# Tensorflow Introduction

Lifeng Fan

# Find toolkit? Tensorflow!

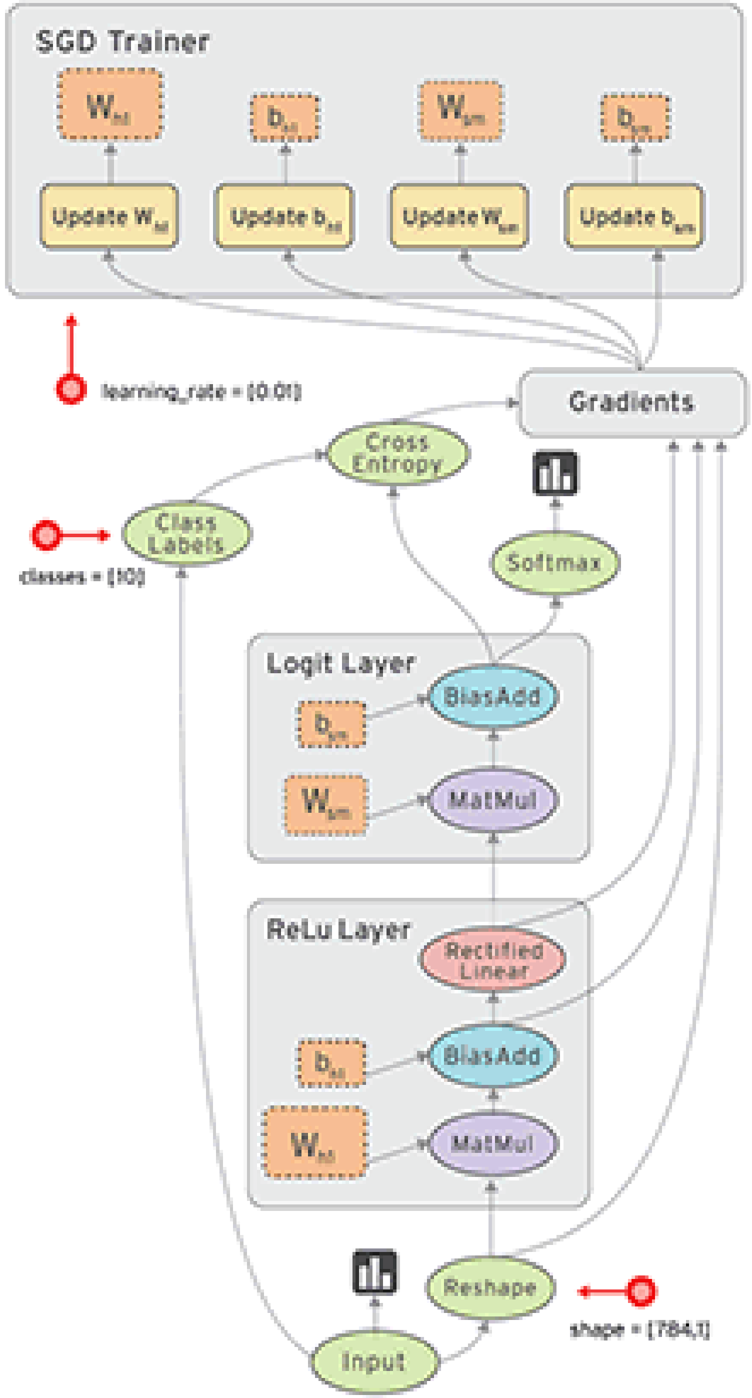
TensorFlow™

Q: What is Tensorflow?

A: A python-based toolkit for neural networks developed by Google.

Q: Advantages?

A: Open source, visualizable, and many metaframeworks for use!



- data-flow graph based
- node: mathematical operation
- line: data between nodes, represented by tensors.

# Tensor?

- Tensorflow: is a framework to define and run computations involving tensors.
- A **tensor** is a generalization of vectors and matrices to potentially higher dimensions. Internally, TensorFlow represents tensors as n-dimensional arrays of base datatypes.
- A **tf.Tensor** has the following properties:

- a data type (**float32**, **int32**, or **string**, for example)

- a shape

- Special Types: tf.Variable, tf.Constant, tf.Placeholder, tf.SparseTensor

# Tensor?

| Rank | Math entity |
|---|---|
| 0 | Scalar (magnitude only) |
| 1 | Vector (magnitude and direction) |
| 2 | Matrix (table of numbers) |
| 3 | 3-Tensor (cube of numbers) |
| n | n-Tensor (you get the idea) |

```
mammal = tf.Variable("Elephant", tf.string)
```

```
mystr = tf.Variable(["Hello"], tf.string)
```

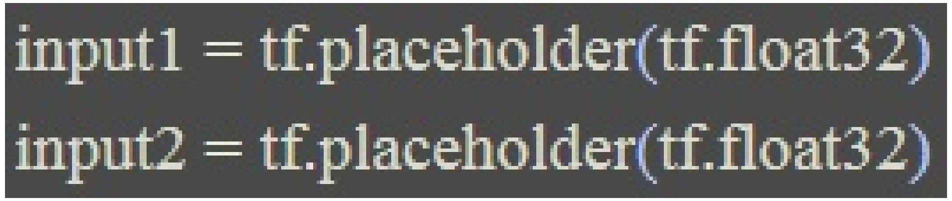
```
linear_squares =
tf.Variable([[4], [9], [16], [25]], tf.int32)
```

```
matrixB = tf.reshape(matrix, [3, -1])
```

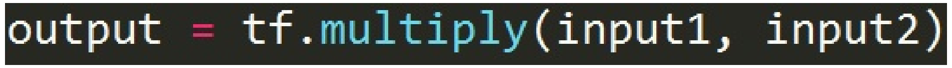
```
float_tensor = tf.cast(tf.constant([1, 2, 3]),
dtype=tf.float32)
```

# Tensor?

- If we want to send the "outside" data into our neural networks, we have to use placeholder as a container.
- 1) Define the data type of the placeholder

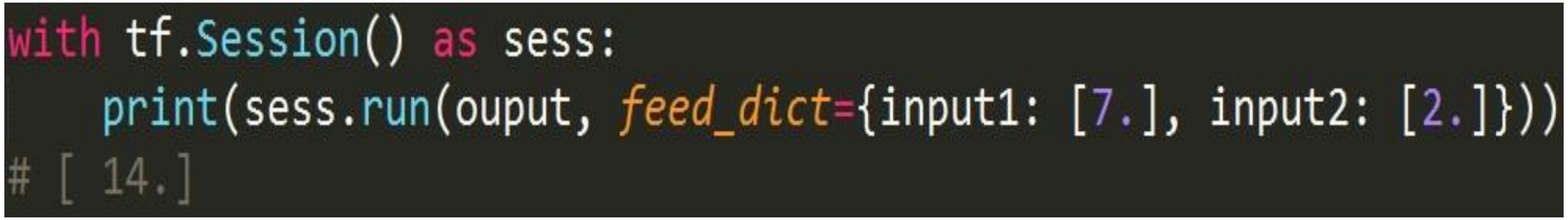


```
input1 = tf.placeholder(tf.float32)
input2 = tf.placeholder(tf.float32)
```



```
output = tf.multiply(input1, input2)
```

- 2) Use session to perform the "sending", and we employ feed_dict={} to designate the variables that will be sent in.



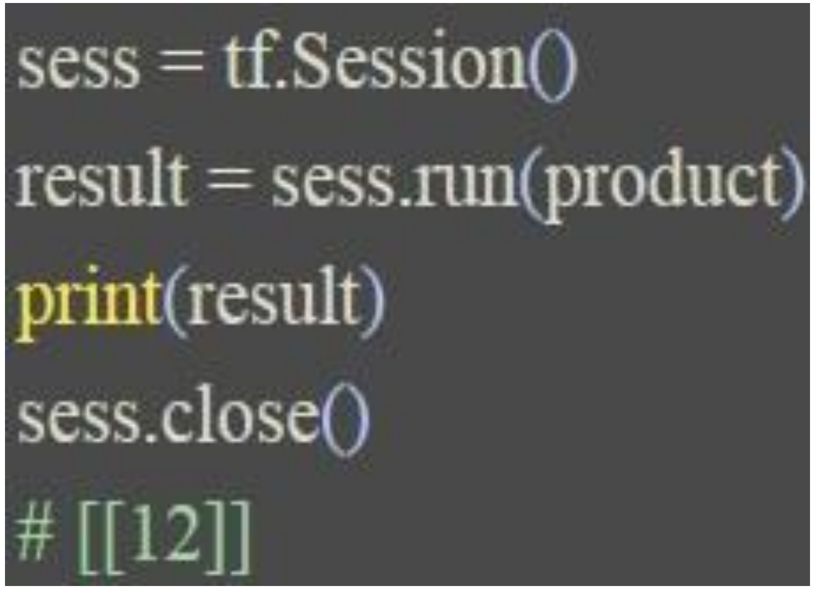
```
with tf.Session() as sess:
    print(sess.run(ouput, feed_dict={input1: [7.], input2: [2.]}))
# [ 14.]
```

# Executing a graph in a tf.Session

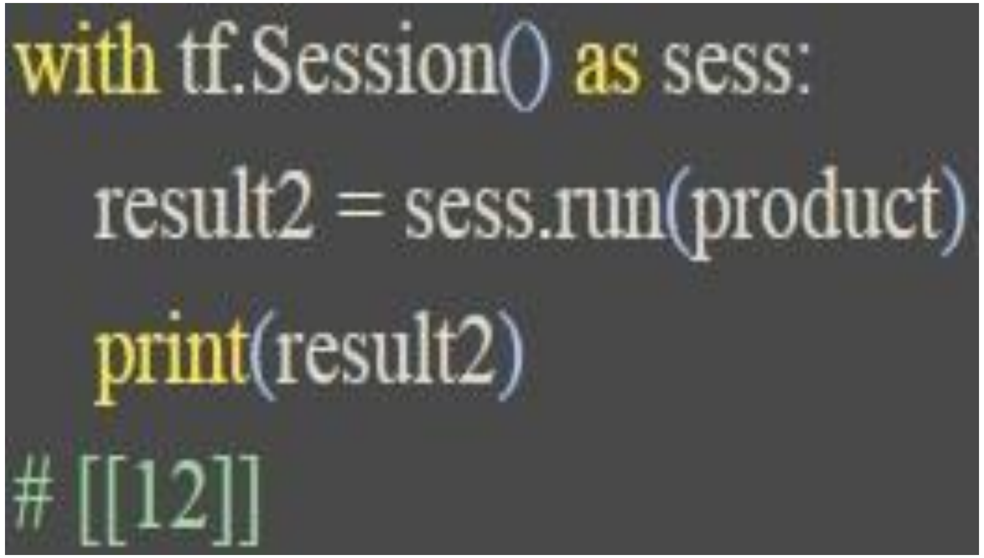
- After you define a dataflow graph, you need to create a TensorFlow session to run parts of the graph.
- Two ways to create a session :

-method 1: create a variable for a session

-method 2: use with to create a session



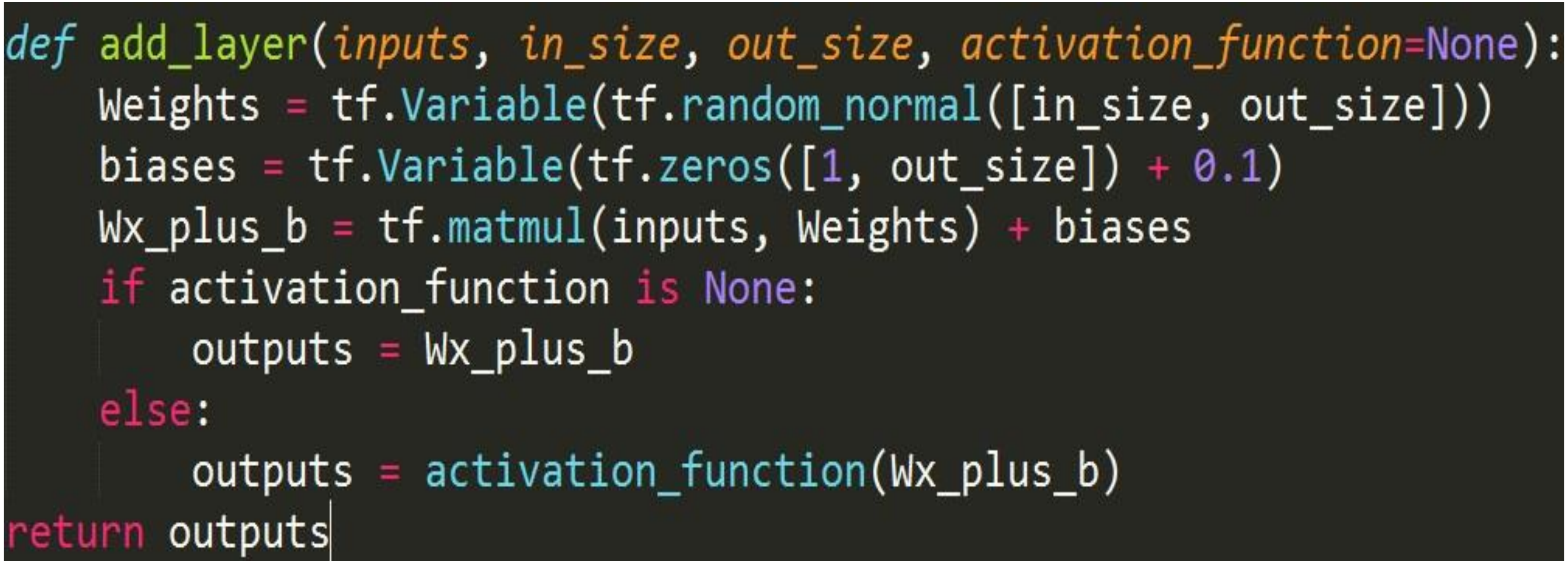
```
sess = tf.Session()
result = sess.run(product)
print(result)
sess.close()
# [[12]]
```



```
with tf.Session() as sess:
    result2 = sess.run(product)
    print(result2)
# [[12]]
```

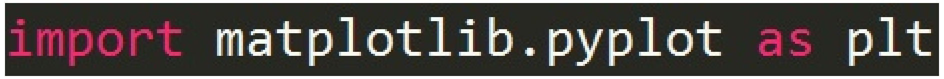
# Define add_layer

- We can define a function add_layer for further additions of layers in our neural networks.

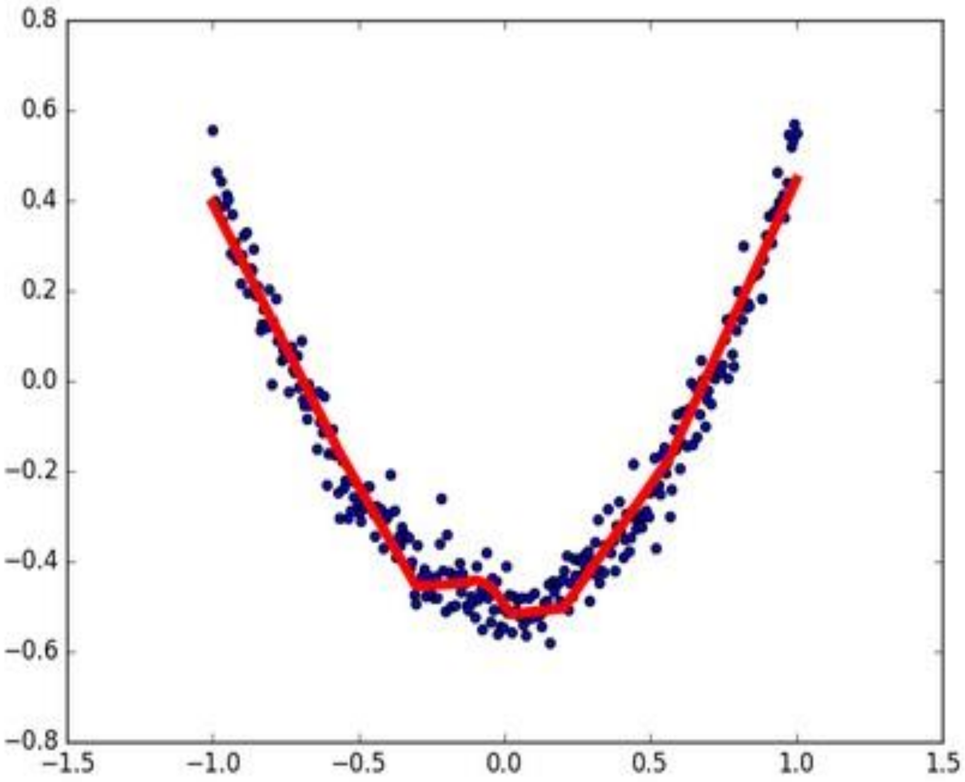


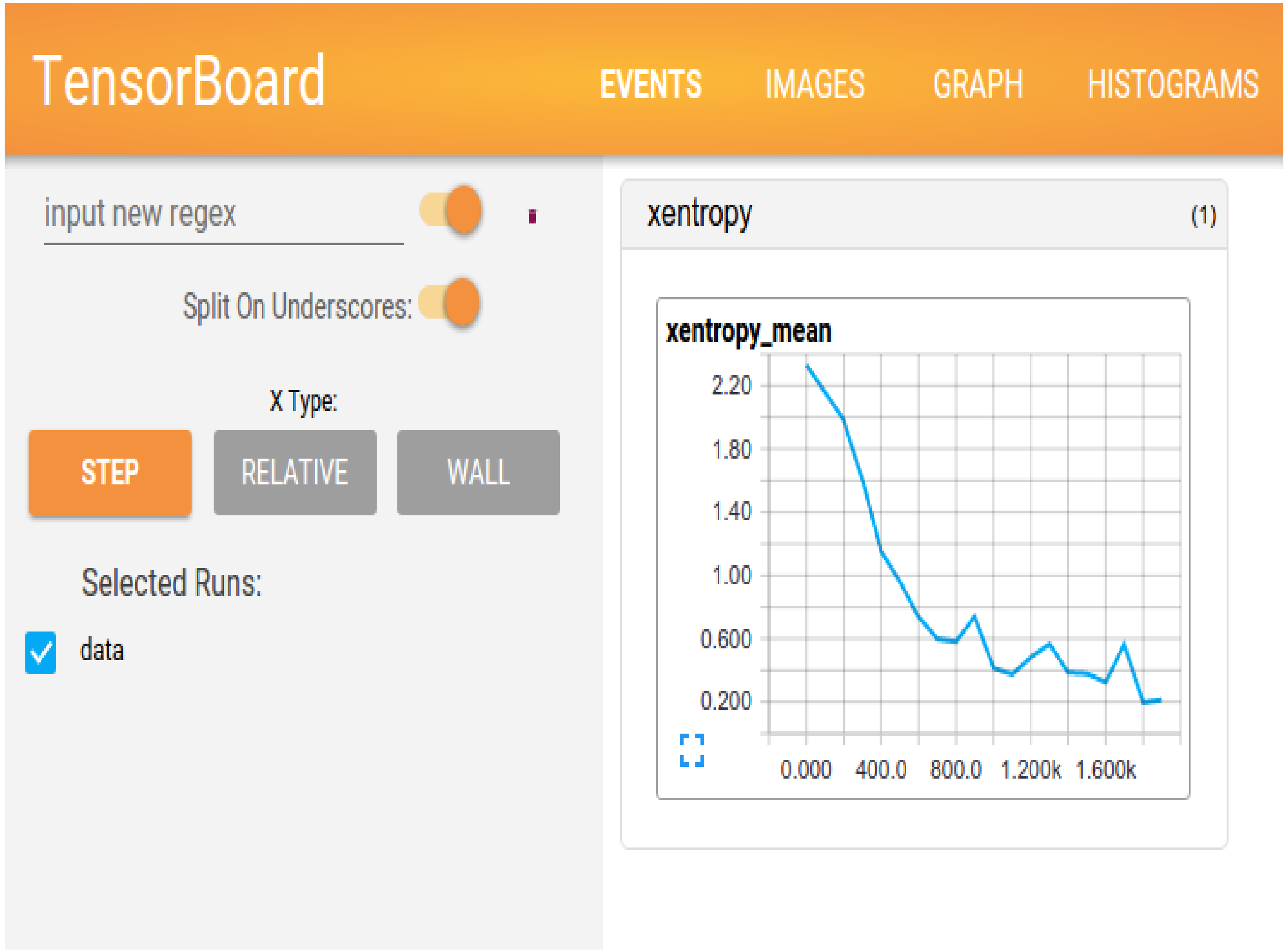
```
def add_layer(inputs, in_size, out_size, activation_function=None):
    Weights = tf.Variable(tf.random_normal([in_size, out_size]))
    biases = tf.Variable(tf.zeros([1, out_size]) + 0.1)
    Wx_plus_b = tf.matmul(inputs, Weights) + biases
    if activation_function is None:
        outputs = Wx_plus_b
    else:
        outputs = activation_function(Wx_plus_b)
return outputs
```

# Result Visualization



```python
import matplotlib.pyplot as plt
```





Split On Underscores: ●

X Type: STEP

Selected Runs:

☑ data

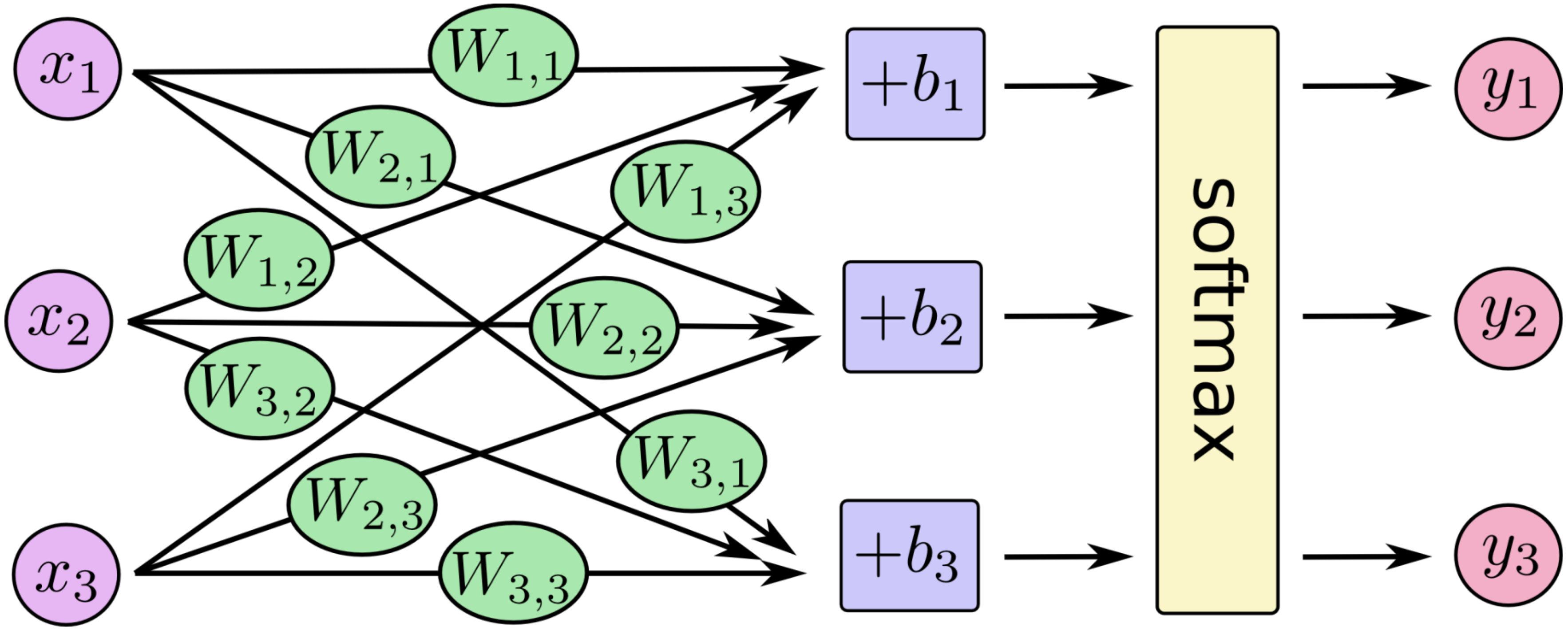
# General Principle

To build a neural network with Tensorflow, you can follow the steps below:

1) Import the modules you need.
2) Define an add_layer function to construct layers.
3) Define the variables and initialize them.
4) Create a session to perform the operation.
5) Build the NN, and send data with placeholders and feed_dict.
6) Train and test the NN, and observe the results with Tensorboard.

# MNIST Softmax Regression

**y=softmax(Wx+b)**



```
import tensorflow as tf
```

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

```
x = tf.placeholder(tf.float32, [None, 784])
```

```
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
```

```
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

```
y_ = tf.placeholder(tf.float32, [None, 10])
```

```
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y),
                                      reduction_indices=[1]))
```

```
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

```
sess = tf.InteractiveSession()

tf.global_variables_initializer().run()

for _ in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))

accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_:
                                                  mnist.test.labels}))
sess.close()
```

# Reference

I. Morvan Zhou’s youtube channel & website
https://www.youtube.com/playlist?list=PLXO45tsB95cKI5AIlf5TxxFPzb-0zeVZ8
https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/

II. Tensorflow
https://www.tensorflow.org/

III. Lectures
https://cs224d.stanford.edu/lectures/CS224d-Lecture7.pdf

http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS17.html

# THANKS