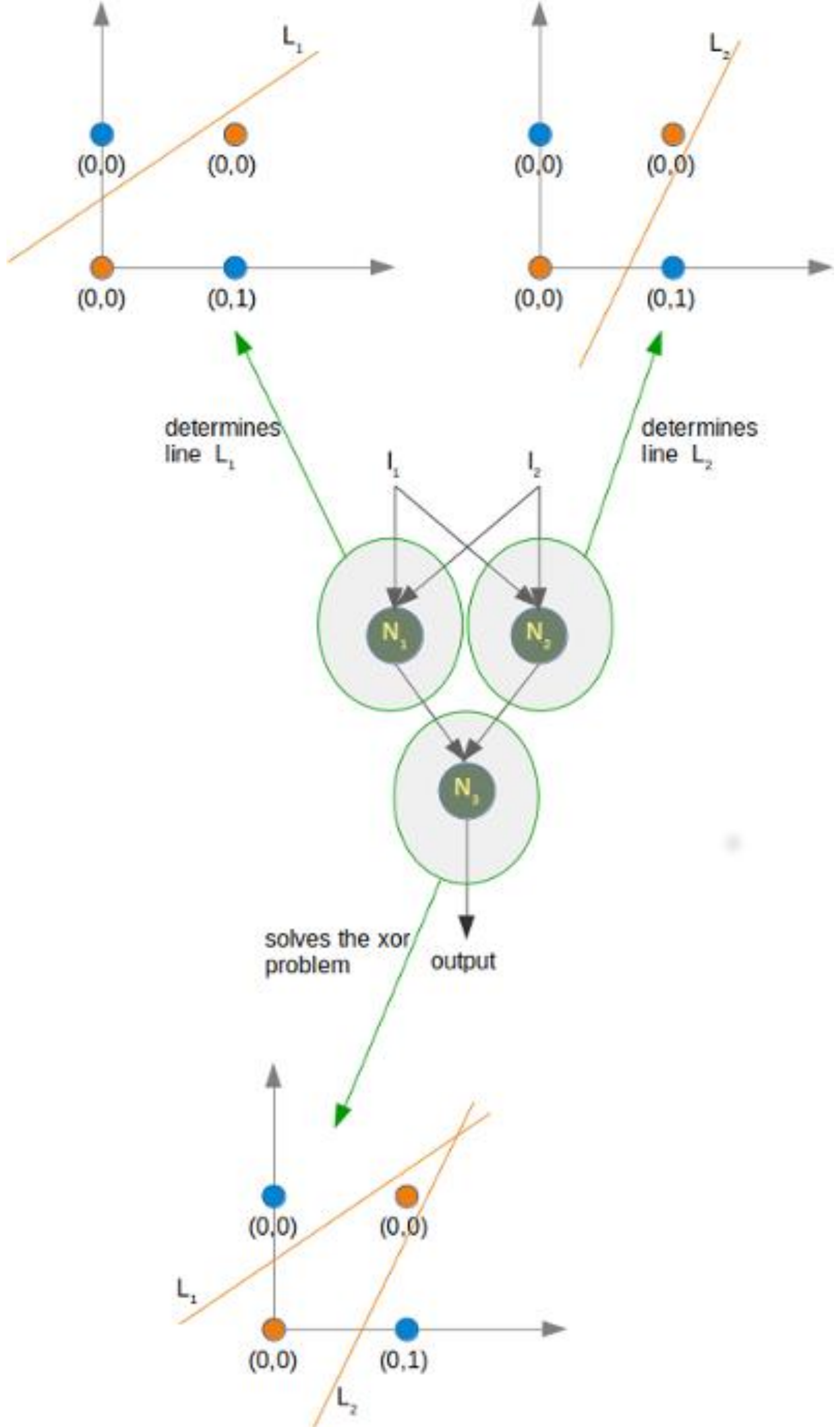
Multi-layer Perceptron

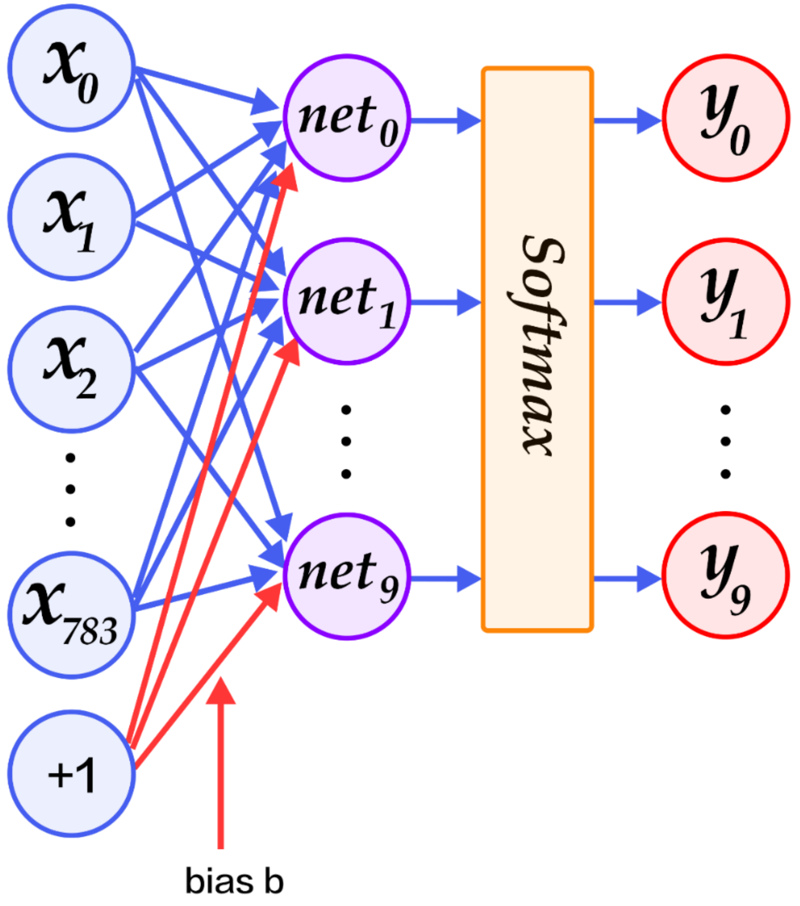
By Comdet Phaudphut, fb.com/comdet, comdet.p@gmail.com

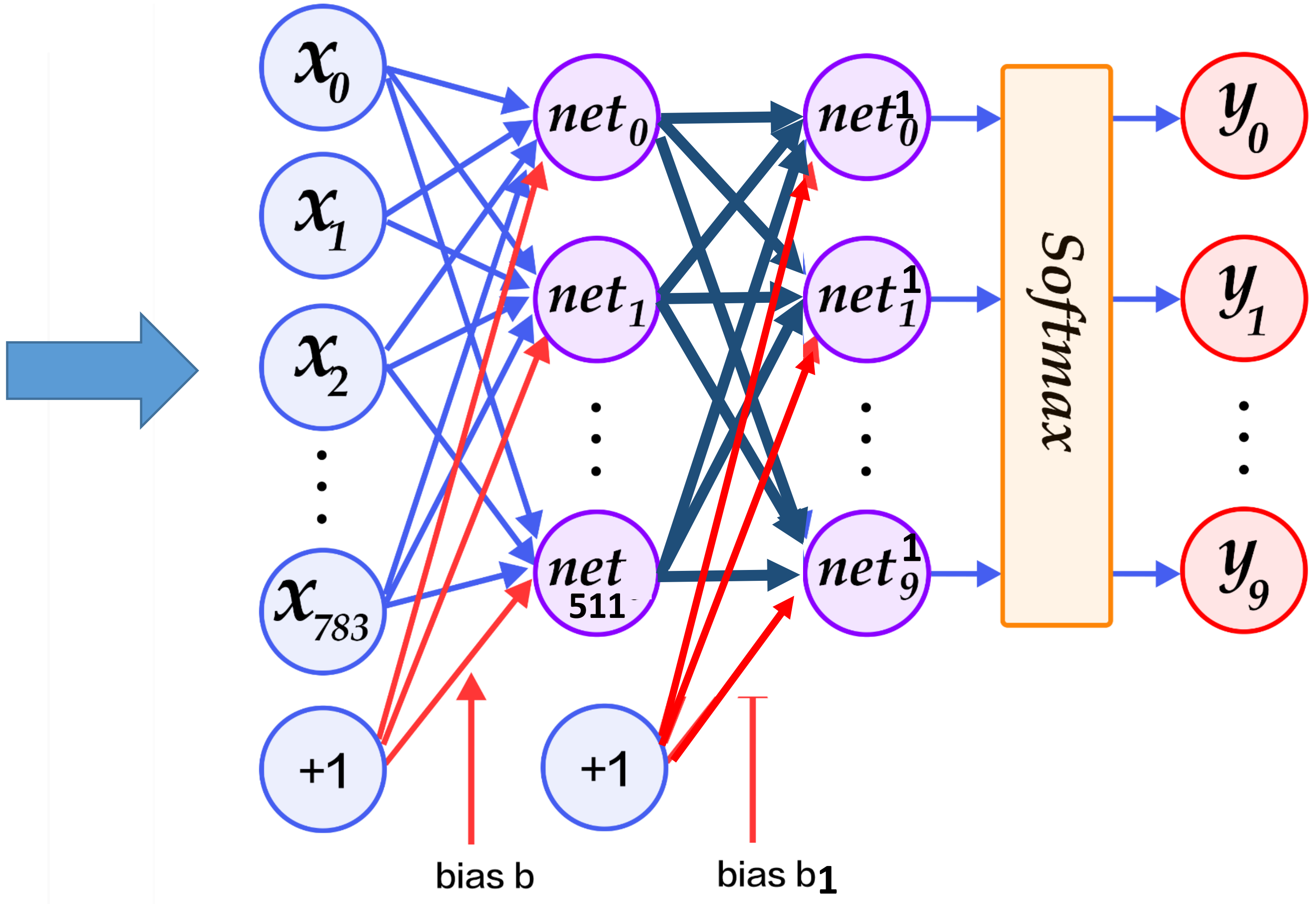
Review Prev Lab

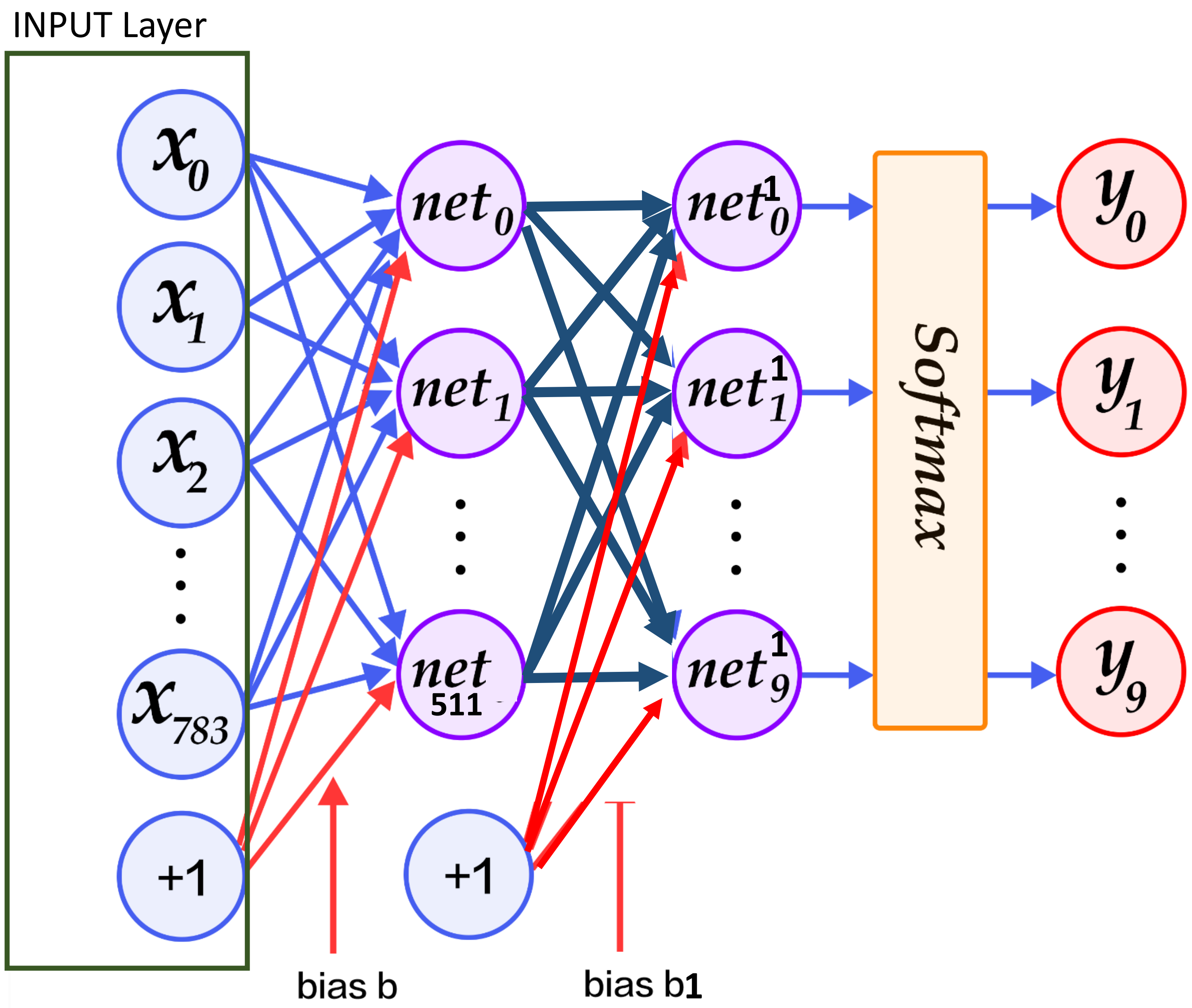
- We add more perceptron to classify more output
- We use softmax function to see log normalize of output
- We use Tensorflow to do our computation job.
- We create optimizer to minimize loss
- We train step and accuracy resulted around 92-93%
- But remember our problem ?
  - XOR!

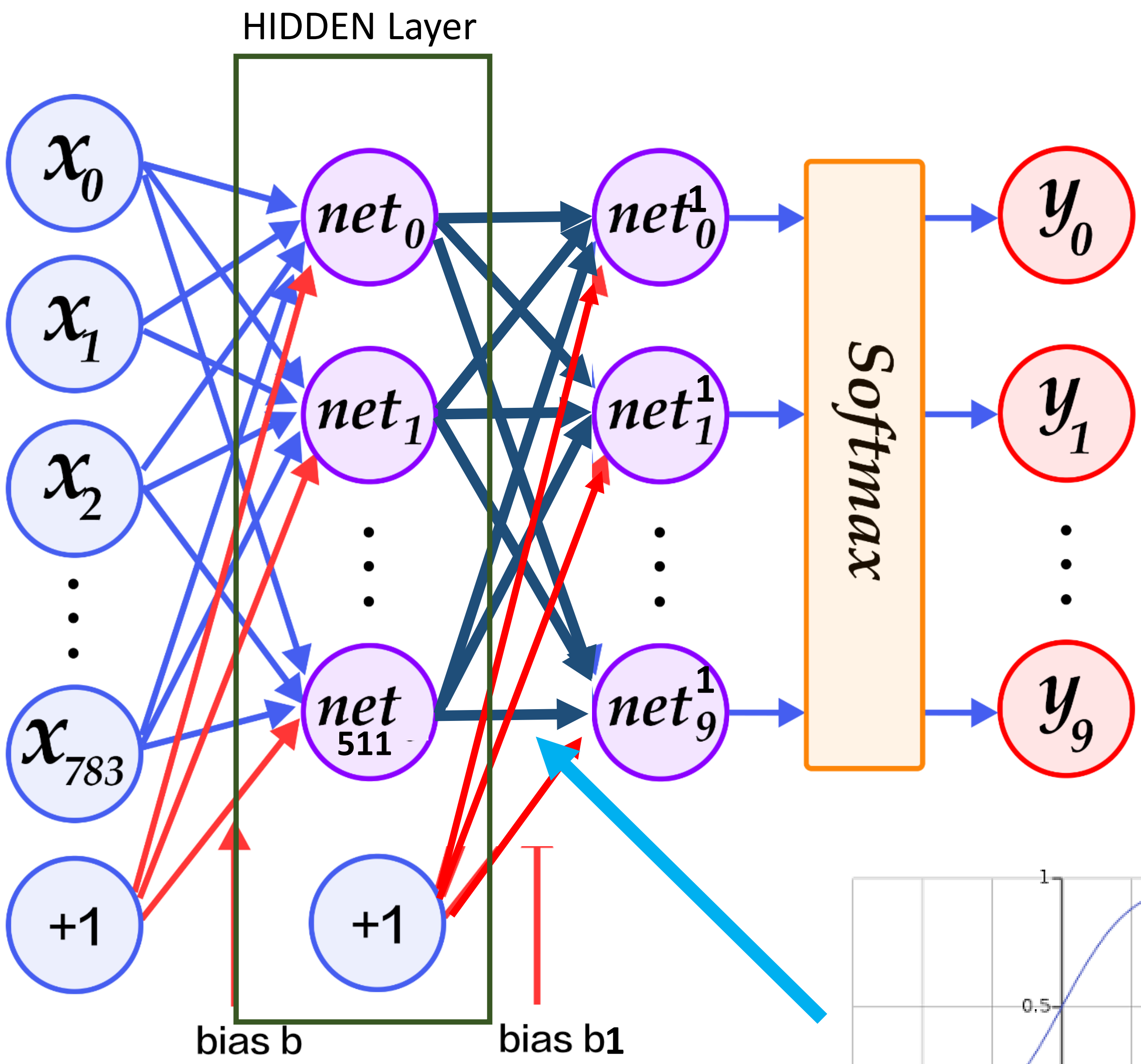
Simple answer for XOR problem, Add more layer

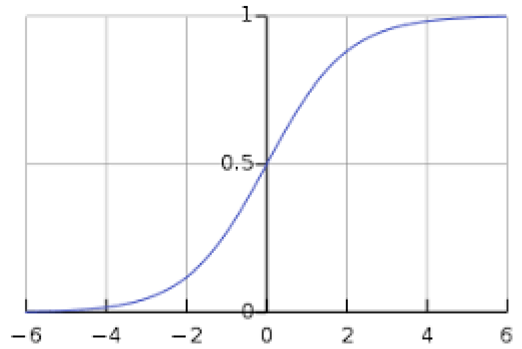



x_0
x_1
x_2
x_{783}
+1
net_0
net_1
net_9
Softmax
y_0
y_1
y_9
bias b

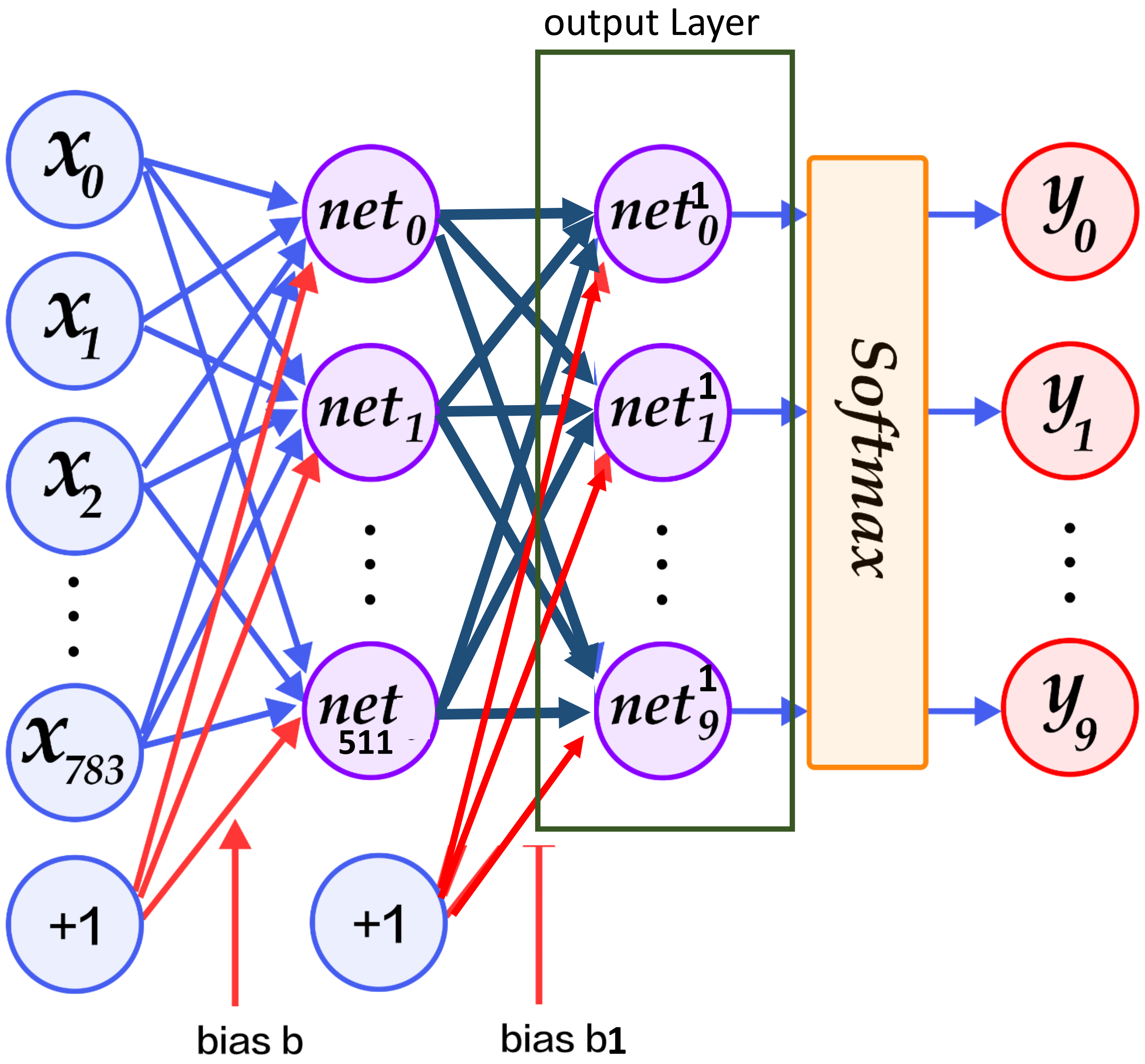

x_0
x_1
x_2
x_{783}
+1
net_0
net_1
net_{511}
+1
net^1_0
net^1_1
net^1_9
Softmax
y_0
y_1
y_9
bias b
bias b1


INPUT Layer
x_0
x_1
x_2
x_{783}
+1
net_0
net_1
net_{511}
+1
net^1_0
net^1_1
net^1_9
Softmax
y_0
y_1
y_9
bias b
bias b1



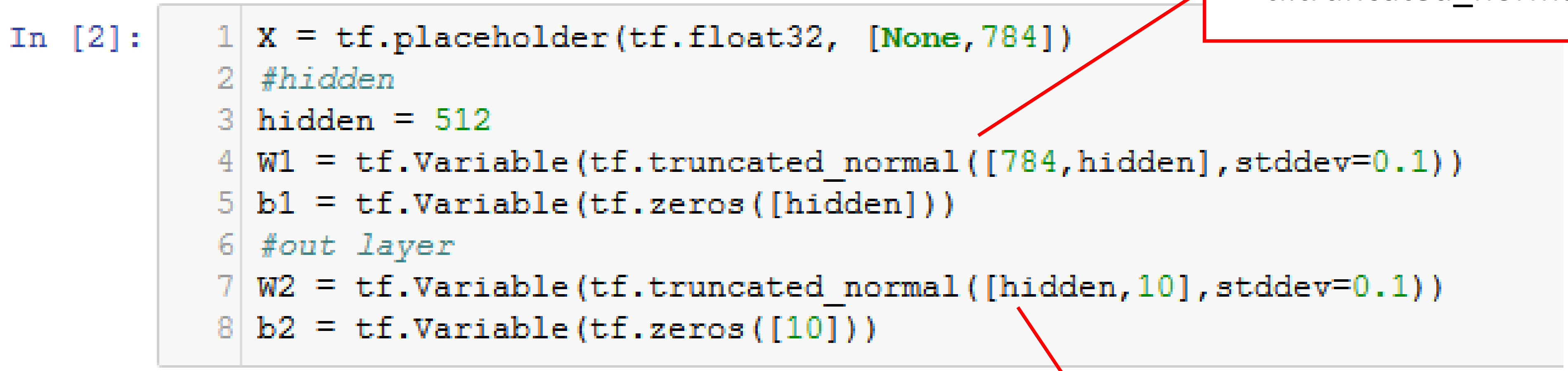


Sigmoid function


output Layer
x_0
x_1
x_2
x_{783}
+1
net_0
net_1
net_{511}
+1
net^1_0
net^1_1
net^1_9
Softmax
y_0
y_1
y_9
bias b
bias b1

What happen?

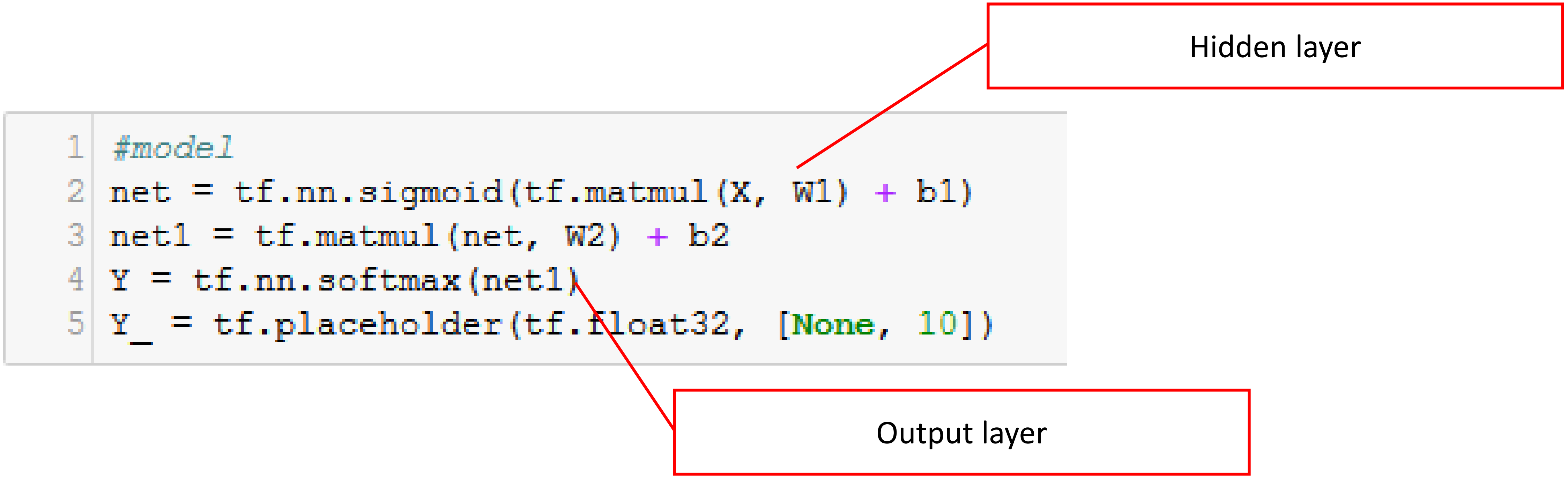
- We know how can we solve the XOR problem if we add more layer that make the next layer more classifiable
- We add more layer and call middle layer we called “hidden” layer
- We select 512 hidden layer but I don’t know why we choose this?
- We add sigmoid activation function between hidden layer and output to make output more stable between 0-1



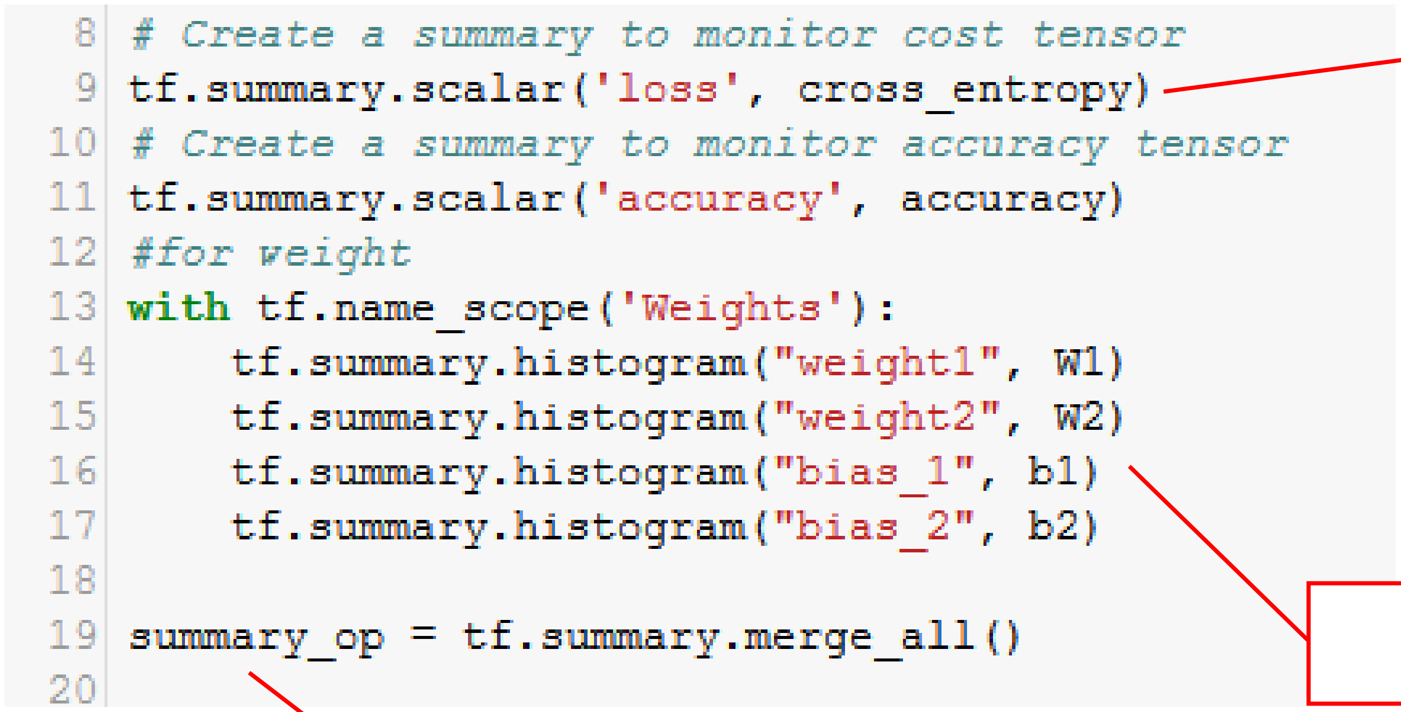
```
In [2]:
X = tf.placeholder(tf.float32, [None,784])
#hidden
hidden = 512
W1 = tf.Variable(tf.truncated_normal([784,hidden],stddev=0.1))
b1 = tf.Variable(tf.zeros([hidden]))
#out layer
W2 = tf.Variable(tf.truncated_normal([hidden,10],stddev=0.1))
b2 = tf.Variable(tf.zeros([10]))
```

tf.truncated_normal คือสุ่มที่ std = 0.1

เพิ่มอีก 1 layer



```python
#model
net = tf.nn.sigmoid(tf.matmul(X, W1) + b1)
net1 = tf.matmul(net, W2) + b2
Y = tf.nn.softmax(net1)
Y_ = tf.placeholder(tf.float32, [None, 10])
```



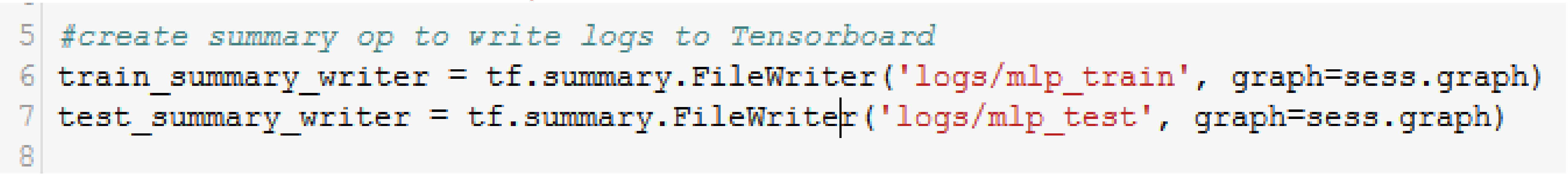
```python
# Create a summary to monitor cost tensor
tf.summary.scalar('loss', cross_entropy)
# Create a summary to monitor accuracy tensor
tf.summary.scalar('accuracy', accuracy)
#for weight
with tf.name_scope('Weights'):
    tf.summary.histogram("weight1", W1)
    tf.summary.histogram("weight2", W2)
    tf.summary.histogram("bias_1", b1)
    tf.summary.histogram("bias_2", b2)

summary_op = tf.summary.merge_all()

```

We want to watch this value every step

We want to watch this bias too

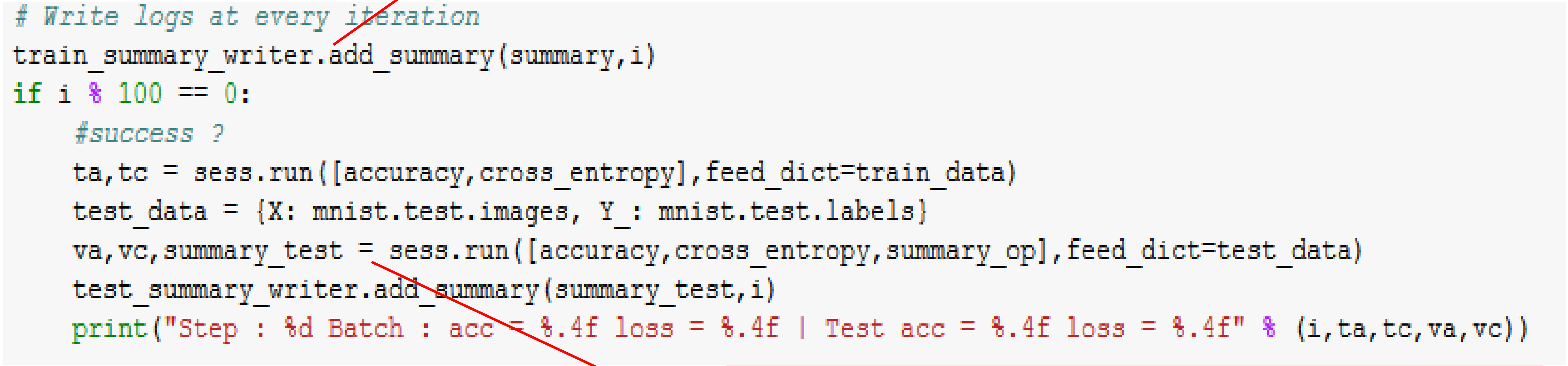
If we use summary_op we cen see loss every step

Tf.summary.FileWriter(‘ dir/file_prefix’
We create this to write that we watch to file



```
#create summary op to write logs to Tensorboard
train_summary_writer = tf.summary.FileWriter('logs/mlp_train', graph=sess.graph)
test_summary_writer = tf.summary.FileWriter('logs/mlp_test', graph=sess.graph)

```

Each step we write cost and accuracy to train_summary_writer



```
# Write logs at every iteration
train_summary_writer.add_summary(summary,i)
if i % 100 == 0:
    #success ?
    ta,tc = sess.run([accuracy,cross_entropy],feed_dict=train_data)
    test_data = {X: mnist.test.images, Y_: mnist.test.labels}
    va,vc,summary_test = sess.run([accuracy,cross_entropy,summary_op],feed_dict=test_data)
    test_summary_writer.add_summary(summary_test,i)
    print("Step : %d Batch : acc = %.4f loss = %.4f | Test acc = %.4f loss = %.4f" % (i,ta,tc,va,vc))
```

Every 100 steps we test accuracy and write to test_summary_writer

Open tensorboard

tensorboard –logdir="logs"

and see what happen!