Introduction to Keras and Tensorflow

Tensorflow & Keras

- Tensorflow framework is created, maintained and used internally by Google.
- Used to run deep learning models across multiple GPUs and CPUs.
- Same code can be used to deploy across a local CPU, cloud GPU, or Android Device.
- It makes use of data flow graphs to build models.

Tensorflow & Keras

- ✓ **Keras** is an open source neural network library written in Python.
- ✓ It is a high level API that can run on top of Tensorflow, Theano and CNTK.
- ✓ The purpose of Keras framework is to use Tensorflow functions in an easier way.
- ✓ It can run on both CPUs and GPUs.

Top 5 uses cases of Tensorflow

Tensorflow is mainly used for Classification, Understanding, Perception, Discovering, Prediction and Creation.

Use Case I

Voice/Sound Recognition

- ✓ Voice Recognition - Mostly used in IOT & Automotive Security
- ✓ Sentiment Analysis - Mostly used in CRM
- ✓ Flaw Detection (Engine Noise) - Mostly used in Aviation and Automotive Industry

Use Case I

Voice/Sound Recognition

- ✓ Voice Activated Assistants - Such as Apple’s Siri, Google Now, Microsoft’s Cortana, Amazon’s Alexa
- ✓ Speech-to-Text - Transcribing the spoken word as Text.
- ✓ Sound Based Applications - Tensorflow algorithms standing in for customer service agents and route customers to the relevant information they need, faster than actual service agents.

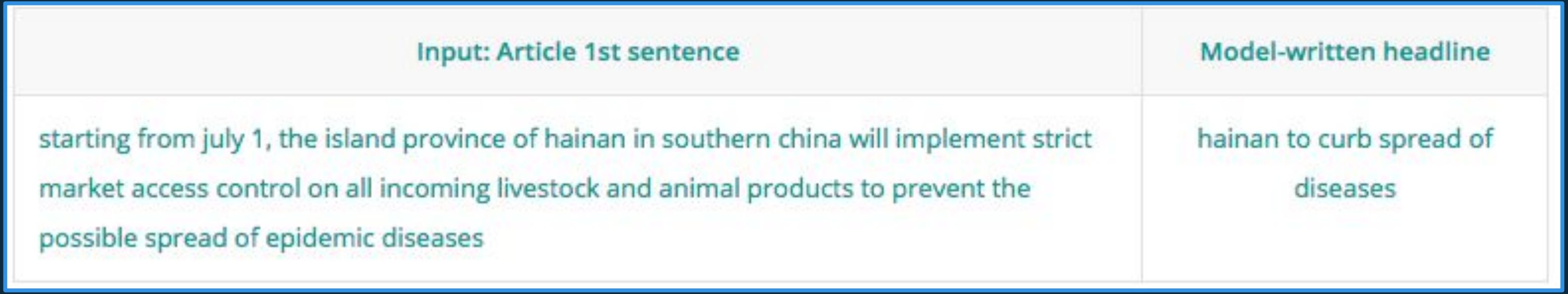
Use Case II

Text Based Applications

- Sentiment Analysis - Social Media
- Threat Detection - Social Media, Government
- Fraud Detection - Insurance, Finance
- Language Detection - Google Translate, Translating Legal Jargons from contracts into plain text.

Use Case II
Text Based Applications

- ✓ Text Summarisation - Producing headlines for news articles

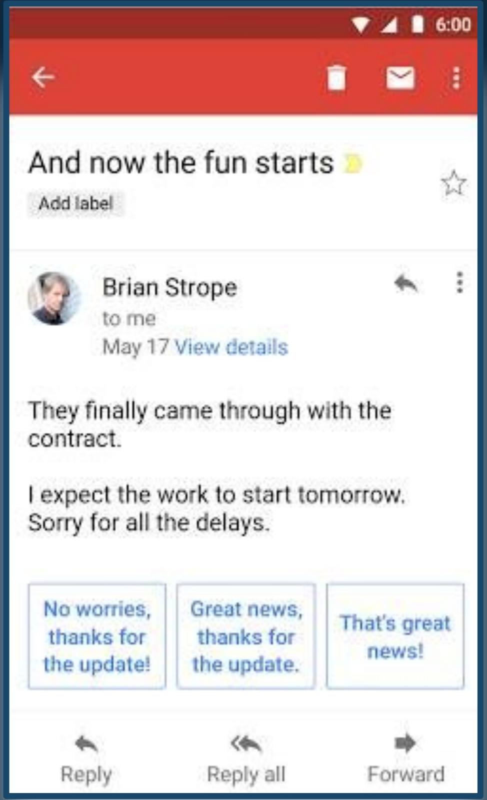


Input: Article 1st sentence	Model-written headline
starting from july 1, the island province of hainan in southern china will implement strict market access control on all incoming livestock and animal products to prevent the possible spread of epidemic diseases	hainan to curb spread of diseases

Use Case II
Text Based Applications

- ✓ Generating Email Responses - Google’s Smart Reply

_Full Paper on *Smart Reply : Automated Response Suggestion For Email*



And now the fun starts

Brian Strope
to me
May 17 View details

They finally came through with the contract.

I expect the work to start tomorrow. Sorry for all the delays.

No worries, thanks for the update!
Great news, thanks for the update.
That's great news!
Reply
Reply all
Forward

Use Case III

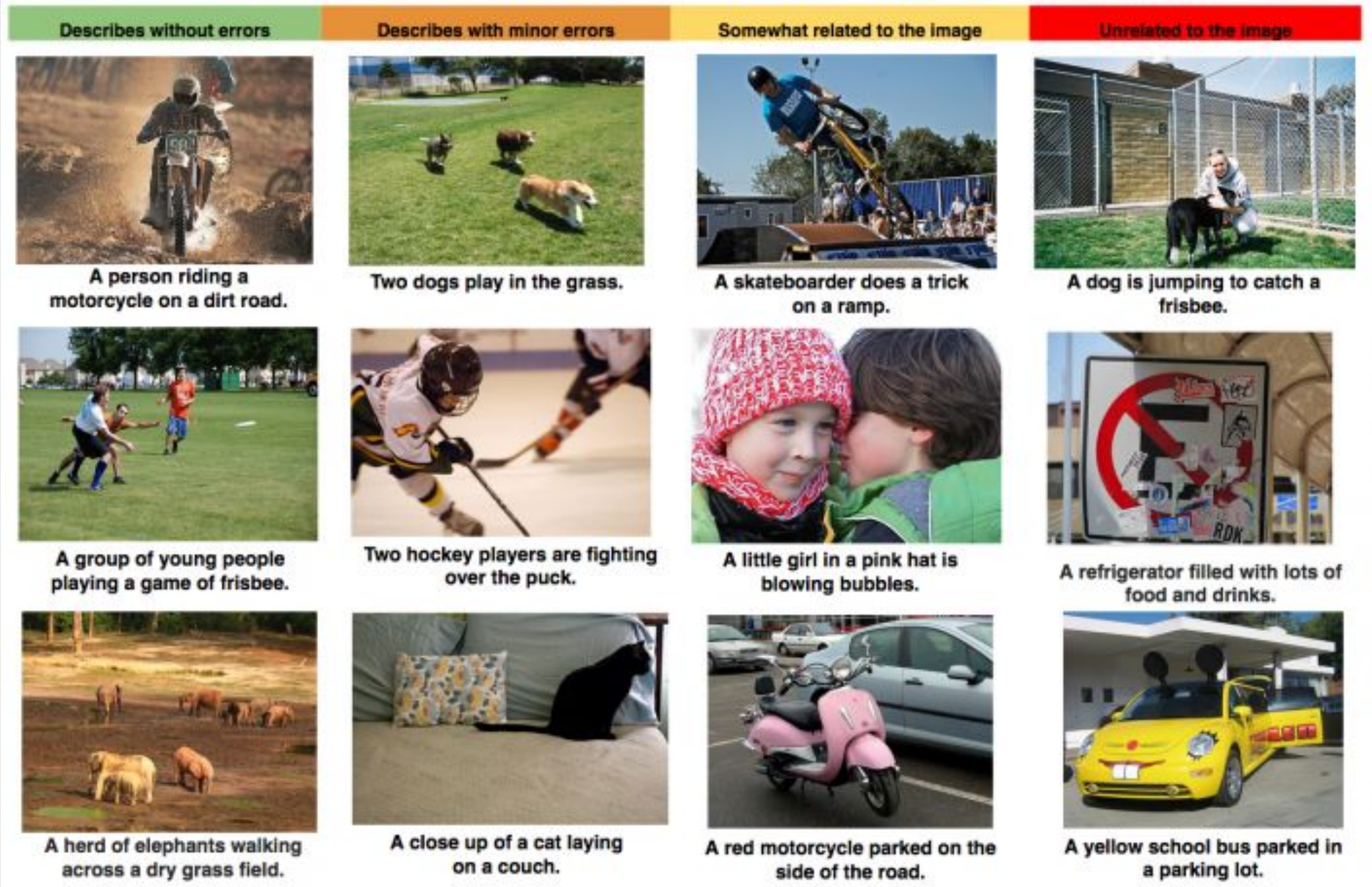
Image Recognition

✓ Face Recognition -	Social Media
✓ Image Search -	Social Media, Government
✓ Motion Detection -	Insurance, Finance
✓ Computer Vision -	Augmentative Reality

Use Case III

Image Recognition

✓ Face Recognition -	Identify People in images
✓ Image Search -	Recognize Objects in images
✓ Motion Detection -	Traffic Signals, Parking Lots
✓ Computer Vision -	Detecting patterns, 3D Image modeling



Describes without errors	Describes with minor errors	Somewhat related to the image	Unrelated to the image
A person riding a motorcycle on a dirt road.	Two dogs play in the grass.	A skateboarder does a trick on a ramp.	A dog is jumping to catch a frisbee.
A group of young people playing a game of frisbee.	Two hockey players are fighting over the puck.	A little girl in a pink hat is blowing bubbles.	A refrigerator filled with lots of food and drinks.
A herd of elephants walking across a dry grass field.	A close up of a cat laying on a couch.	A red motorcycle parked on the side of the road.	A yellow school bus parked in a parking lot.

Use Case IV
Time Series

- **TensorFlow Time Series Algorithms** are used for analyzing time series data in order to extract meaningful statistics.
- The most common use case for Time Series is **Recommendation**.
- Analyze customer activity and compare it to the millions of other users to determine what the customer might like to purchase or watch.
- TensorFlow Time Series algorithms are mainly the field of interest to **Finance, Accounting, Government, Security and IoT with Risk Detections, Predictive Analysis and Enterprise/Resource Planning.**

Use Case V
Video Detection

- ✓ TensorFlow neural networks also work on video data.

- ✓ **Large scale Video Classification** datasets like *YouTube-8M* aim to accelerate research on
  - Large-scale video understanding
  - Representation learning
  - Noisy data modeling
  - Transfer learning
  - Domain adaptation approaches for video.

Use Case V
Video Detection

- Other use cases -
    - Motion Detection
    - Real-Time Thread Detection in Gaming
    - Airport Security

- Another Highly Relevant Project is being conducted by Nasa.
    - Designing a system with TensorFlow for orbit classification and object clustering of asteroids.
    - As a result, they can classify and predict NEOs (near earth objects).

Getting Started with Keras

Benefits of using Keras

- It allows easy and fast prototyping being user friendly, modular and extensible

- Supports both CNN and RNN as well as combinations of the two.

- Runs seamlessly on CPU and GPU.

Composing Models in Keras

There are two ways of composing a model in Keras -

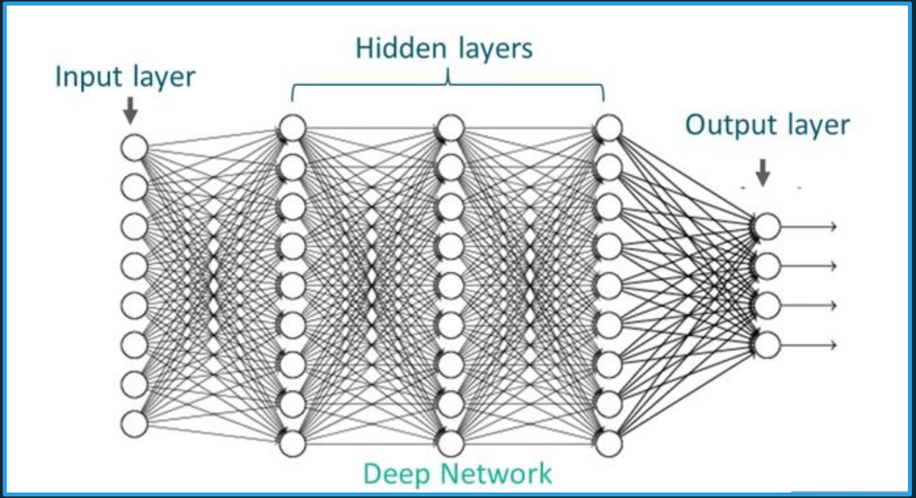
- Sequential Composition
- Functional Composition

Composing Models in Keras

Sequential Composition -

- Linear Stack of Layers

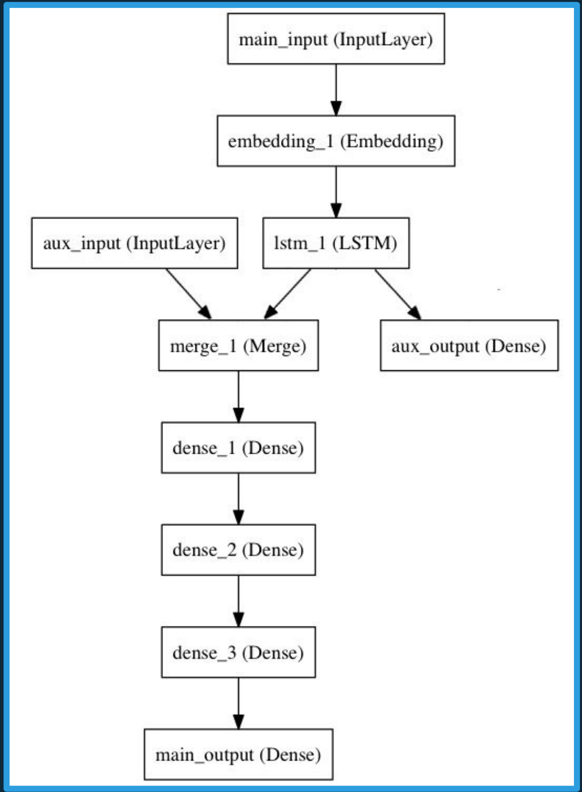
- For example, Stacking convolution layers one above the other



Composing Models in Keras

Functional Composition -

- Makes it possible to create complex models such as acyclic graphs, or multi input - multi output models.



Predefined Neural Network Layers

Keras has a number of predefined layers -

- **Dense Layer**

The net is **dense** means that each neuron in a layer is connected to all neurons located in the previous layer and to all the neurons in the following layer.

Predefined Neural Network Layers

Keras has a number of predefined layers -

- **Recurrent Neural Network Layer**

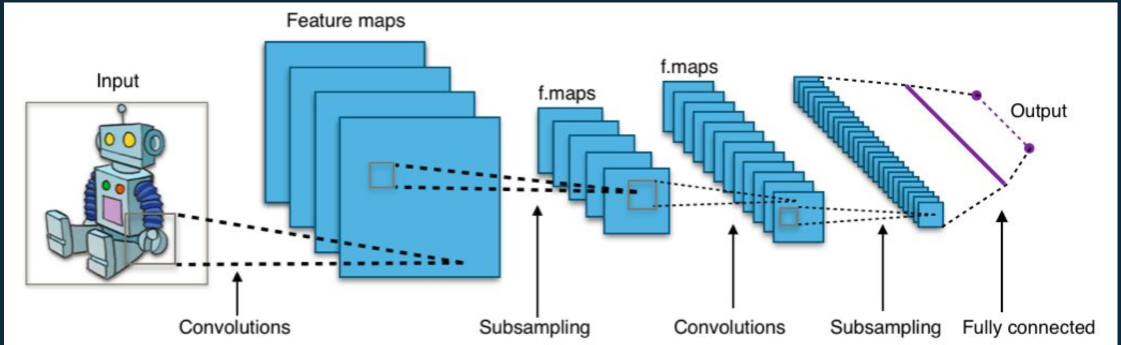
Recurrent neural networks are a class of neural networks that exploit the sequential nature of their input where the occurrence of an element in the sequence is dependent on the elements that appeared before it..

Predefined Neural Network Layers

Keras has a number of predefined layers -

- **Convolution or Pooling Layer**

ConvNets are a class of neural networks using convolutional and pooling operations for progressively learning. This learning via progressive abstraction resembles vision models that have evolved over millions of years inside the human brain.



Predefined Neural Network Layers

Keras has a number of predefined layers -

- **Regularization Layer**

Regularization is a way to prevent overfitting. Multiple layers have parameters for regularization. The following is the list of regularization parameters commonly used for dense and convolutional modules:

kernel_regularizer: Regularizer function applied to the weight matrix

bias_regularizer: Regularizer function applied to the bias

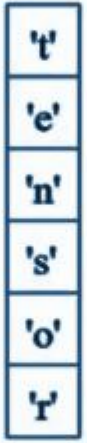
vectoractivity_regularizer: Regularizer function applied to the output of the layer (its activation)

Getting Started with Tensorflow

What are Tensors?

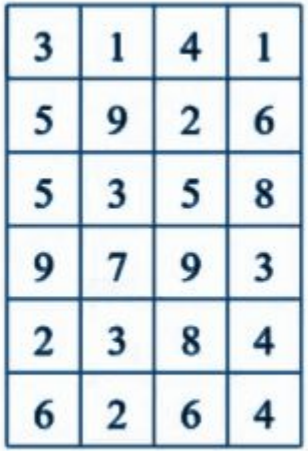
→ Tensors are the standard way of representing data in Tensorflow.

→ Tensors are multidimensional arrays, an extension of 2D matrices to data with higher dimensions.



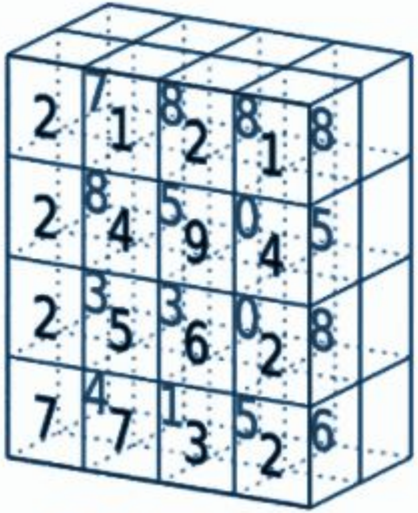
't'
'e'
'n'
's'
'o'
'r'

Tensor of dimension[1]

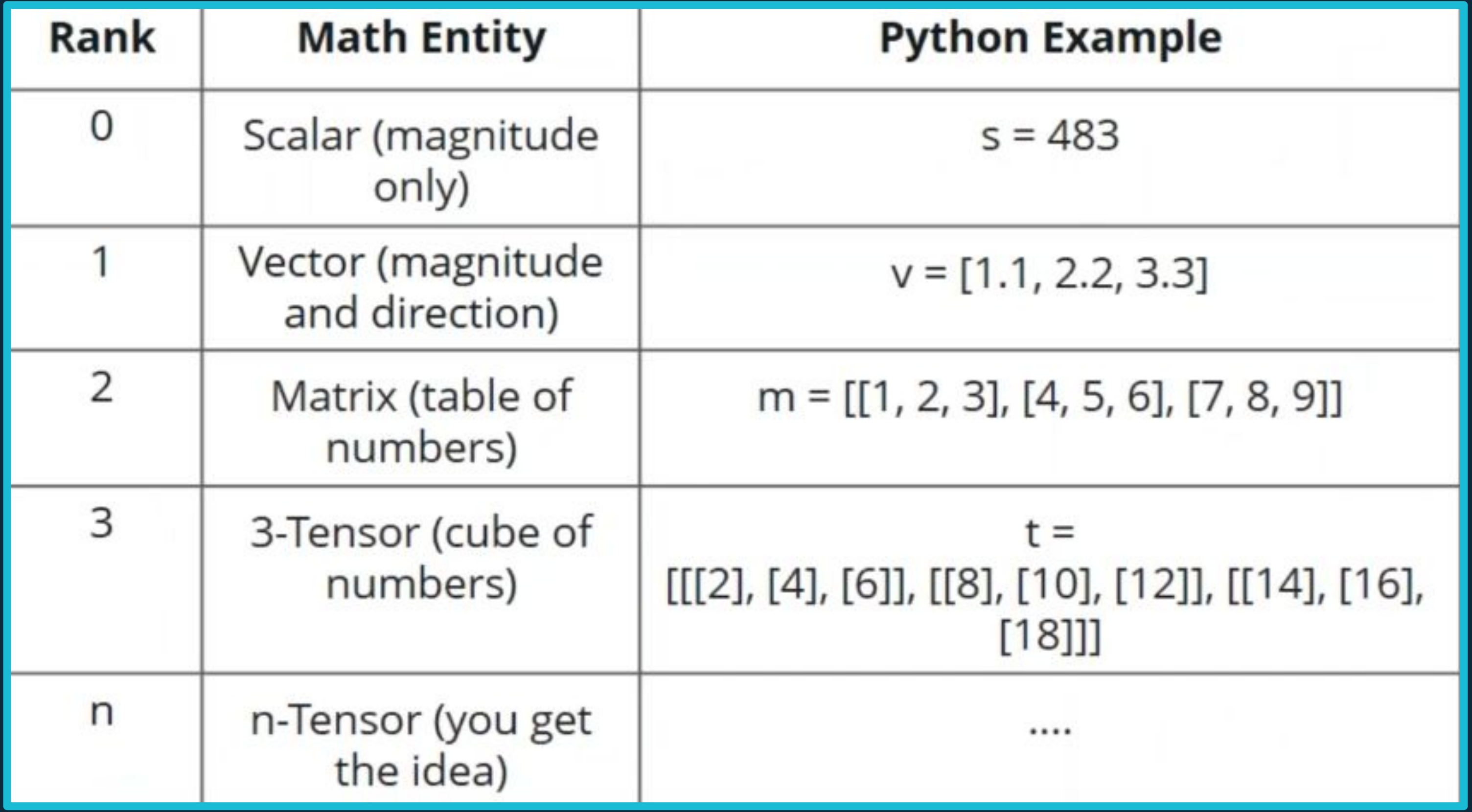


3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

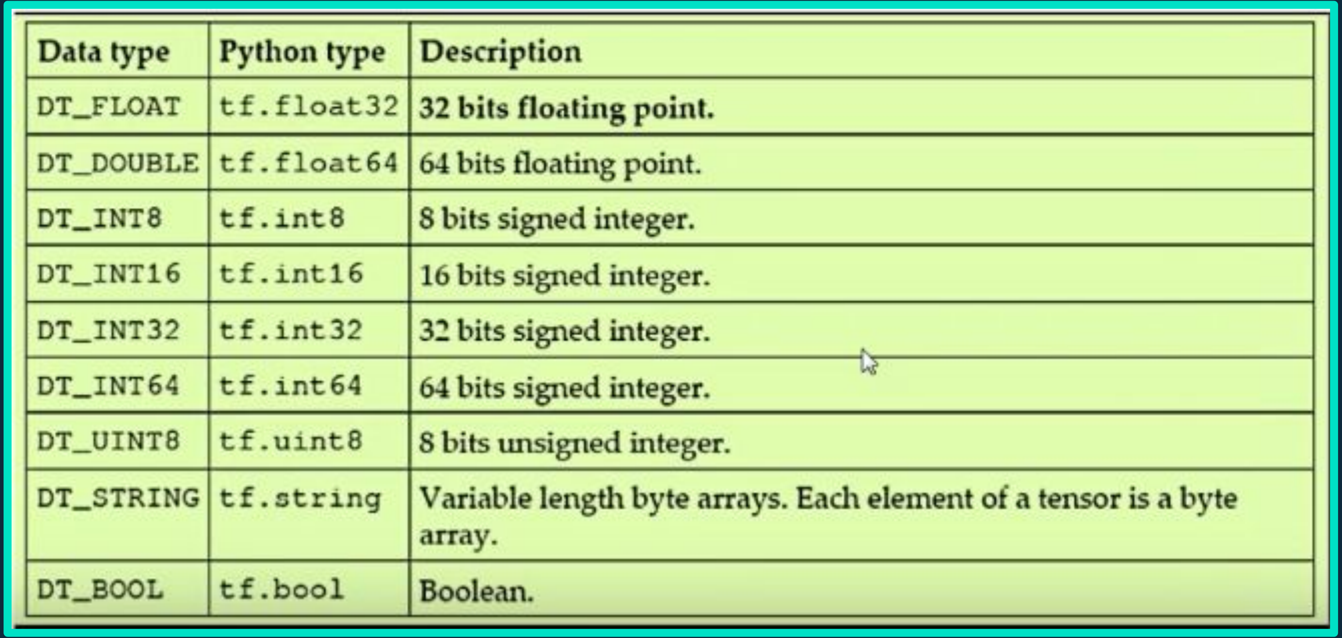
Tensor of dimensions[2]



Tensor of dimensions[3]



Rank	Math Entity	Python Example
0	Scalar (magnitude only)	s = 483
1	Vector (magnitude and direction)	v = [1.1, 2.2, 3.3]
2	Matrix (table of numbers)	m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
3	3-Tensor (cube of numbers)	t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]
n	n-Tensor (you get the idea)	....



Data type	Python type	Description
DT_FLOAT	tf.float32	32 bits floating point.
DT_DOUBLE	tf.float64	64 bits floating point.
DT_INT8	tf.int8	8 bits signed integer.
DT_INT16	tf.int16	16 bits signed integer.
DT_INT32	tf.int32	32 bits signed integer.
DT_INT64	tf.int64	64 bits signed integer.
DT_UINT8	tf.uint8	8 bits unsigned integer.
DT_STRING	tf.string	Variable length byte arrays. Each element of a tensor is a byte array.
DT_BOOL	tf.bool	Boolean.

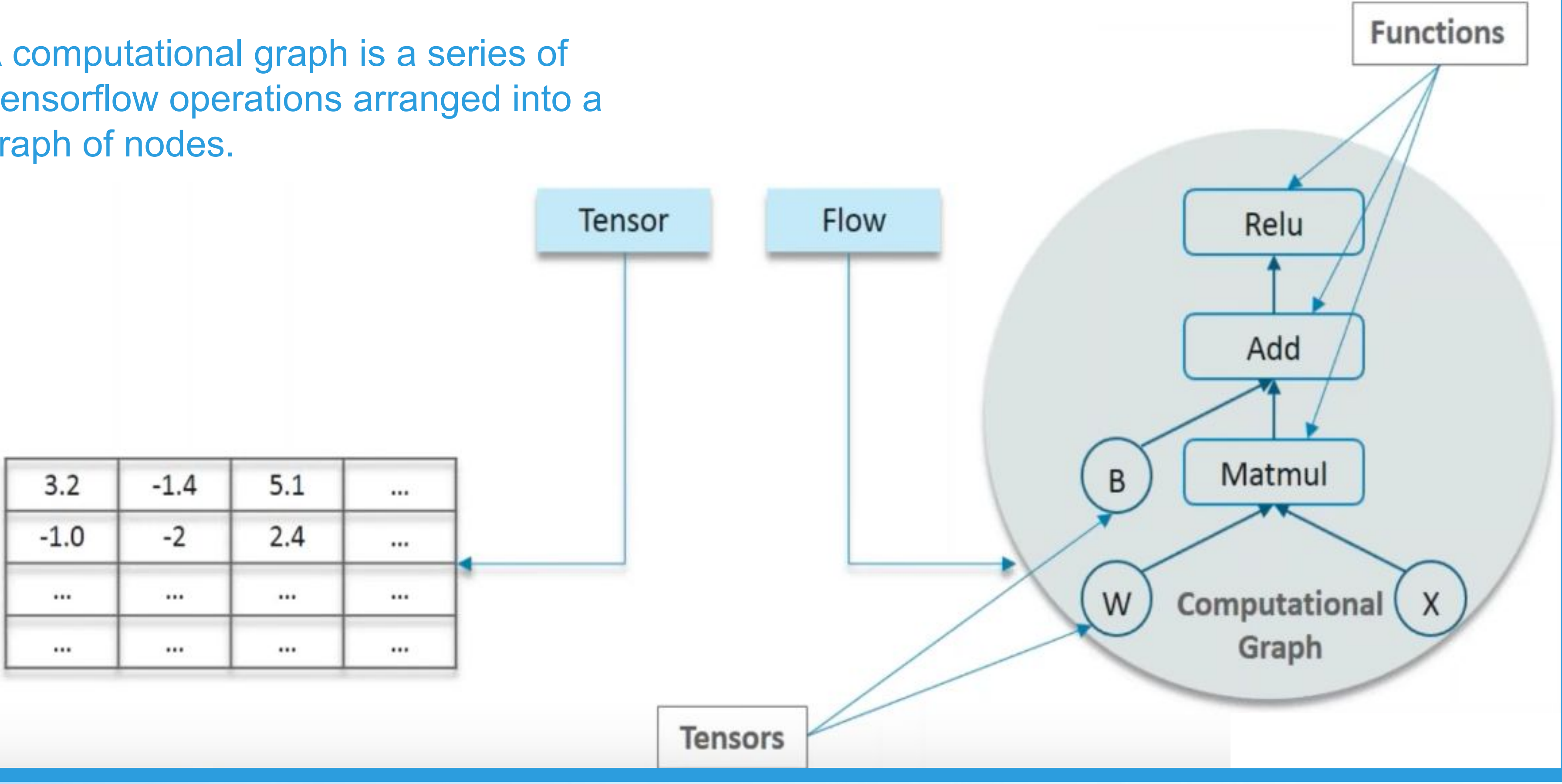
What is Tensorflow?

Tensorflow is a Python Library used to implement deep networks.

In Tensorflow, computation is approached as a data flow graph.

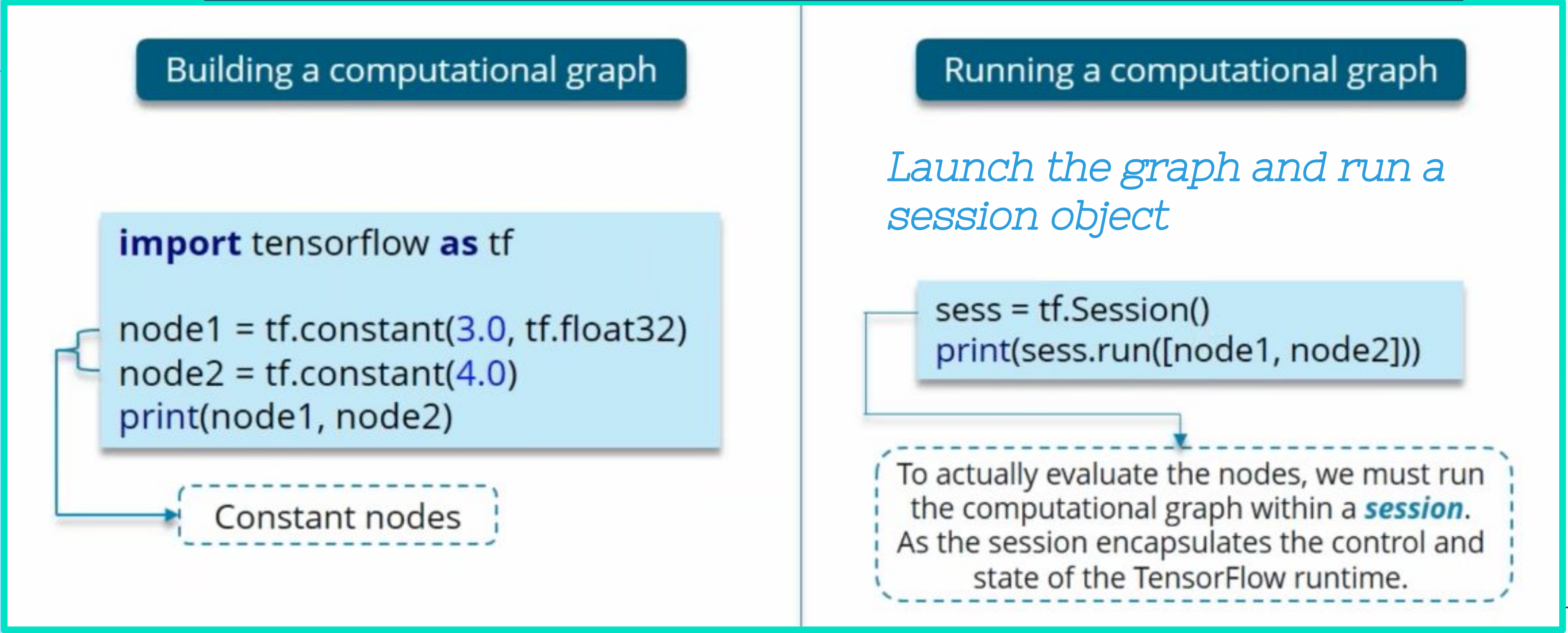
Tensorflow Computational Graph

A computational graph is a series of Tensorflow operations arranged into a graph of nodes.



3.2	-1.4	5.1	...
-1.0	-2	2.4	...
...	...	...	...
...	...	...	...

Building and Running a Graph



Building a computational graph

```
import tensorflow as tf

node1 = tf.constant(3.0, tf.float32)
node2 = tf.constant(4.0)
print(node1, node2)
```

Constant nodes

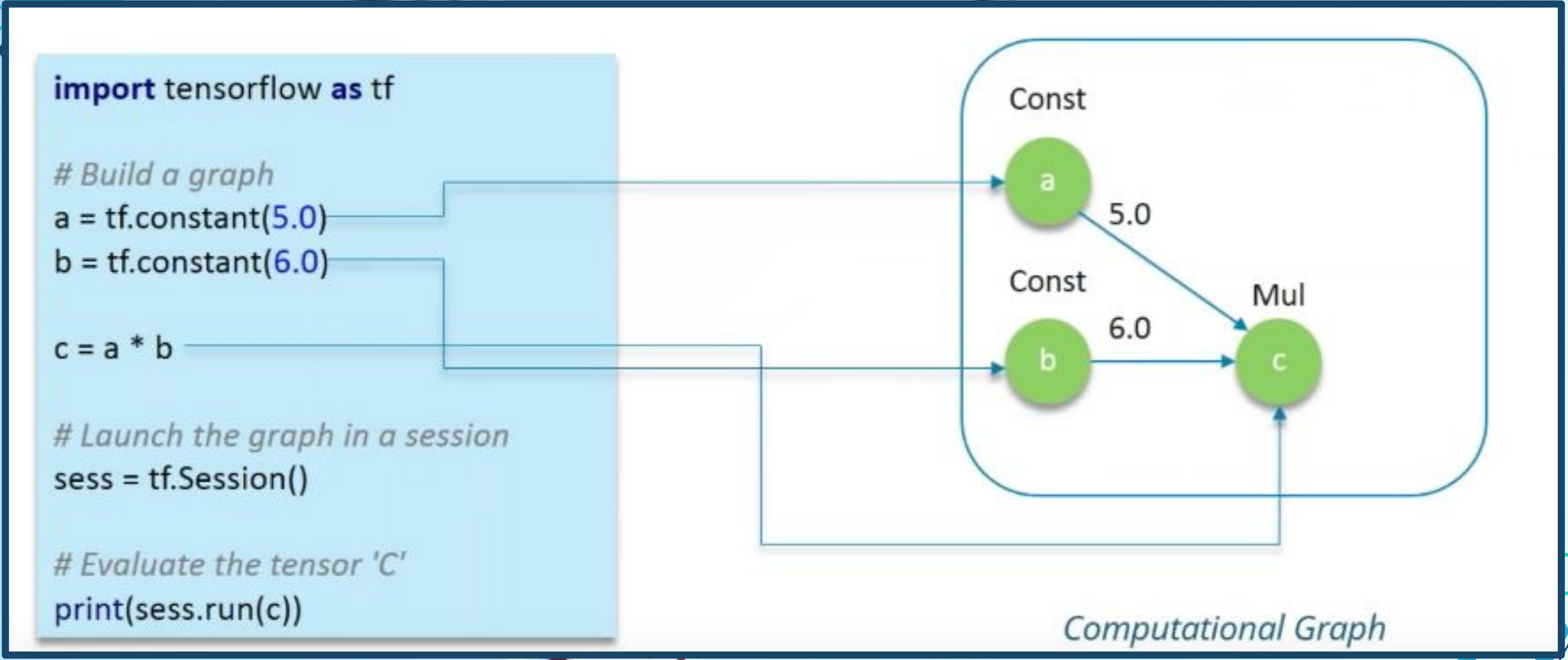
Running a computational graph

Launch the graph and run a session object

```
sess = tf.Session()
print(sess.run([node1, node2]))
```

To actually evaluate the nodes, we must run the computational graph within a ***session***. As the session encapsulates the control and state of the TensorFlow runtime.

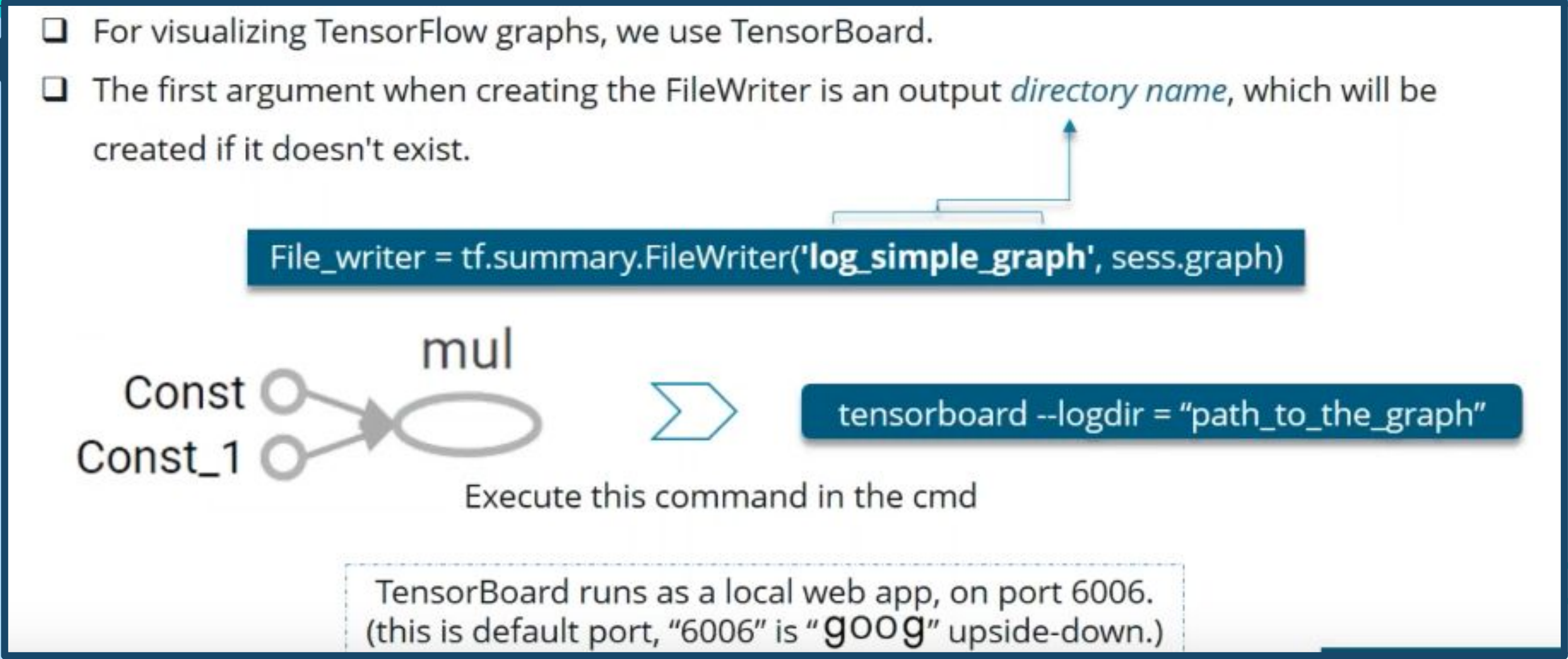
Tensorflow : Example



```
import tensorflow as tf

# Build a graph
a = tf.constant(5.0)
b = tf.constant(6.0)

c = a * b

# Launch the graph in a session
sess = tf.Session()

# Evaluate the tensor 'C'
print(sess.run(c))
```

Computational Graph

Visualizing Tensorflow Graph : *TensorBoard*



❑ For visualizing TensorFlow graphs, we use TensorBoard.

❑ The first argument when creating the FileWriter is an output *directory name*, which will be created if it doesn't exist.

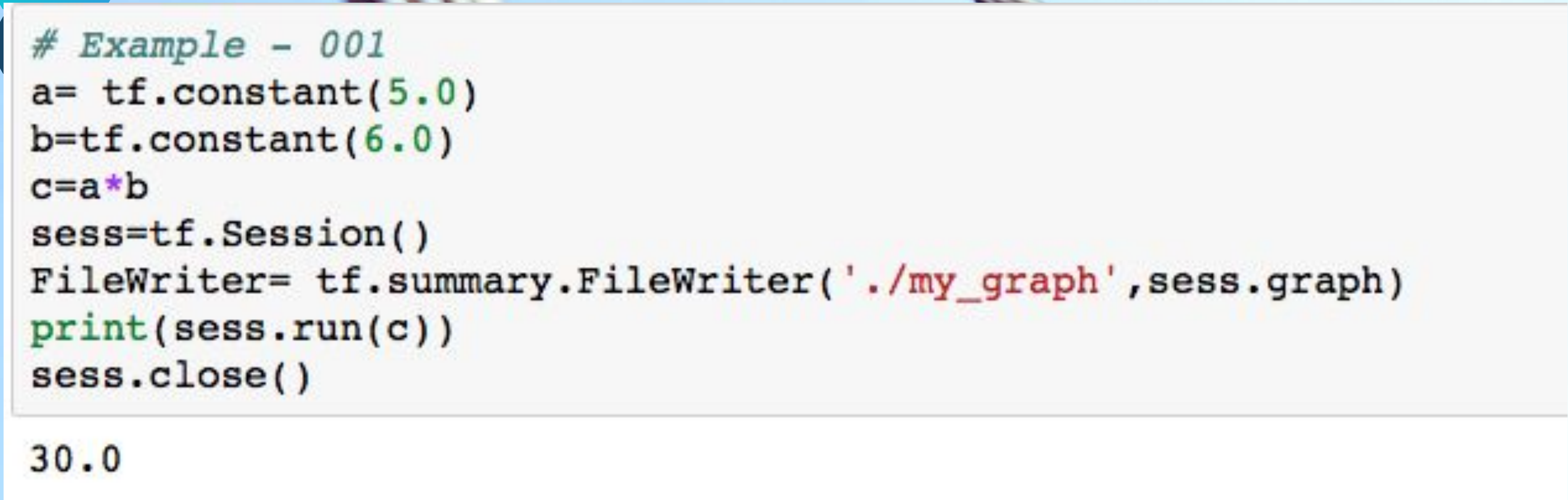
```
File_writer = tf.summary.FileWriter('log_simple_graph', sess.graph)
```

```
tensorboard --logdir = "path_to_the_graph"
```

Execute this command in the cmd

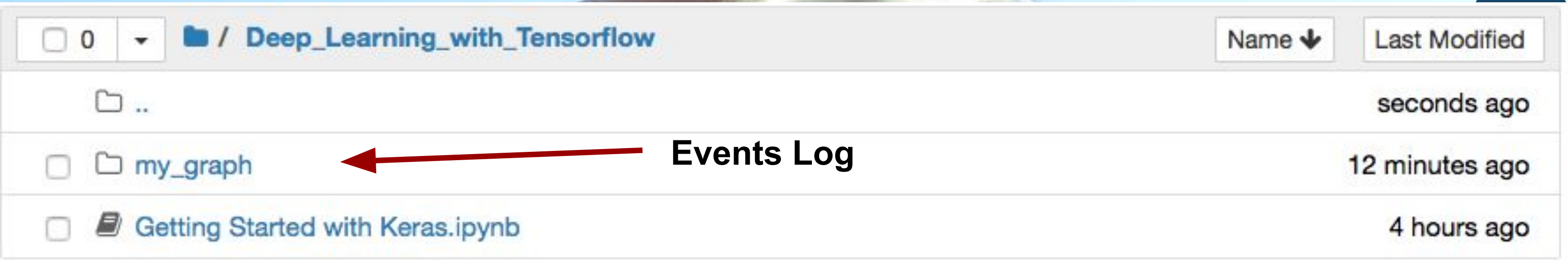
TensorBoard runs as a local web app, on port 6006.
(this is default port, "6006" is "goog" upside-down.)

Visualizing Tensorflow Graph : *TensorBoard*



```
# Example - 001
a= tf.constant(5.0)
b=tf.constant(6.0)
c=a*b
sess=tf.Session()
FileWriter= tf.summary.FileWriter('./my_graph',sess.graph)
print(sess.run(c))
sess.close()
```

```
30.0
```

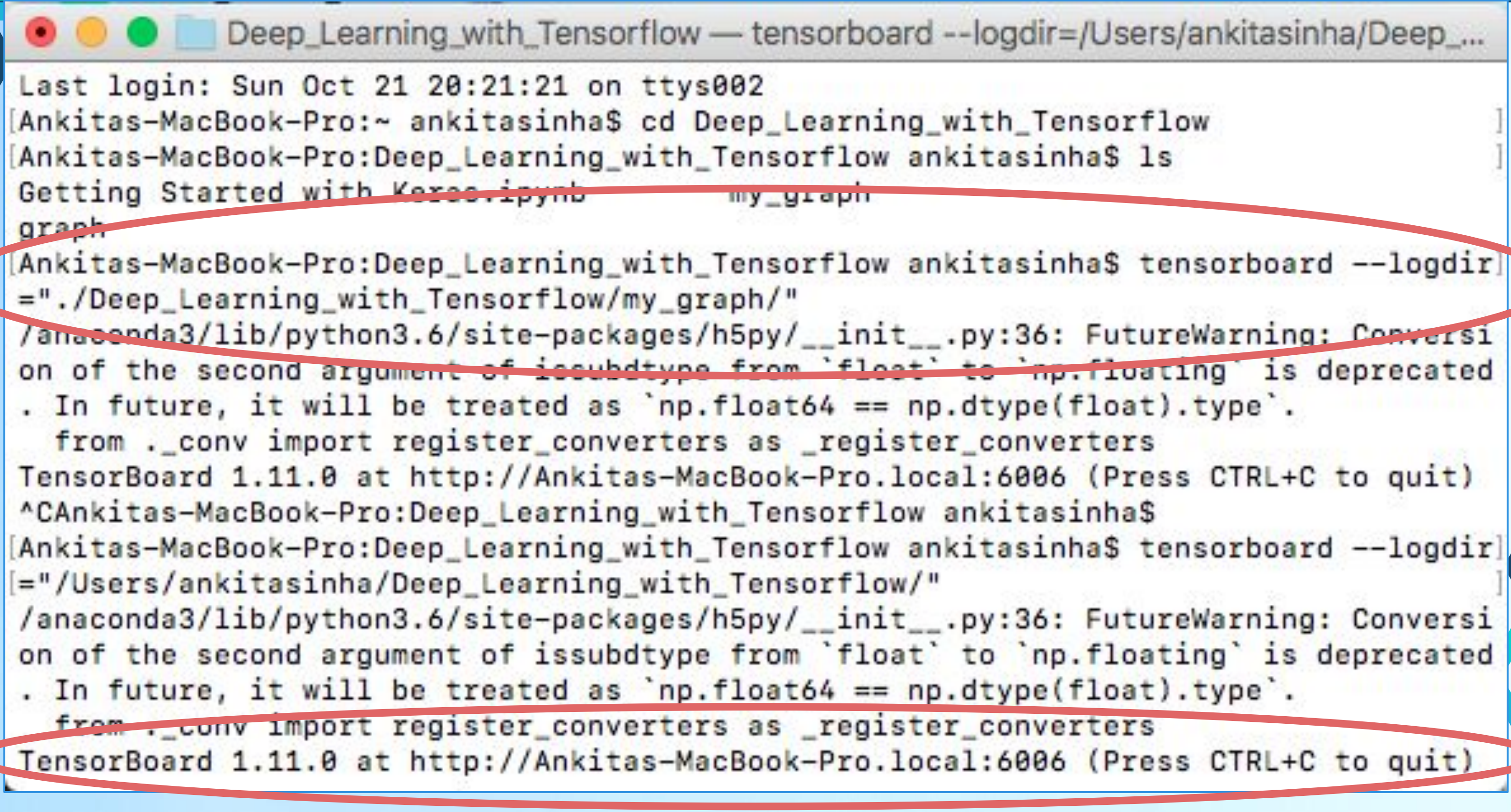


☐ 0 ▾ / Deep_Learning_with_Tensorflow — Name ↓ — Last Modified

	..	seconds ago
☐	my_graph	12 minutes ago
☐	Getting Started with Keras.ipynb	4 hours ago

Events Log

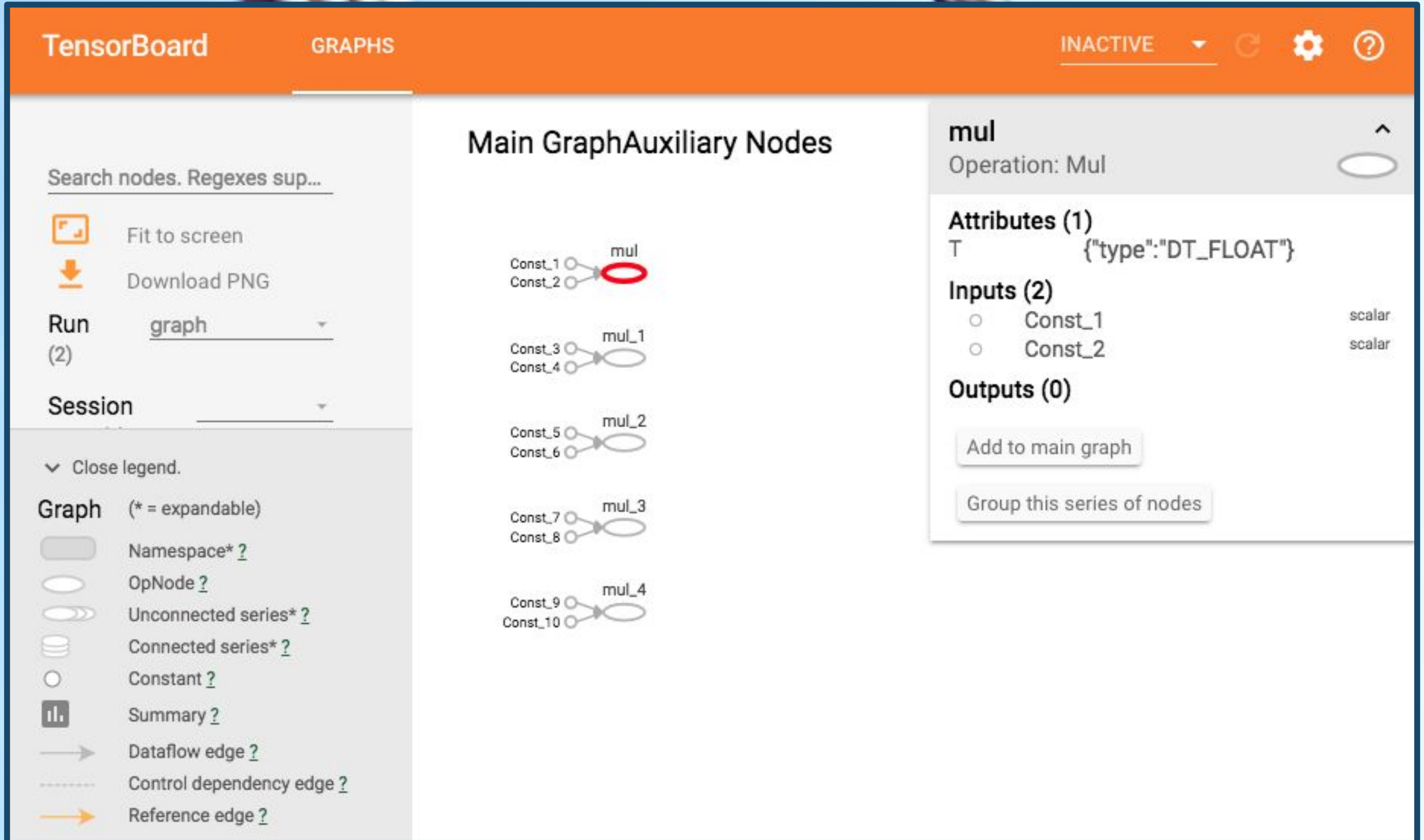
Visualizing Tensorflow Graph : *TensorBoard*



Deep_Learning_with_Tensorflow — tensorboard --logdir=/Users/ankitasinha/Deep_...

```
Last login: Sun Oct 21 20:21:21 on ttys002
[Ankitas-MacBook-Pro:~ ankitasinha$ cd Deep_Learning_with_Tensorflow          ]
[Ankitas-MacBook-Pro:Deep_Learning_with_Tensorflow ankitasinha$ ls            ]
Getting Started with Keras.ipynb        my_graph
graph
[Ankitas-MacBook-Pro:Deep_Learning_with_Tensorflow ankitasinha$ tensorboard --logdir]
="./Deep_Learning_with_Tensorflow/my_graph/"
/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversi
on of the second argument of issubdtype from `float` to `np.floating` is deprecated
. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
TensorBoard 1.11.0 at http://Ankitas-MacBook-Pro.local:6006 (Press CTRL+C to quit)
^CAnkitas-MacBook-Pro:Deep_Learning_with_Tensorflow ankitasinha$
[Ankitas-MacBook-Pro:Deep_Learning_with_Tensorflow ankitasinha$ tensorboard --logdir]
[="/Users/ankitasinha/Deep_Learning_with_Tensorflow/"                          ]
/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversi
on of the second argument of issubdtype from `float` to `np.floating` is deprecated
. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
TensorBoard 1.11.0 at http://Ankitas-MacBook-Pro.local:6006 (Press CTRL+C to quit)
```

Visualizing Tensorflow Graph : *TensorBoard*



Run (2) graph

Session

Graph (* = expandable)

- Namespace* ?
- OpNode ?
- Unconnected series* ?
- Connected series* ?
- Constant ?
- Summary ?
- Dataflow edge ?
- Control dependency edge ?
- Reference edge ?

Main GraphAuxiliary Nodes

mul

Operation: Mul

Attributes (1)

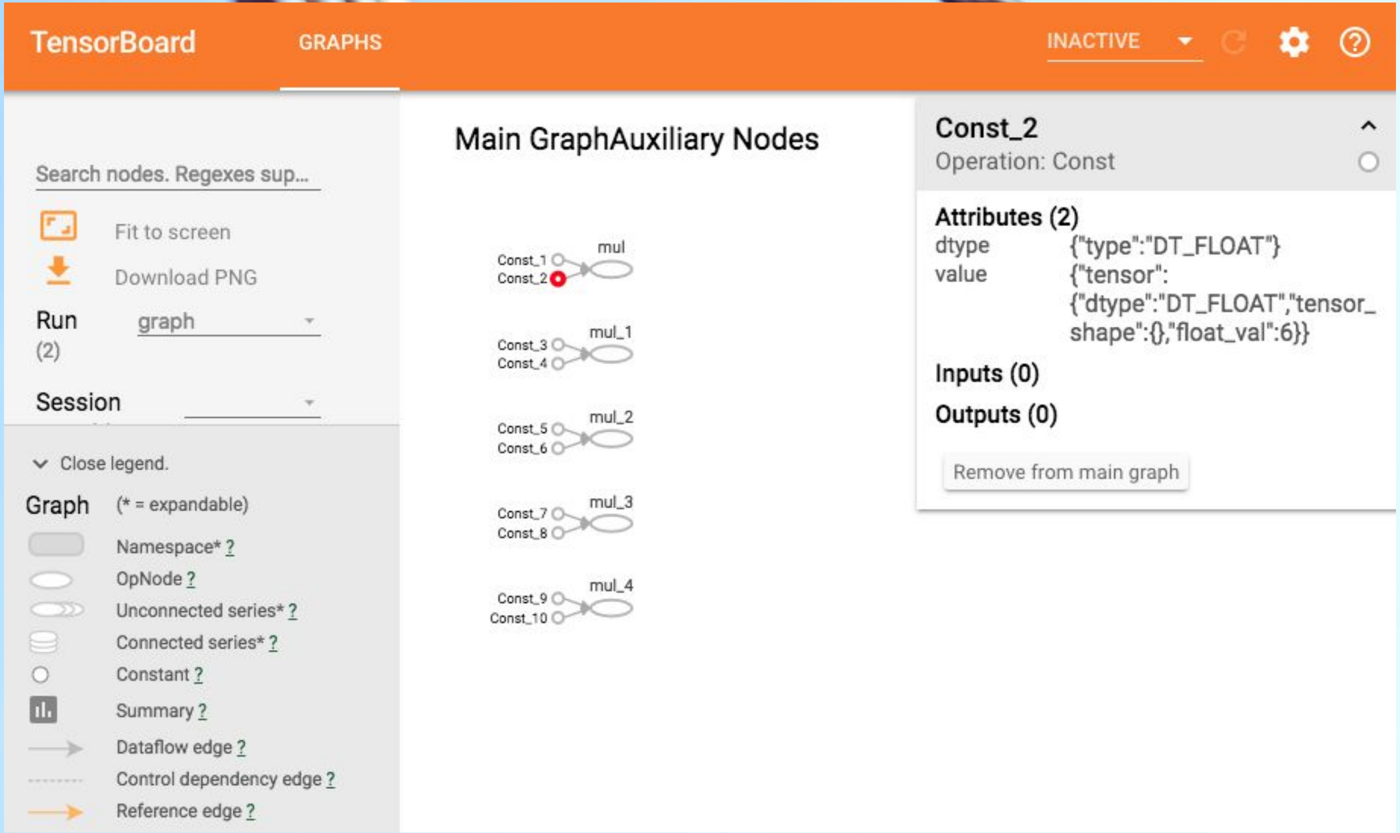
T {"type":"DT_FLOAT"}

Inputs (2)

○ Const_1	scalar
○ Const_2	scalar

Outputs (0)

Visualizing Tensorflow Graph : *TensorBoard*



Const_2
Operation: Const

Attributes (2)

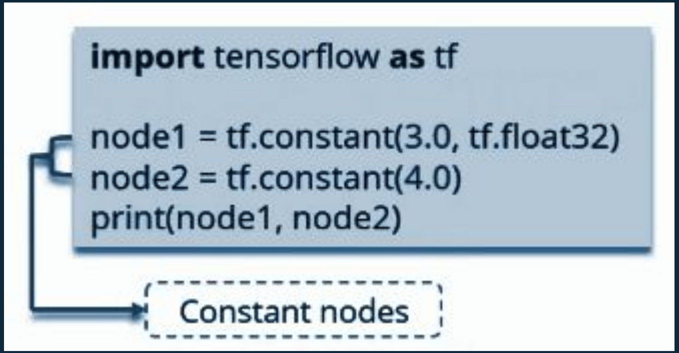
dtype	{"type":"DT_FLOAT"}
value	{"tensor": {"dtype":"DT_FLOAT","tensor_shape":{},"float_val":6}}

Inputs (0)

Outputs (0)

Constants

A type of node that takes no input and outputs a value it stores internally.

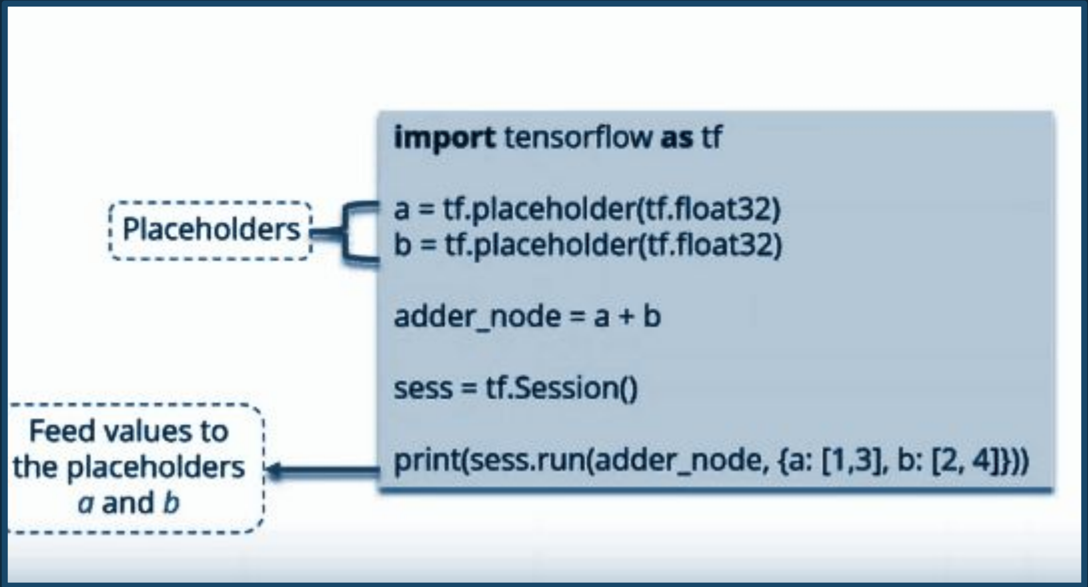


```
import tensorflow as tf

node1 = tf.constant(3.0, tf.float32)
node2 = tf.constant(4.0)
print(node1, node2)
```

Placeholder

- A graph can be parameterized to accept external inputs, known as placeholders.

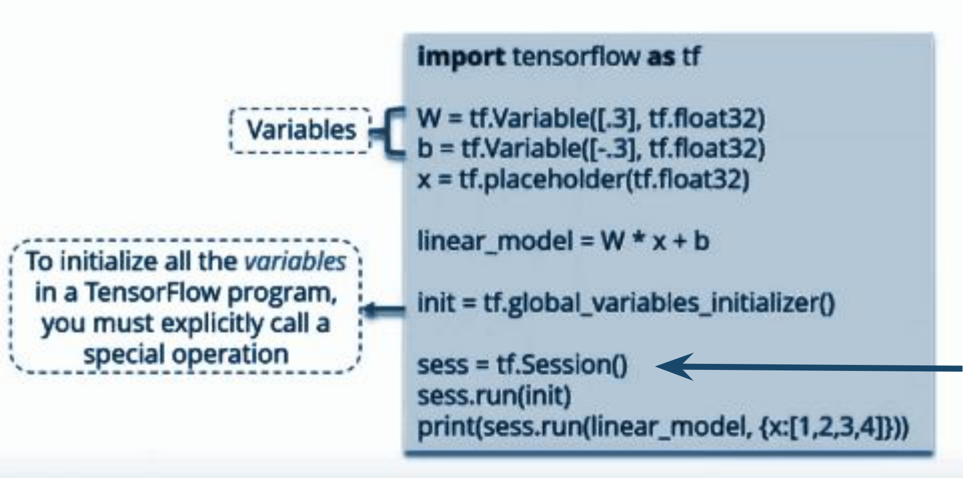
- A placeholder is a promise to provide a value later.



```
import tensorflow as tf

a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)

adder_node = a + b

sess = tf.Session()

print(sess.run(adder_node, {a: [1,3], b: [2, 4]}))
```

Variable

- To make the model trainable, we need to be able to modify the graph to get new outputs with the same input.
- Variables allow us to add trainable parameters to a graph.
- Variables are in memory buffers containing tensors.



```
import tensorflow as tf

W = tf.Variable([.3], tf.float32)
b = tf.Variable([-.3], tf.float32)
x = tf.placeholder(tf.float32)

linear_model = W * x + b

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(linear_model, {x:[1,2,3,4]}))
```

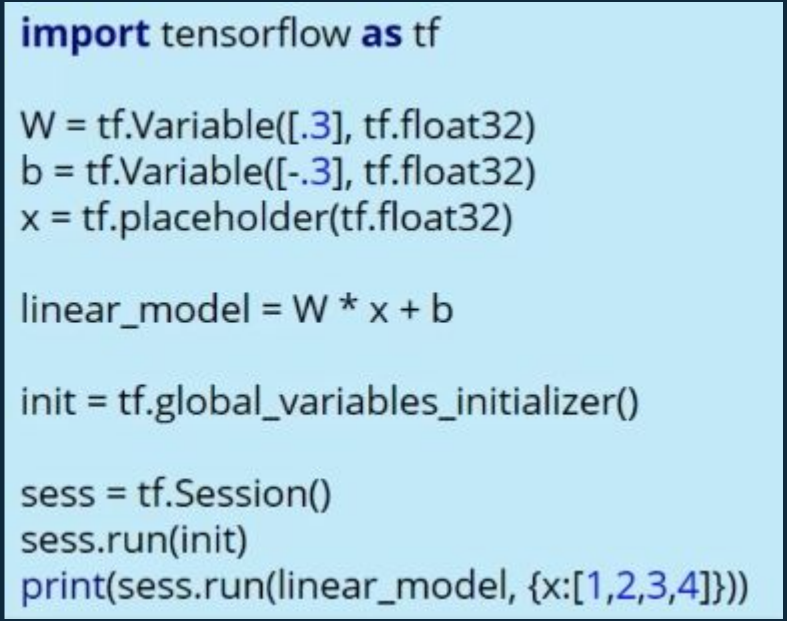
Variables

To initialize all the *variables* in a TensorFlow program, you must explicitly call a special operation

Launch Model and run init

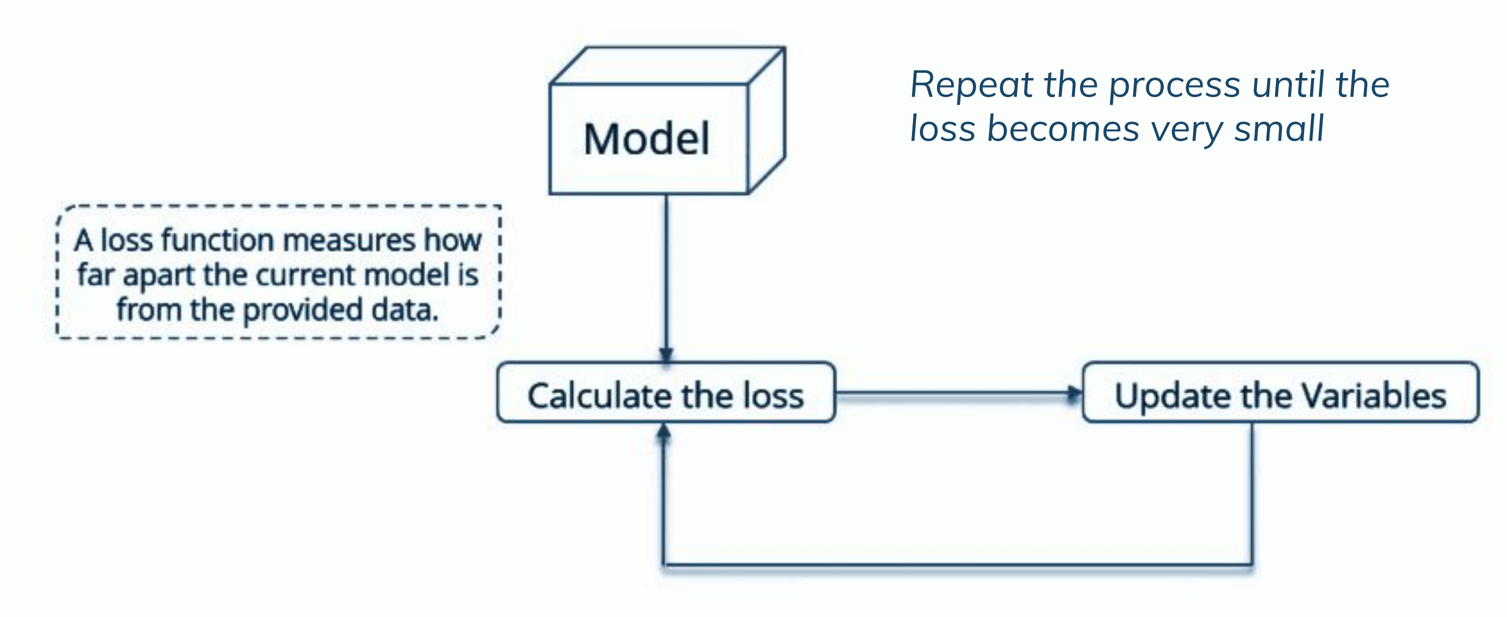
Executing The Model

Simple Linear Model



```
import tensorflow as tf

W = tf.Variable([.3], tf.float32)
b = tf.Variable([-.3], tf.float32)
x = tf.placeholder(tf.float32)

linear_model = W * x + b

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(linear_model, {x:[1,2,3,4]}))
```

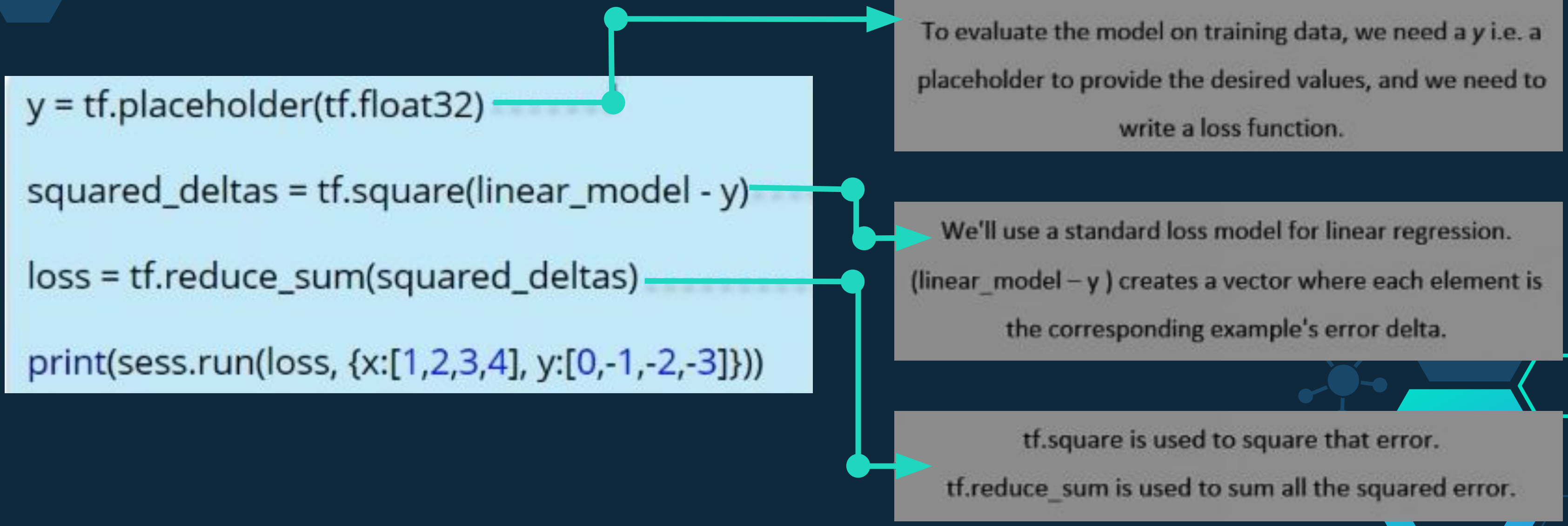
How to Increase the Efficiency of the Model


Model
A loss function measures how far apart the current model is from the provided data.
Calculate the loss
Update the Variables

Repeat the process until the loss becomes very small

Calculating the Loss

- In order to understand how good the model is, we should know the loss or error.



```
y = tf.placeholder(tf.float32)

squared_deltas = tf.square(linear_model - y)

loss = tf.reduce_sum(squared_deltas)

print(sess.run(loss, {x:[1,2,3,4], y:[0,-1,-2,-3]}))
```

To evaluate the model on training data, we need a *y* i.e. a placeholder to provide the desired values, and we need to write a loss function.

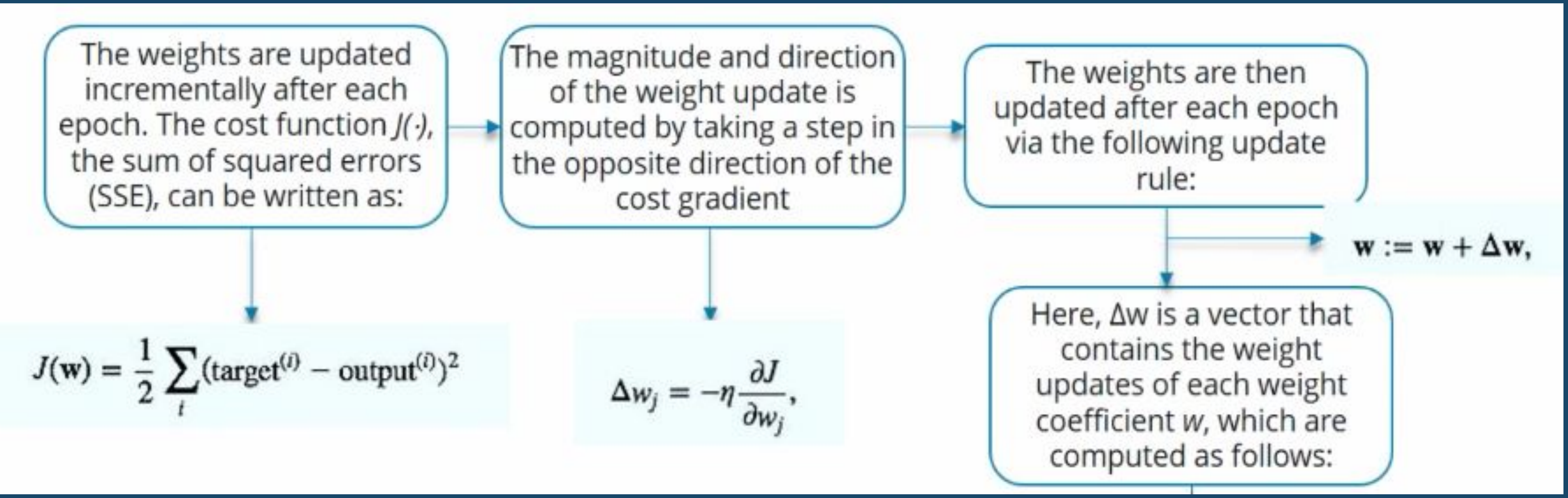
We'll use a standard loss model for linear regression. (linear_model – y) creates a vector where each element is the corresponding example's error delta.

tf.square is used to square that error.
tf.reduce_sum is used to sum all the squared error.

Reducing the Loss

- Optimizer modifies each variable according to the magnitude of the derivative of loss with respect to that variable.
- Calculating Change in the loss with the change in the variable to update value of variables.
- Gradient Descent Optimizer

Batch Gradient Descent



The weights are updated incrementally after each epoch. The cost function *J*(·), the sum of squared errors (SSE), can be written as:

$$J(\mathbf{w}) = \frac{1}{2}\sum_i \left(\text{target}^{(i)} - \text{output}^{(i)}\right)^2$$

The magnitude and direction of the weight update is computed by taking a step in the opposite direction of the cost gradient

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j},$$

The weights are then updated after each epoch via the following update rule:

$$\mathbf{w} := \mathbf{w} + \Delta\mathbf{w},$$

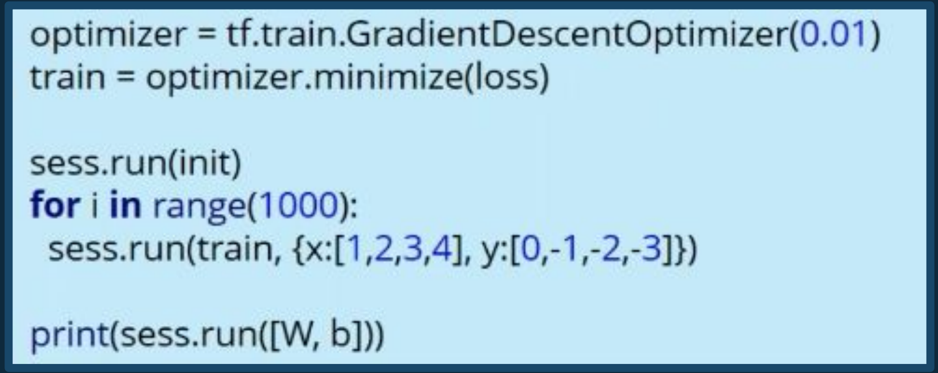
Here, Δw is a vector that contains the weight updates of each weight coefficient *w*, which are computed as follows:

Reducing the Loss

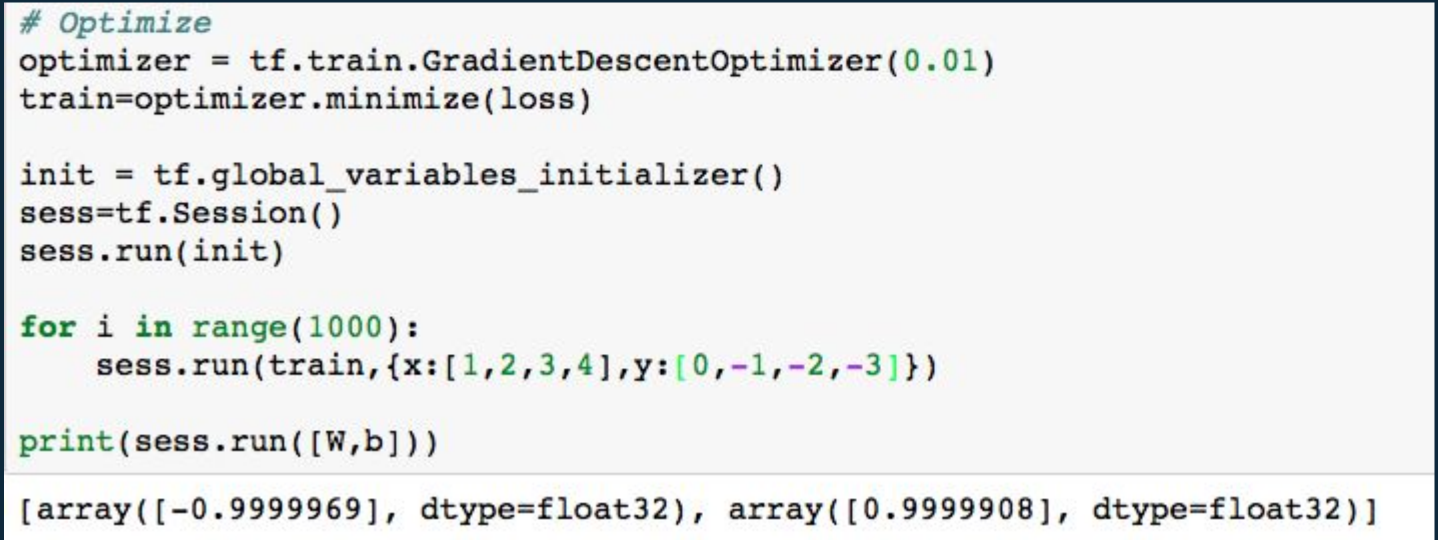
- Suppose we want to find the best parameters (W) for our learning algorithm.
- We can apply the same analogy and find the best possible values for that parameter.

Implement the Optimizer

Reduce the loss & update the variables



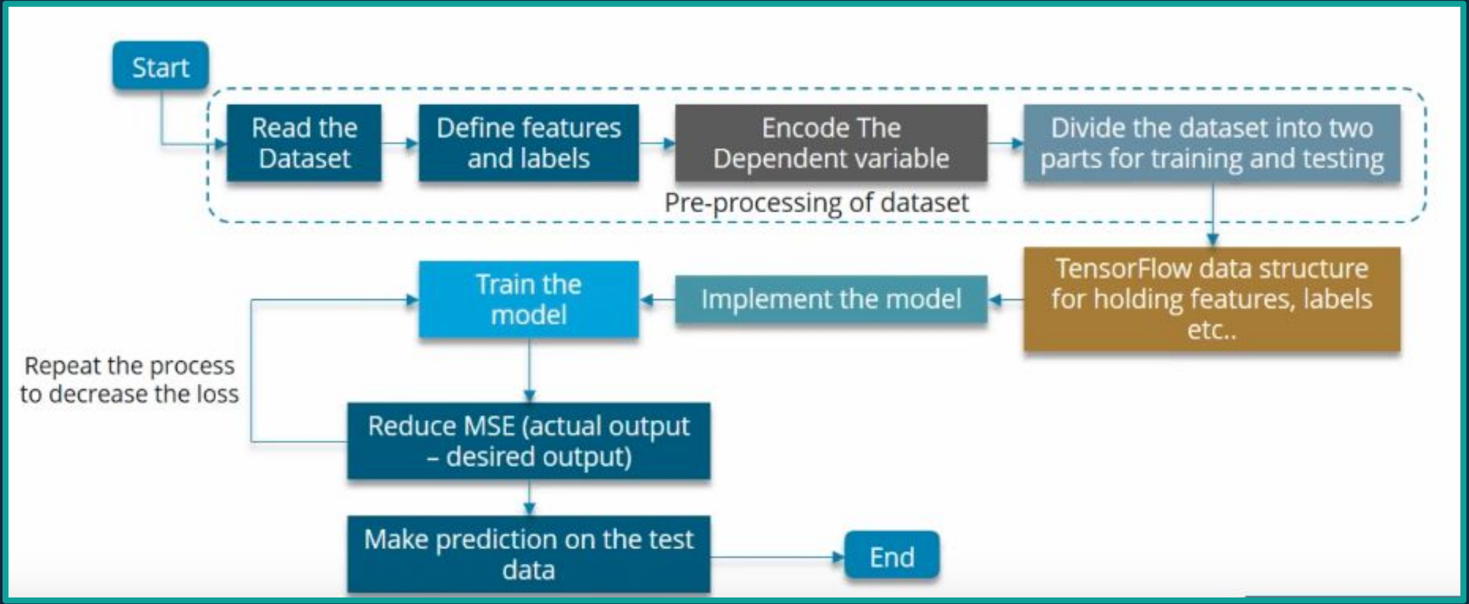
```
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)

sess.run(init)
for i in range(1000):
  sess.run(train, {x:[1,2,3,4], y:[0,-1,-2,-3]})

print(sess.run([W, b]))
```



```
# Optimize
optimizer = tf.train.GradientDescentOptimizer(0.01)
train=optimizer.minimize(loss)

init = tf.global_variables_initializer()
sess=tf.Session()
sess.run(init)

for i in range(1000):
    sess.run(train,{x:[1,2,3,4],y:[0,-1,-2,-3]})

print(sess.run([W,b]))
```

```
[array([-0.9999969], dtype=float32), array([0.9999908], dtype=float32)]
```

Implementing the Use Case



Start

Read the Dataset

Define features and labels

Encode The Dependent variable

Divide the dataset into two parts for training and testing

Pre-processing of dataset

TensorFlow data structure for holding features, labels etc..

Implement the model

Train the model

Reduce MSE (actual output – desired output)

Repeat the process to decrease the loss

Make prediction on the test data

End

Further Reading

- ◇ *Keras : The Python Learning Library*
- ◇ *Tensorflow : An Open Source Deep Learning Framework*

More on Keras & Tensorflow

- Installation
- Hands-on LAB
- Implementation of use case

Next

- **Hands-On Lab on Deep Learning Projects**
  - Deep Learning Frameworks
  - Feed Forward Neural Network
  - CNN & RNN
  - GAN and Computer Vision
  - Neural Network & Data Processing
  - Internals of Neural Network

Thanks!

Any questions?

You can find me at:

- ◇ @ankita90
- ◇ ankita.sinha8118@gmail.com

Presentation template by SlidesCarnival