COMP 576 Deep Learning

Guangyuan Yu(gy12)

October 3, 2017

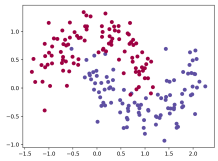
1 1 Backpropagation is a SNN

1.1 a Dataset

1.2 b.alpha-beta pruning

10 leaves are visited. The cut off parts are in the block. Leaves with "x" are not visited. Nodes with "x" are not calculated.

Figure 1: center



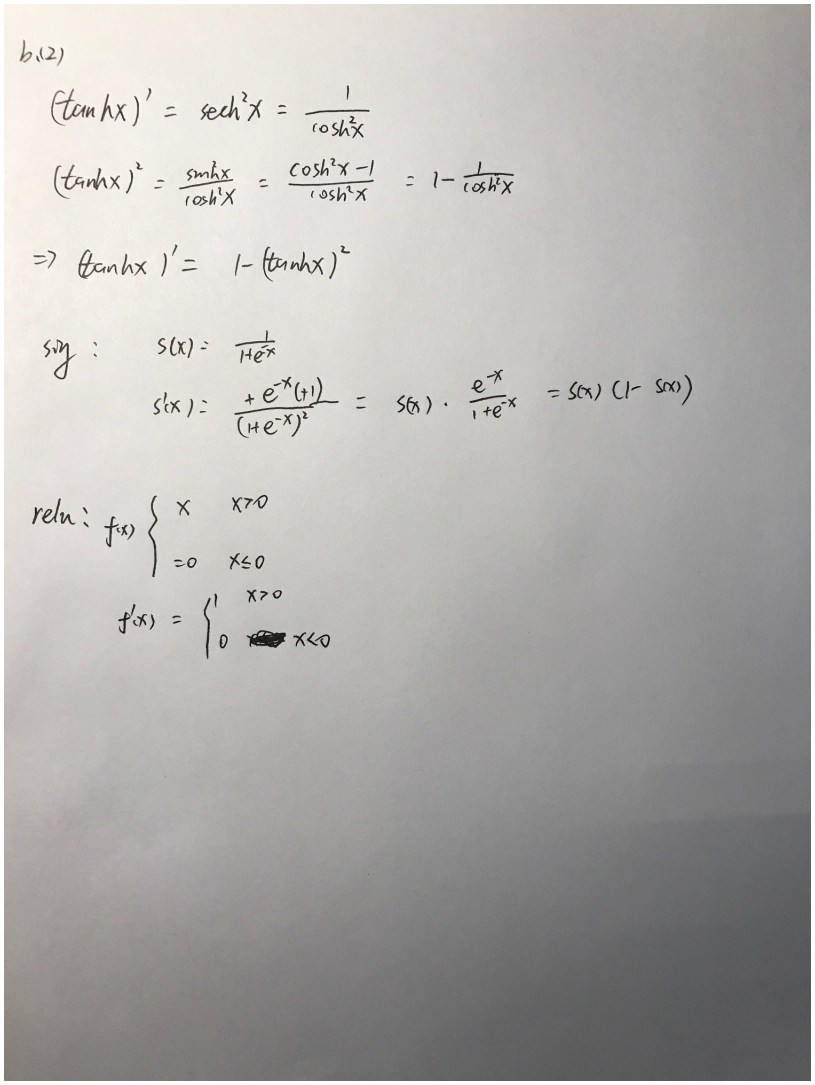
1.3 b.Activation Function

1.

```
if type == 'tanh':
            value = np.tanh(z)
        elif type == 'sigmoid':
            value = 1 / (1 + np.exp(-z))
                elif type == 'relu':
```

```
        value = z * (z > 0)
    else:
        raise Exception('Invalid activation function!')
```

Figure 2: center



3.

```
if type == 'tanh':
```

```
        value = np.tanh(z)
        diffvalue = 1 - (value * value)



    elif type == 'sigmoid':
        value = 1 / (1 + np.exp(-z))

        diffvalue = value * (1 - value)
    elif type == 'relu':

        diffvalue = (z > 0) * 1
```

1.4 c

1

```
self.z1 = np.dot(X, self.W1) + self.b1
        self.a1 = actFun(self.z1)
        self.z2 = np.dot(self.a1, self.W2) + self.b2

        exp_scores = np.exp(self.z2)
        self.probs = exp_scores / np.sum(exp_scores, axis=1, keepdims=T
```

2.

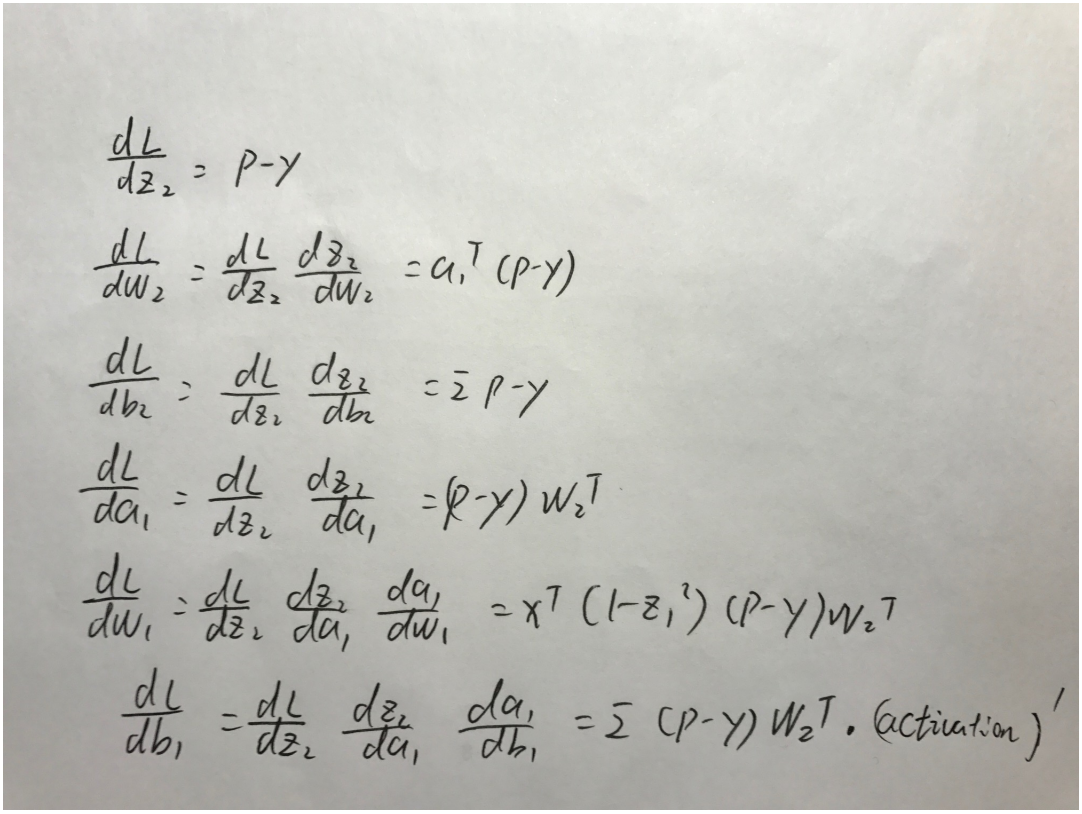
```
data_loss = np.sum(-np.log(self.probs[range(num_examples), y]))
        # data_loss =


        data_loss += self.reg_lambda / 2 * (np.sum(np.square(self.W1)) -
```

1.5 d.

1

Figure 3: center



2

```
num_examples = len(X)
        delta3 = self.probs
        delta3[range(num_examples), y] -= 1

        dW2 = (self.a1.T).dot(delta3)
        db2 = np.sum(delta3, axis=0, keepdims=True)
        delta2 = delta3.dot(self.W2.T) * self.diff_actFun(self.z1, self
        dW1 = np.dot(X.T, delta2)
        db1 = np.sum(delta2, axis=0)
```

1.6 e.

1

Figure 4: center

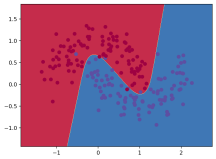


Figure 5: center

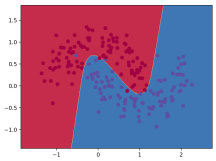
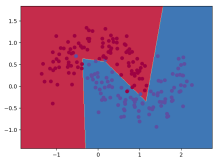
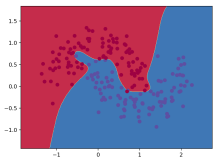


Figure 6: center



The decision boundary for sig and tanh are similar and smooth, while for relu, the boundary is very sharp. This is because the relu function is not smooth. Some data point are not labeled correctly. 2

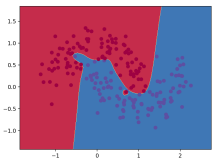
Figure 7: center



Now I have 10 unit in hidden layers, The new plot is more curved and more data point are right labeled. It means increasing the hidden unit, the ability of the neurial network is increasing.

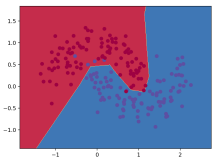
1.7 f.

Figure 8: center



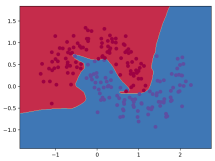
It has 10 layers, and hidden layer have 6 units,relu.

Figure 9: center



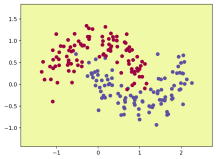
The decision boundary is slightly different from 10 layerthan, this one is more like a straight line. Since it is straight, there are more data point are not labeled correctly.

Figure 10: center



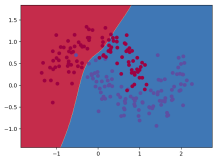
Then I used a 20 layer with 10 unit. now the relu could give a smooth boundary too.

Figure 11: center



When it is too deep, the loss would not change, it is alway around 0.7. It is because the relu could not pass the gradient back. it is dead relu.

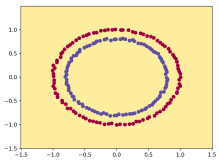
Figure 12: center



The same structure for tanh function, the problem will be overfitting. the loss goes to 0.06 then ,finally go to 0.5. But, tanh can back propagate. Since the gradient of tanh is not flat, it alwasy works.

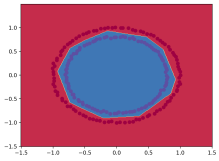
1.8 On another dataset

Figure 13: center



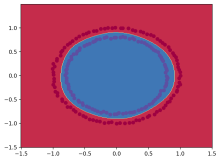
On this dataset, tanh also dead on deep network. 30 layers. 30 unit. Relu also dead here.

Figure 14: center



This is a relu with 4 layer, 4 unit. The boundary is not smooth.

Figure 15: center



This is tanh, the boundry is smooth.

2 Training a simple DCN on MNIST

2.1 A

SEE THE CODE FILE (5) admin relu step 5400, training accuracy 0.98 step 5500, training accuracy 1 test accuracy 0.988 The training takes 53.495326 second to finish (6)

Figure 16: center

2.2 B

Figure 17: center

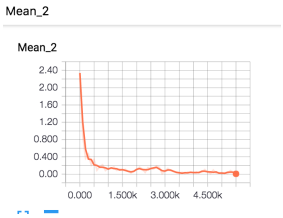


Figure 18: center

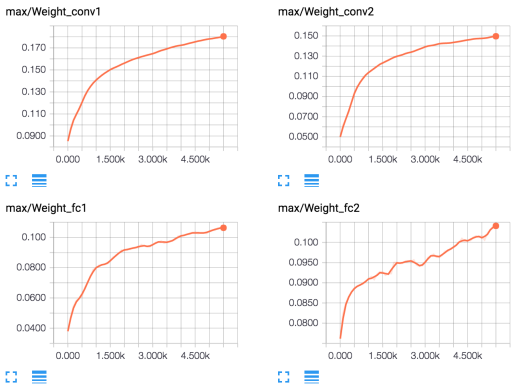


Figure 19: center

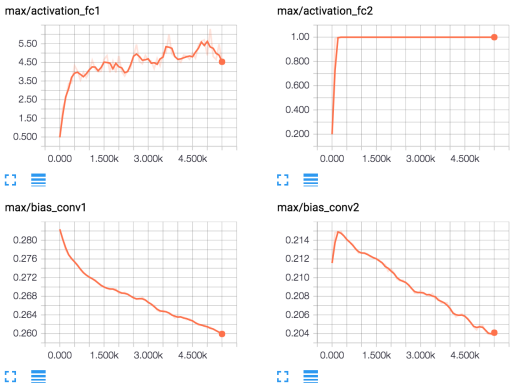


Figure 20: center

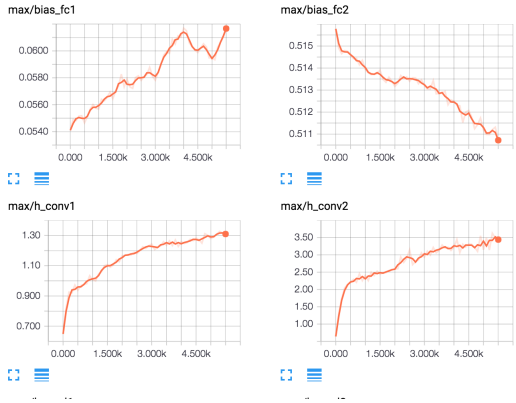


Figure 21: center

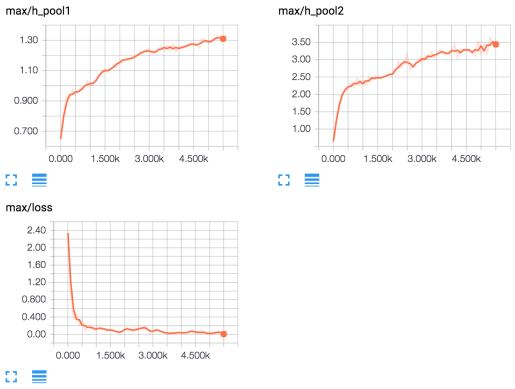


Figure 22: center

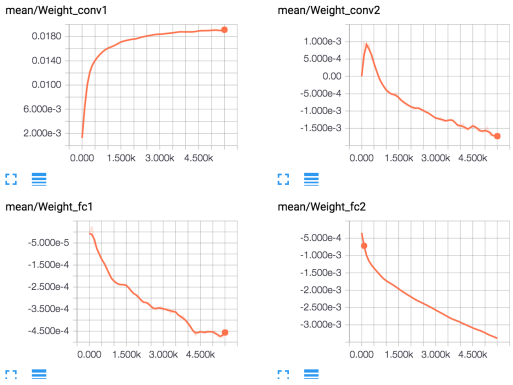


Figure 23: center

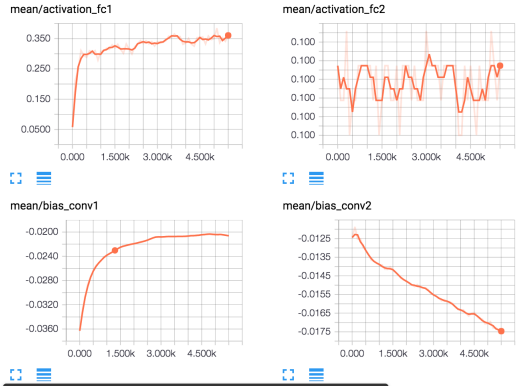


Figure 24: center

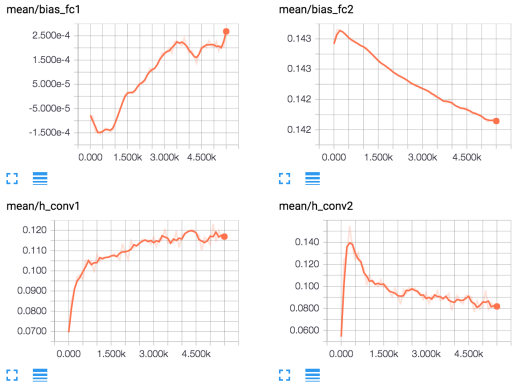


Figure 25: center

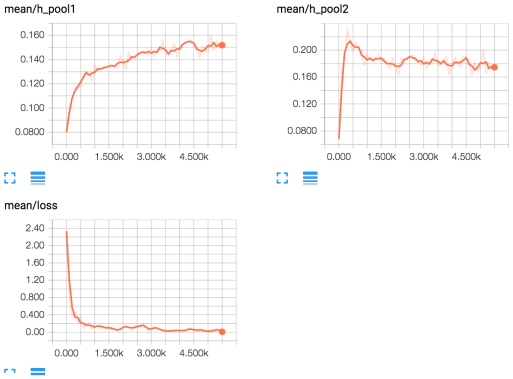


Figure 26: center

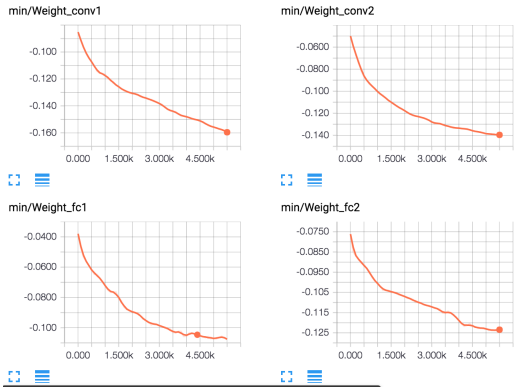


Figure 27: center

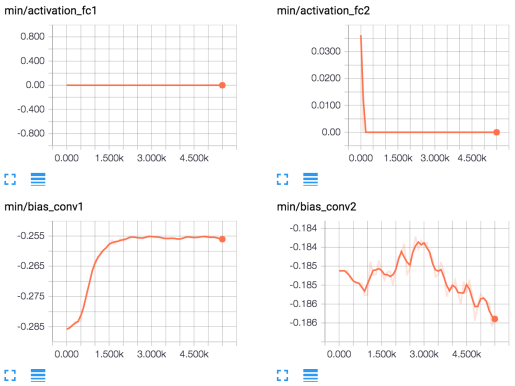


Figure 28: center

Figure 29: center

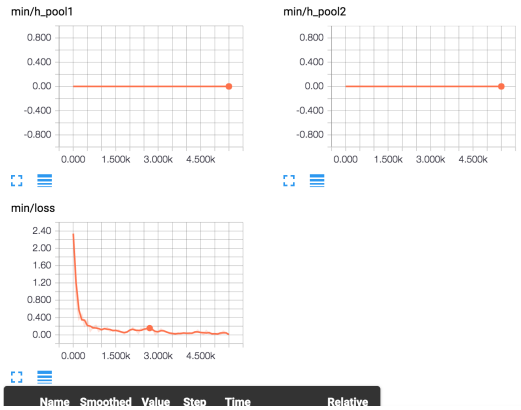


Figure 30: center

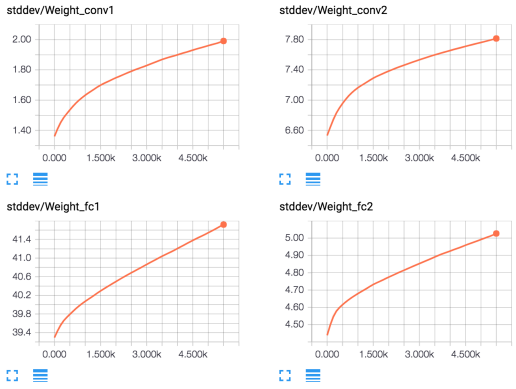


Figure 31: center

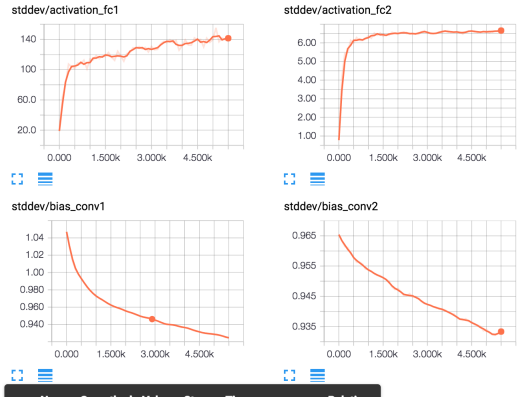


Figure 32: center

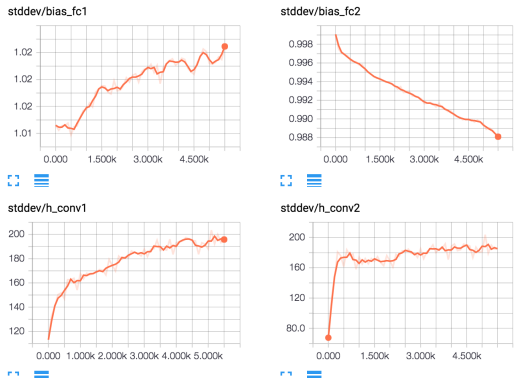
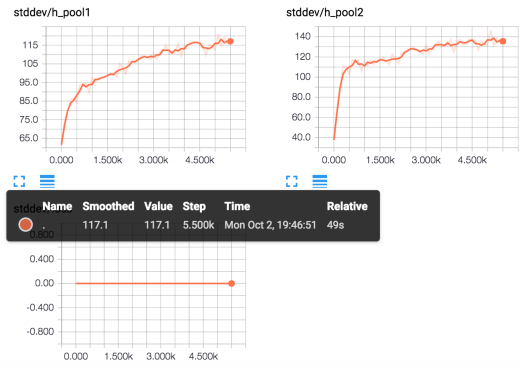


Figure 33: center



I put the validation accuracy monitor in a son file called "valid" And put parts of the plots here.

Figure 34: center


```
step 3300, training accuracy 1
test accuracy 0.9853
validation accuracy 0.984
step 3400, training accuracy 0.96
step 3500, training accuracy 1
step 3600, training accuracy 1
step 3700, training accuracy 1
step 3800, training accuracy 0.98
step 3900, training accuracy 1
step 4000, training accuracy 1
step 4100, training accuracy 0.98
step 4200, training accuracy 0.98
step 4300, training accuracy 1
step 4400, training accuracy 1
test accuracy 0.988
validation accuracy 0.9878
step 4500, training accuracy 1
step 4600, training accuracy 1
step 4700, training accuracy 0.98
step 4800, training accuracy 0.98
step 4900, training accuracy 1
```

```
step 5000, training accuracy 1
step 5100, training accuracy 0.98
step 5200, training accuracy 1
step 5300, training accuracy 1
step 5400, training accuracy 0.98
step 5500, training accuracy 1
test accuracy 0.9899
validation accuracy 0.989
test accuracy 0.9899
The training takes 54.957134 second to finish
```

2.3 C

I use sigmoid function, RMSPropOptimizer. After 5500 training step 5400, training accuracy 0.98 step 5500, training accuracy 0.98 test accuracy 0.9454 validation accuracy 0.95 The training takes 106.673226 second to finish. The accuracy goes up slowly at the begining.

FIGURES:

Figure 35: center

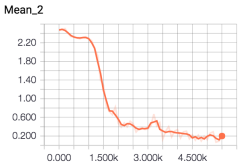


Figure 36: center

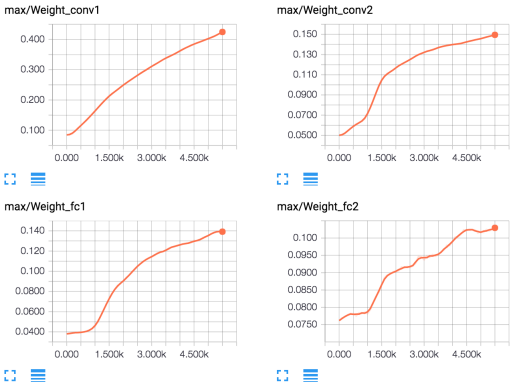


Figure 37: center

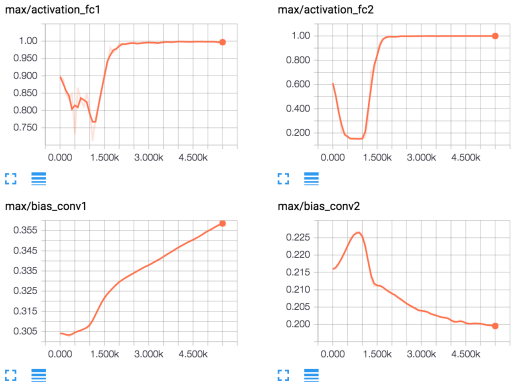


Figure 38: center

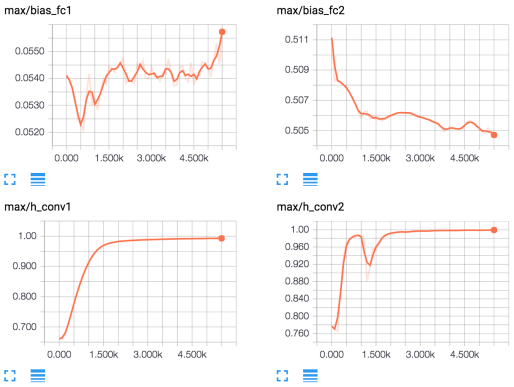


Figure 39: center

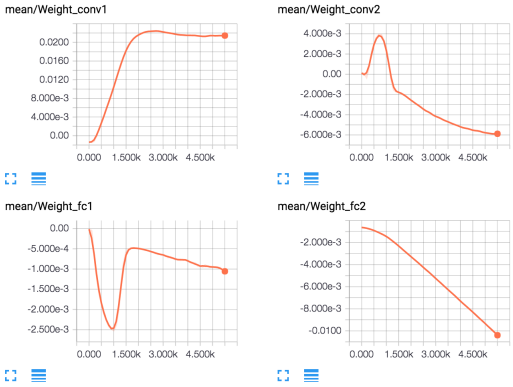


Figure 40: center

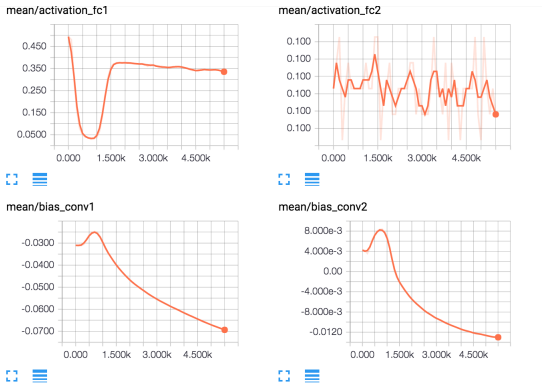


Figure 41: center

Figure 42: center

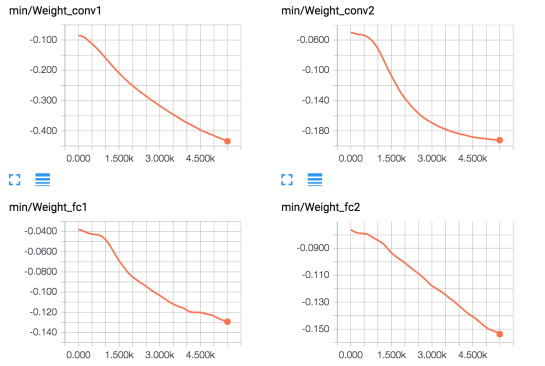


Figure 43: center

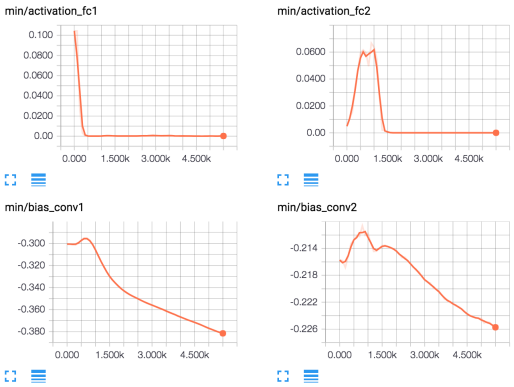


Figure 44: center

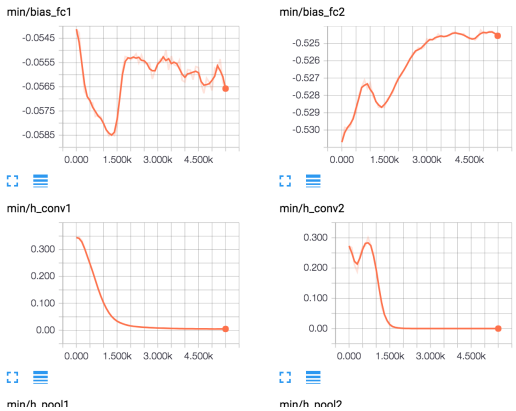


Figure 45: center

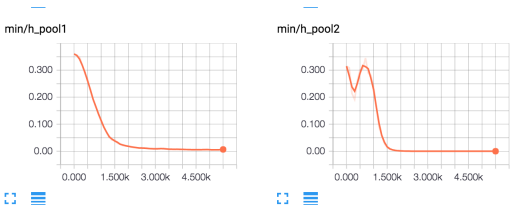


Figure 46: center

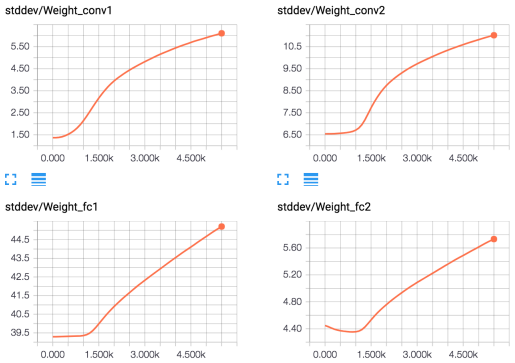


Figure 47: center

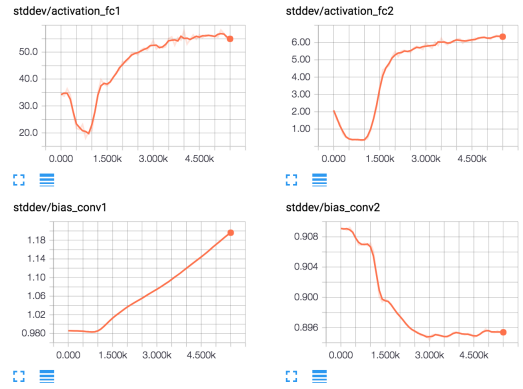


Figure 48: center

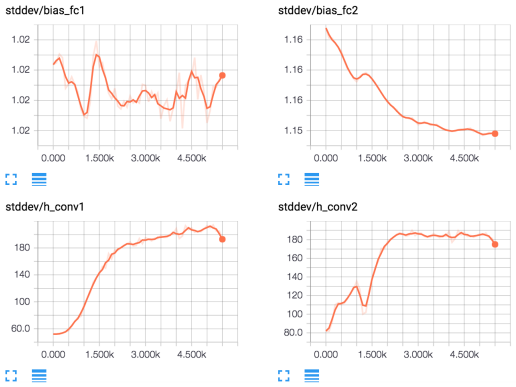
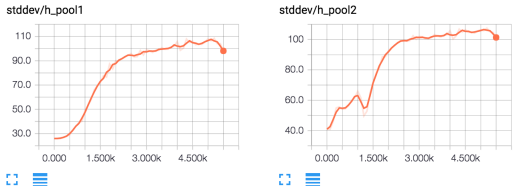


Figure 49: center



I used relu function, RMSPropOptimizer. After 5500 training, the validation accuracy is 0.98, which is better than sigmond funciton. Usually relu has better performance than sigmond. The tanh function has final validation accuracy of 0.985. It also has good performance. Since the amount of pictures is large, I didn't put them here.