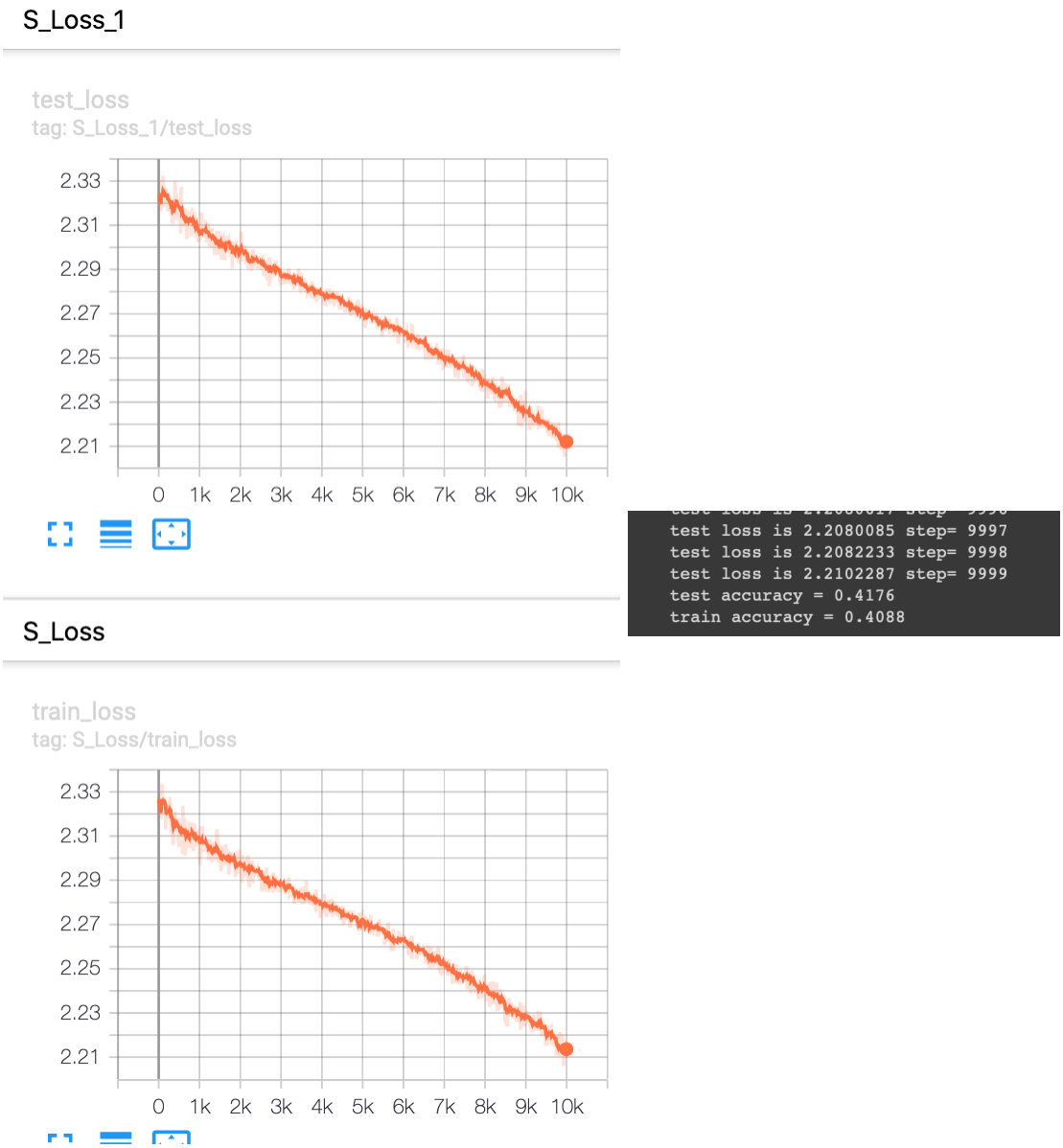


Figure 29: Results for batch size=1000 and learning rate=0.001

```
test loss is 1.1759219 step= 9997
test loss is 1.1690097 step= 9998
test loss is 1.1676729 step= 9999
test accuracy = 0.7239
train accuracy = 0.7188727
```

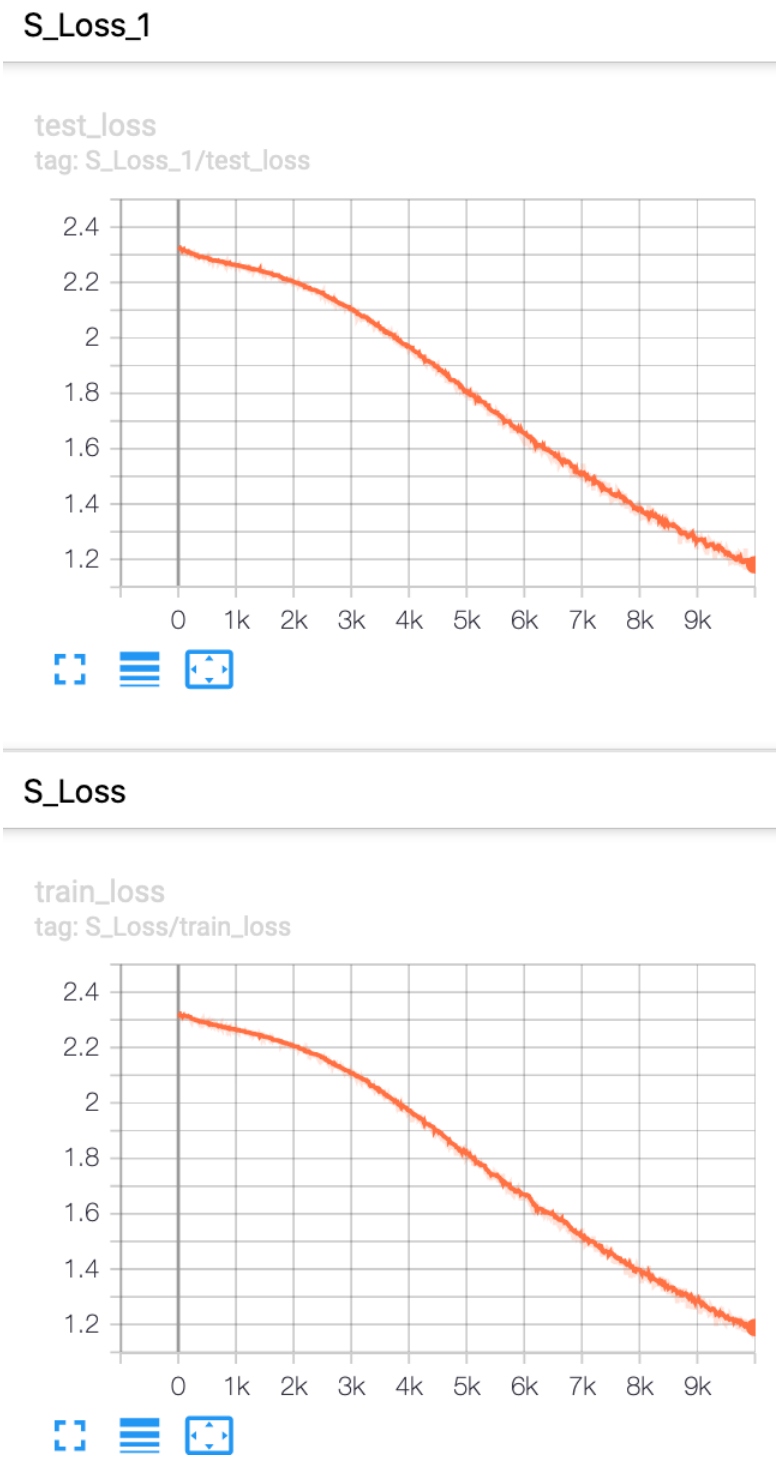


Figure 30: Results for batch size=1000 and learning rate=0.005

```
test loss is 0.6257734 step= 9992
test loss is 0.63524705 step= 9993
test loss is 0.675271 step= 9994
test loss is 0.7141918 step= 9995
test loss is 0.6499939 step= 9996
test loss is 0.6146231 step= 9997
test loss is 0.63178825 step= 9998
test loss is 0.6414 step= 9999
test accuracy = 0.853
train accuracy = 0.84145457
```

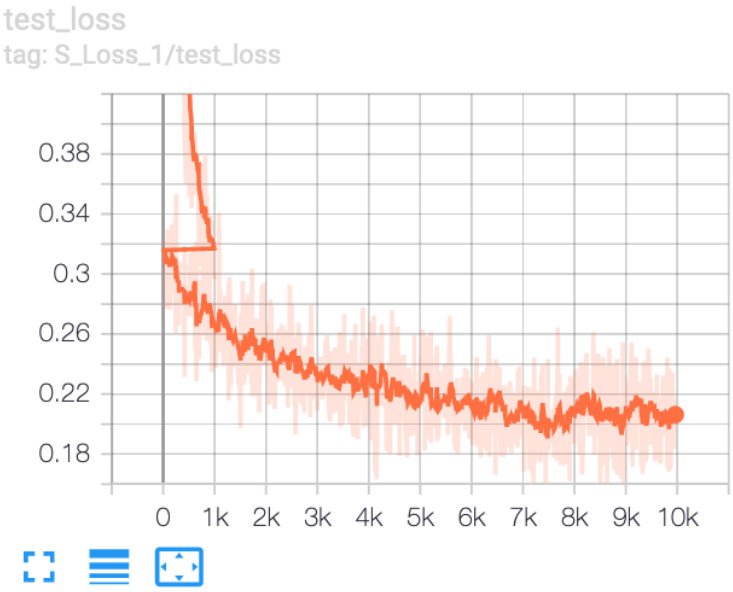
Figure 31: Results for batch size=1000 and learning rate=0.01


```
test loss is 0.33251405 step= 9996
test loss is 0.335923 step= 9997
test loss is 0.30355278 step= 9998
test loss is 0.3191156 step= 9999
test accuracy = 0.917
train accuracy = 0.9103091
```

Figure 32: Results for batch size=1000 and learning rate=0.05

```
test loss is 0.27702028 step= 9995
test loss is 0.27449238 step= 9996
test loss is 0.2928721 step= 9997
test loss is 0.26626834 step= 9998
test loss is 0.27912244 step= 9999
test accuracy = 0.9242
train accuracy = 0.92498183
```

Figure 33: Results for batch size=1000 and learning rate=0.1




```
test loss is 0.17873728 step= 9994
test loss is 0.21535185 step= 9995
test loss is 0.23105453 step= 9996
test loss is 0.21110499 step= 9997
test loss is 0.2304419 step= 9998
test loss is 0.18471295 step= 9999
test accuracy = 0.9403
train accuracy = 0.9495091
```

S_Loss

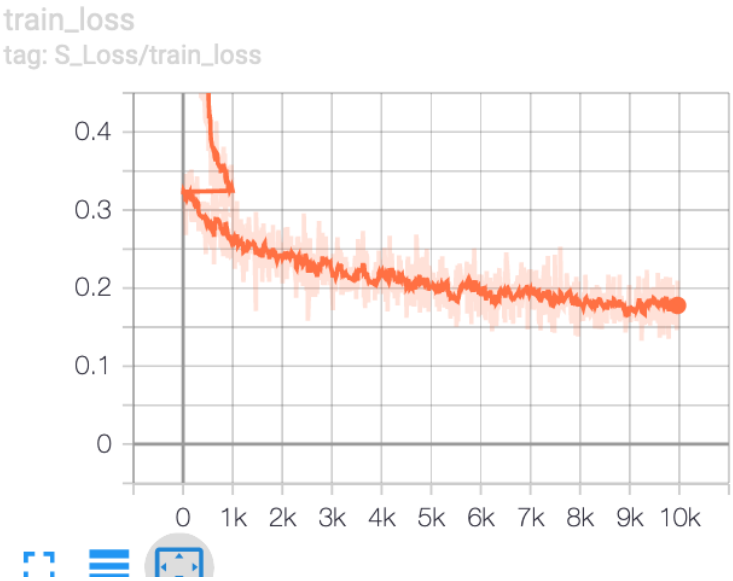


Figure 34: Results for batch size=1000 and learning rate=0.5

Problem 1 continued on next page. . .

```
test loss is 0.2027654 step= 9996
test loss is 0.23105831 step= 9997
test loss is 0.21892594 step= 9998
test loss is 0.20805961 step= 9999
test accuracy = 0.937
train accuracy = 0.9550909
```

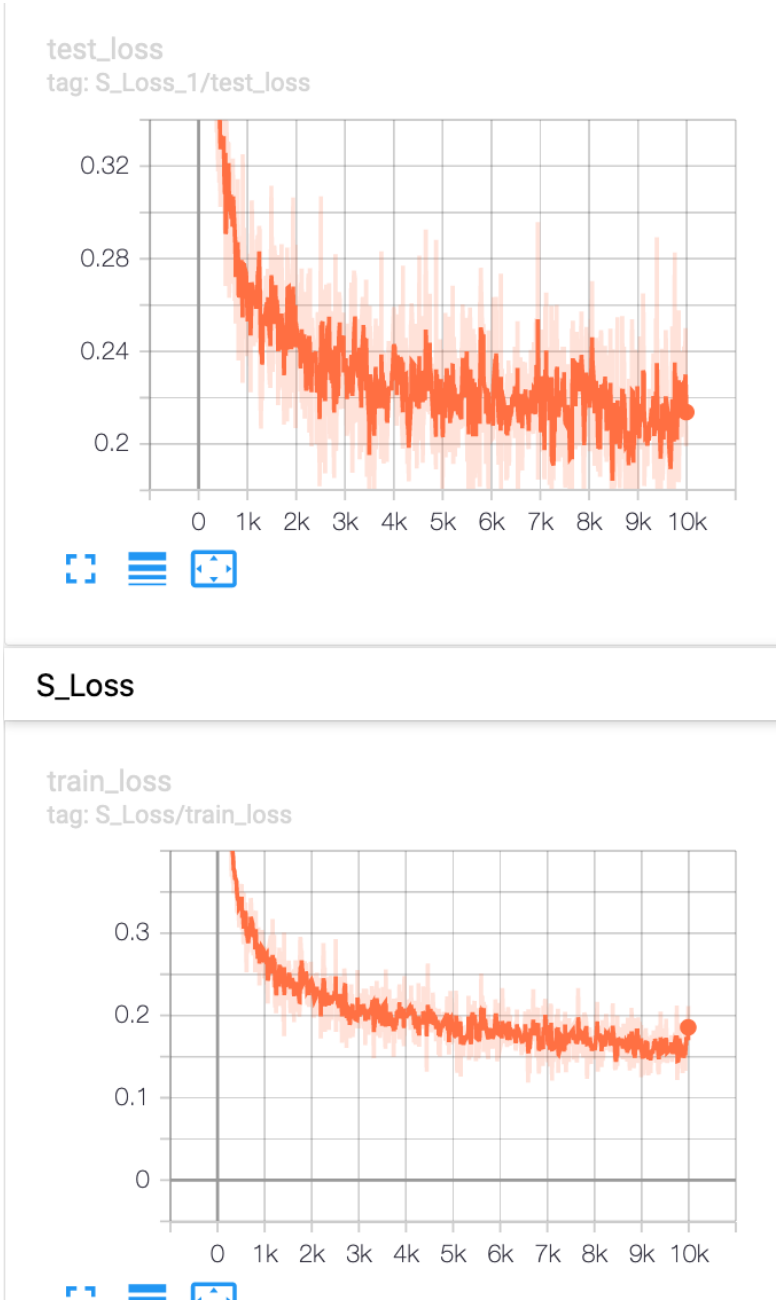


Figure 35: Results for batch size=1000 and learning rate=1

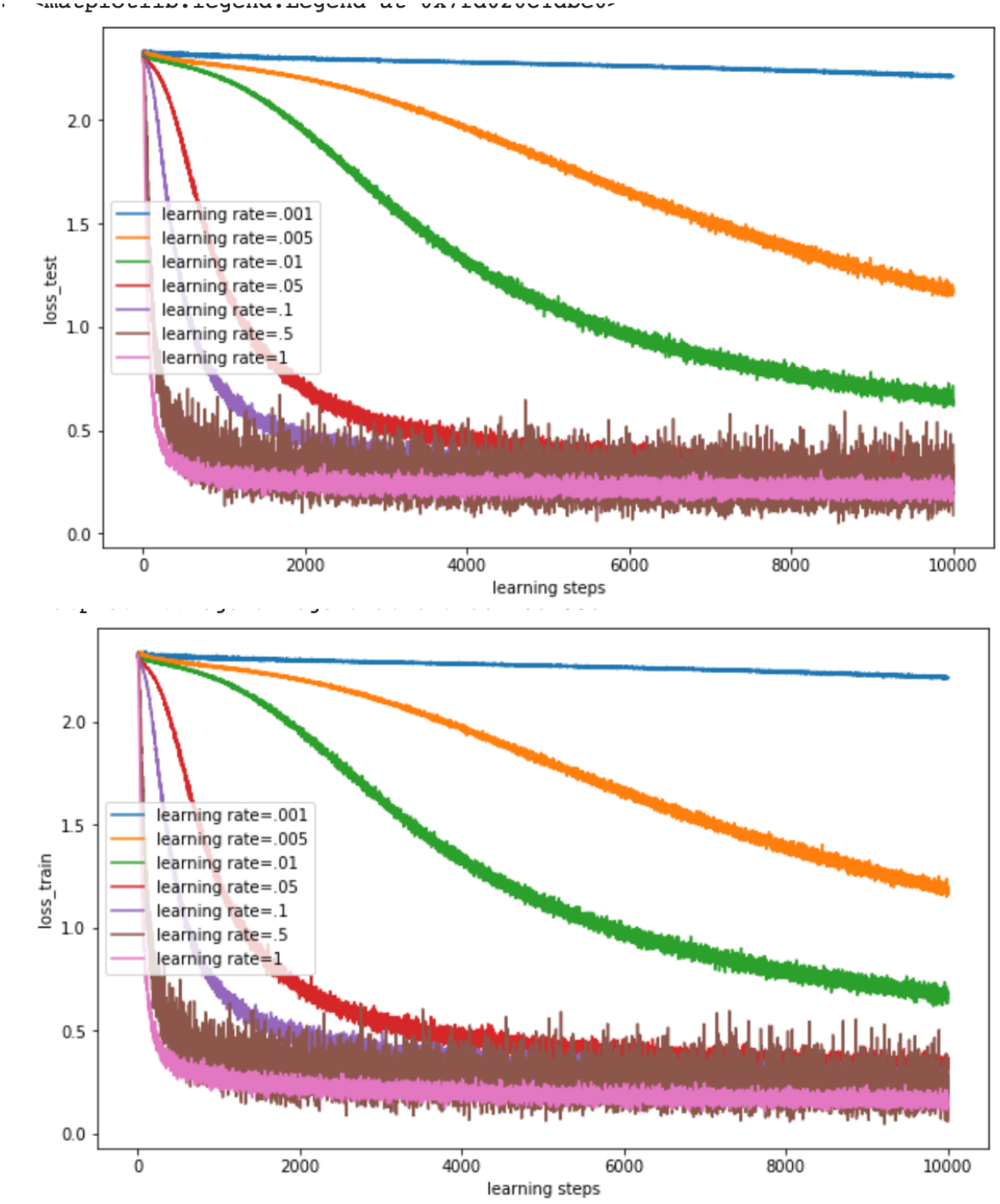


Figure 36: Results for batch size=1000 for different learning rates

Part 6:

Early-stopping is one of the regularization methods that prevent network from overfitting by restricting network complexity by means of adding a term to cost function that this term should greatly demonstrate the complexity of model which usually is *l2 norm* or *l1 norm* of weights.Early stopping considers validation error and train error and as soon as validation error started to increase as epochs goes by, algorithm stops the training procedure at that point because this is the point where addition of *variance* and *bias* of the model is minimized so generalization error reached it's optimum value and empirical error also is acceptable.Another justification for regularization is to prevent network from just memorizing data and not to find the pattern or structure of the data. From below figures it's clear that we train the network in 60000 *epoch* and 100000 *epoch* and check validation set error in each 1000 learning step to find out whether or not model is overfitted or not and whenever validation error started to increase we stop the learning procedure .In the code that is submitted,early-stopping part is commented.What was challenging here is to overfit the neural networks which can be achieved by many learning steps,but for better illustration i plotted results in different manners so as to better understanding.We can see from below that approximately **20000** learning steps is good choice and we should stop the algorithm in this step.Note that val-loss refers to validation loss.

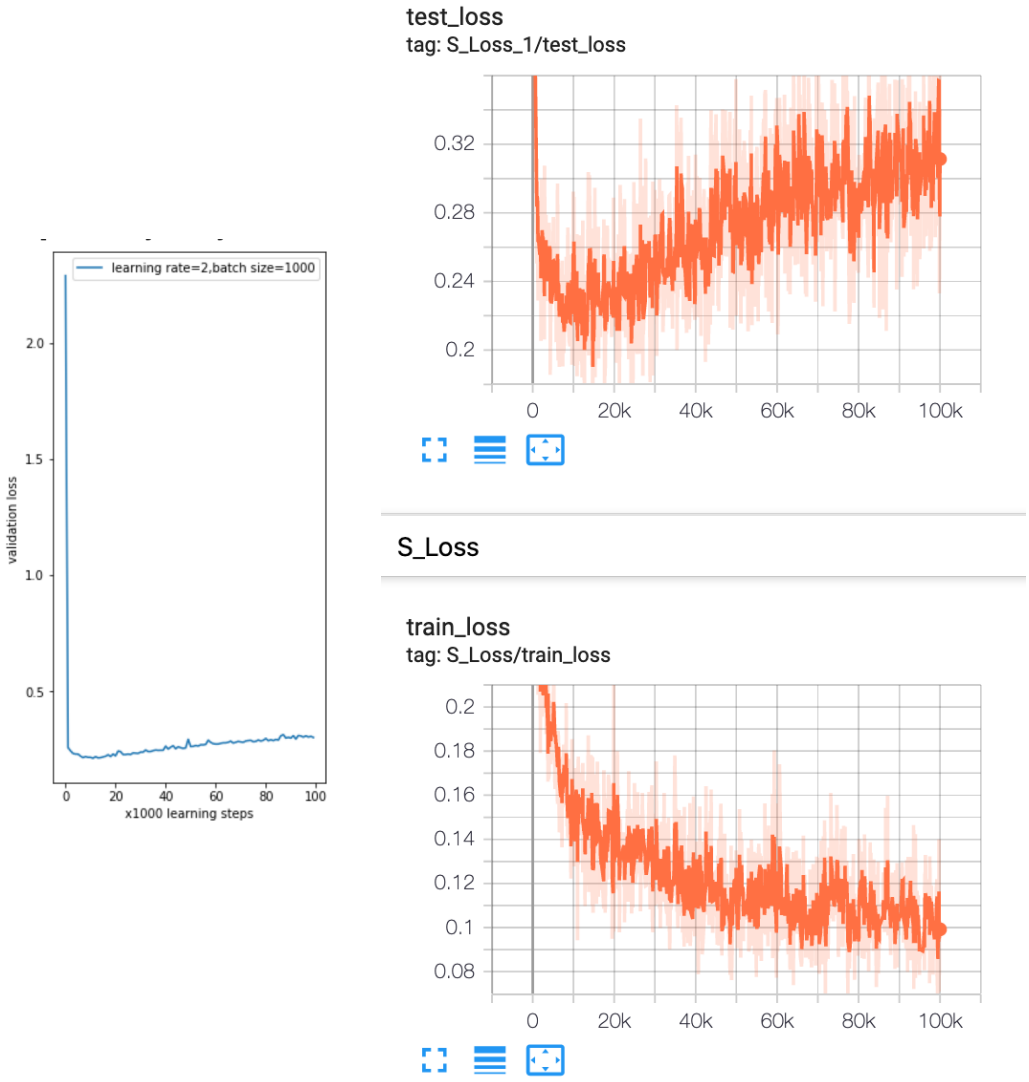


Figure 37: Results for batch size=1000 and learning rate=2

```
validation loss is 0.3040483 step= 97000
validation loss is 0.3071608 step= 98000
validation loss is 0.3022921 step= 99000
test accuracy = 0.932
train accuracy = 0.9703818
```

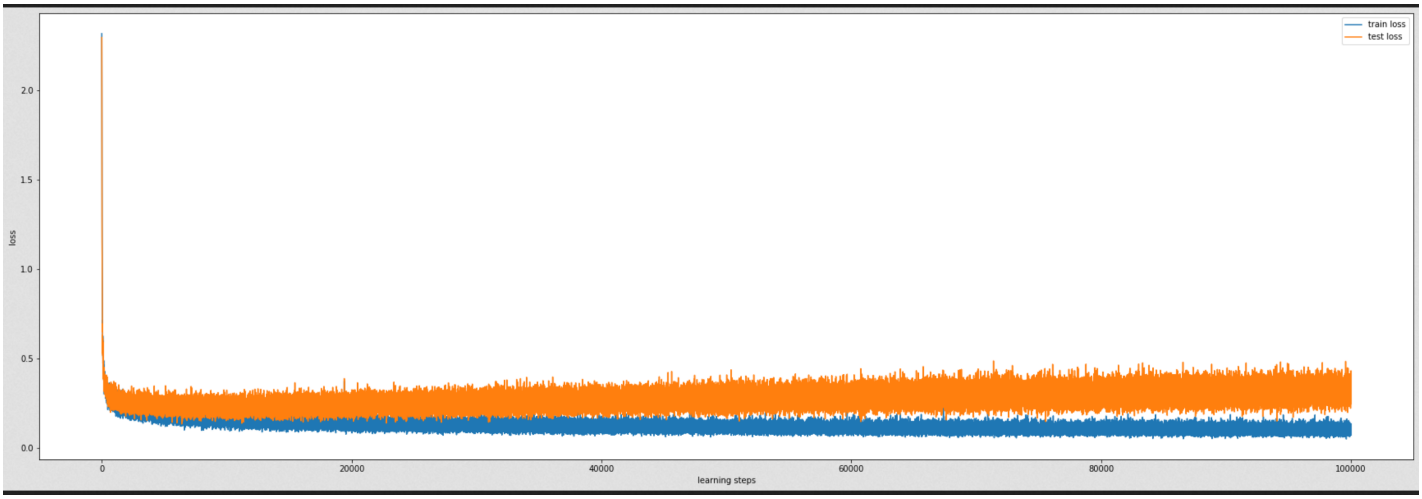


Figure 38: Results for batch size=1000 and learning rate=2

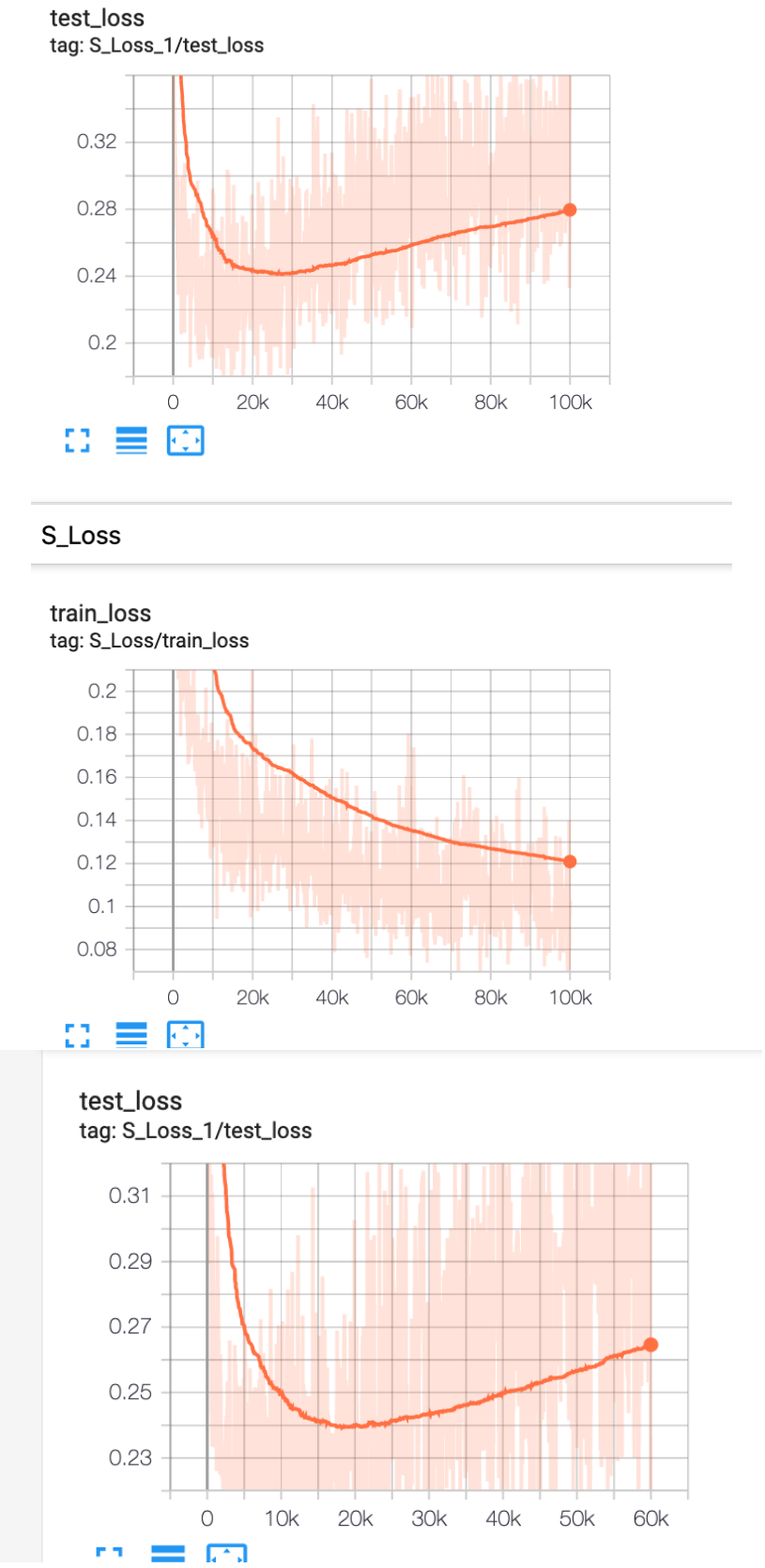
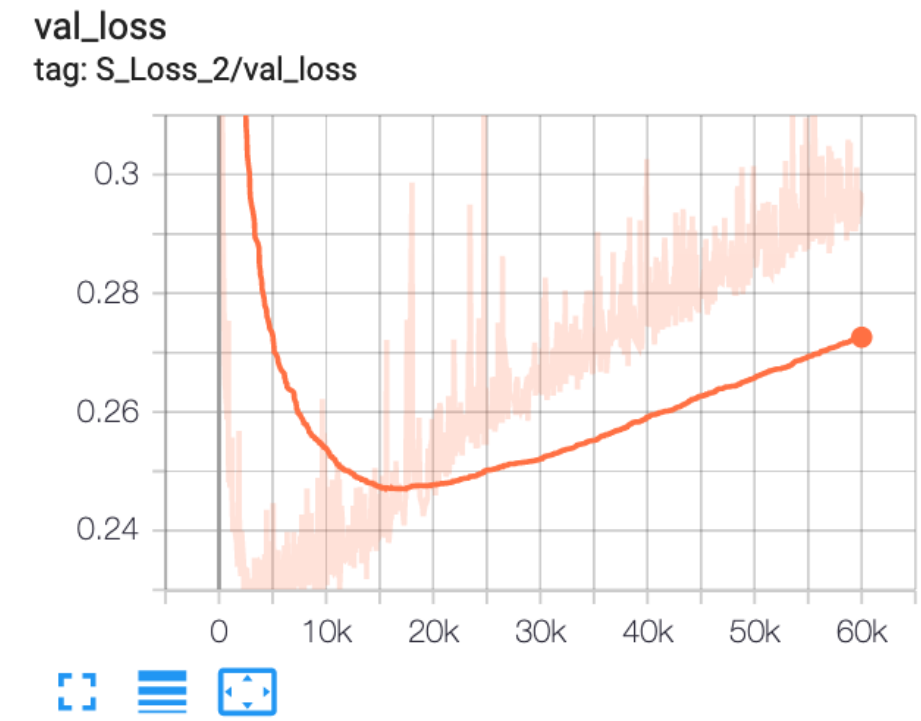


Figure 39: Results for batch size=1000 and learning rate=2



S_Loss

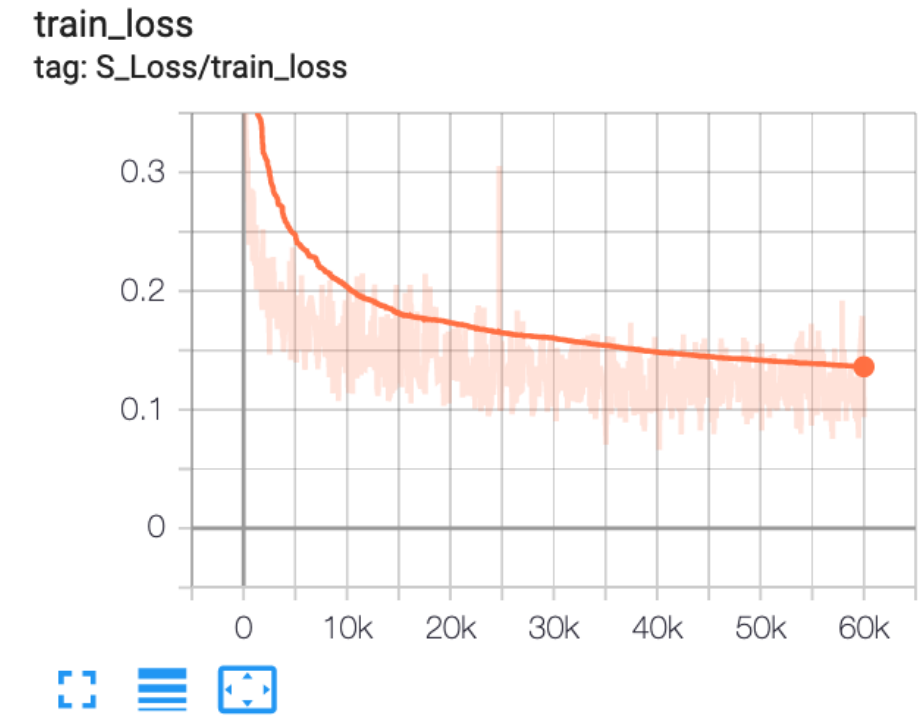


Figure 40: Results for batch size=1000 and learning rate=2