Machine Learning and Deep Learning with TensorFlow

History

1st Generation: DistBelif

Dean. et al. 2011
Major Products:

- Inception (Image Categorization)
- Google Search
- Google Traslate
- Google Photos

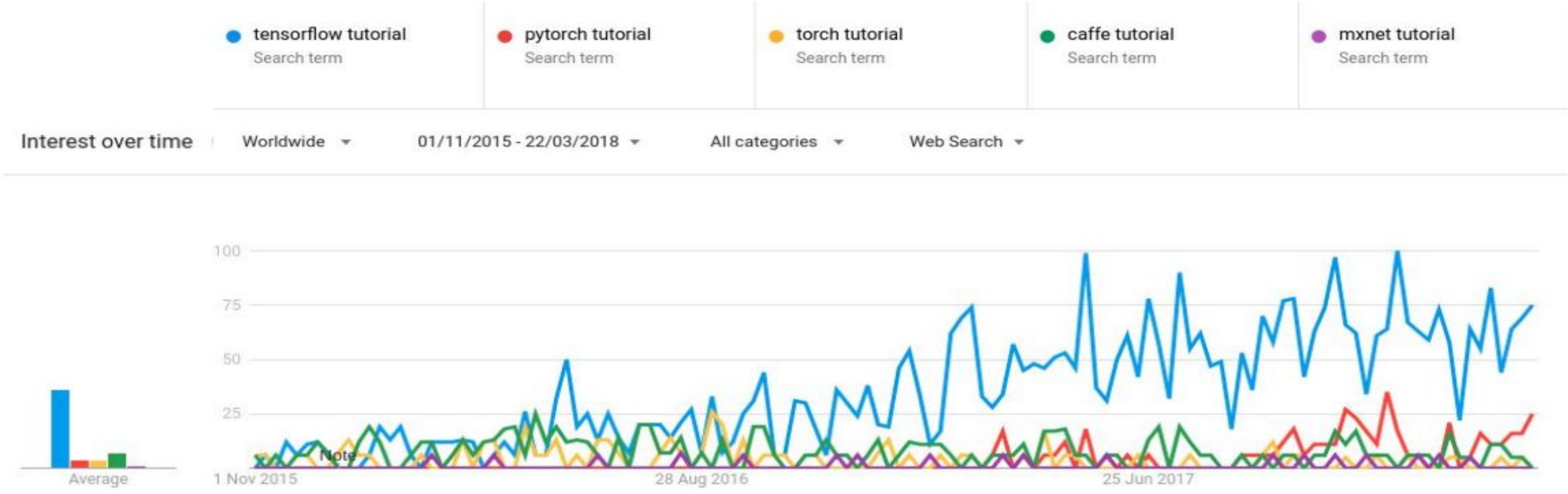
2nd Generation: TensorFlow

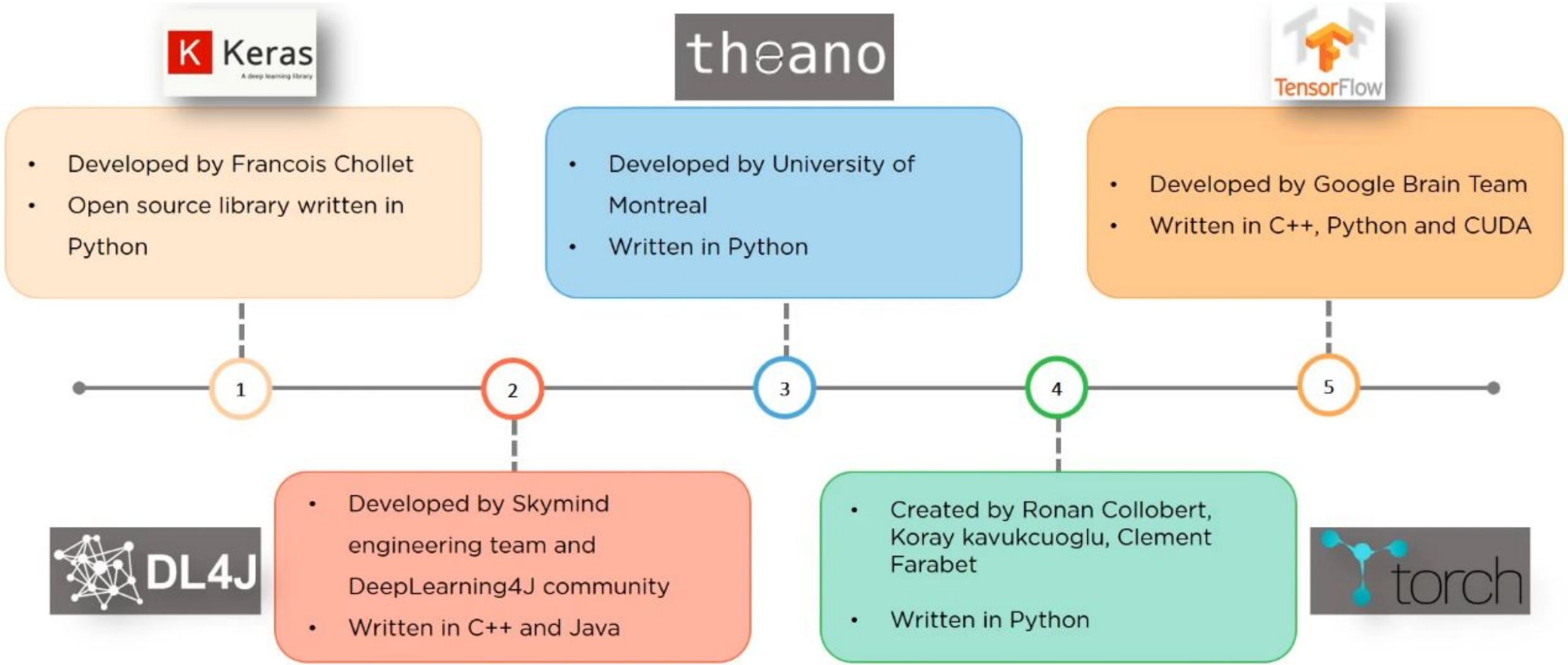
- Dean. et al. 2015 (1st November)
- Most of DistBelief users at Google have already switched to TensorFlow

Jeff Dean – One of the main developer of DistBelief and TensorFlow

Why TensorFlow

- "TensorFlow™ is an open source software library for numerical computation using data flow graphs."
- One of many frameworks for deep learning computations
- Scalable and flexible



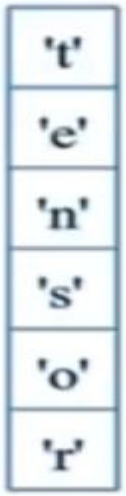

Keras
A deep learning library
Developed by Francois Chollet
Open source library written in Python
1
theano
Developed by University of Montreal
Written in Python
3
TensorFlow
Developed by Google Brain Team
Written in C++, Python and CUDA
5
DL4J
Developed by Skymind engineering team and DeepLearning4J community
Written in C++ and Java
2
Created by Ronan Collobert, Koray kavukcuoglu, Clement Farabet
Written in Python
4
torch

Companies using TensorFlow

- Google
- OpenAI
- DeepMind
- Snapchat
- Uber
- Airbus
- eBay
- Dropbox
- A bunch of startups…

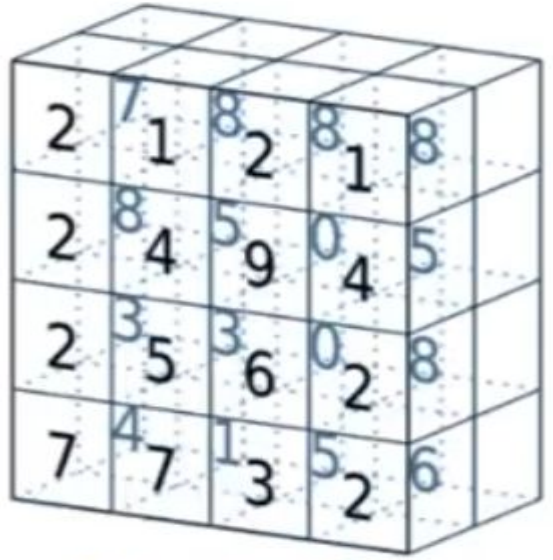
What are Tensors?

- Tensors are the standard way of representing data in Tensor Flow (deep learning).
- Tensors are multidimensional arrays, an extension of two-dimensional tables(matrices) to data with higher dimension



Tensor of dimension[1]

Tensor of dimensions[2]



Tensor of dimensions[3]

Tensors Rank

Rank	Math Entity	Python Example
0	Scalar(magnitude only)	S=434
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 the numbers)	T=[[[2],[4],[6],[[8],[10],[12]],[[14],[16],[18]]]
n	N-Tensor(you get the idea)	…

Tensor Data Types

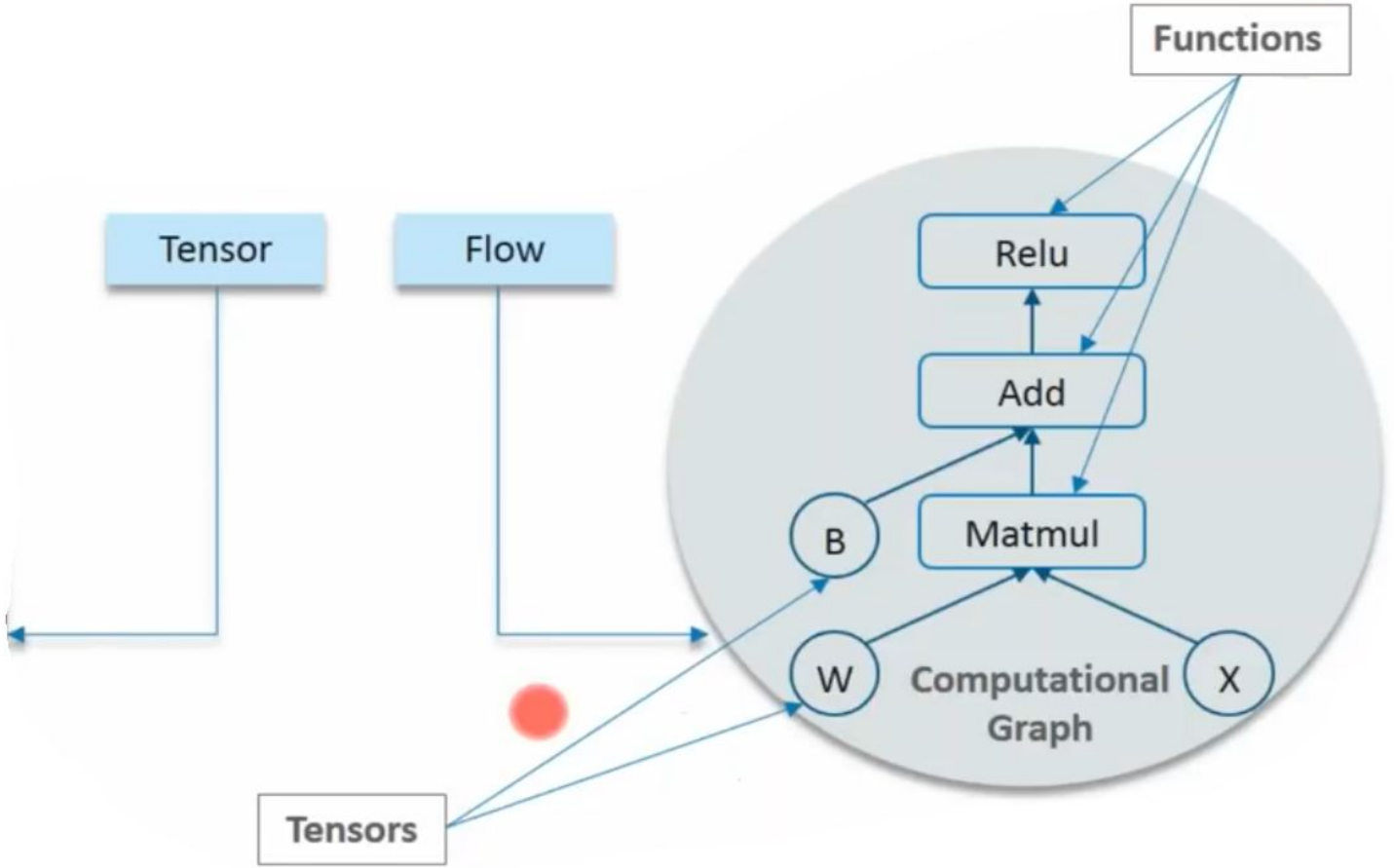
In addition to dimensionality Tensors have different data types as well,you can assign any one of these data types to a tensor.

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.

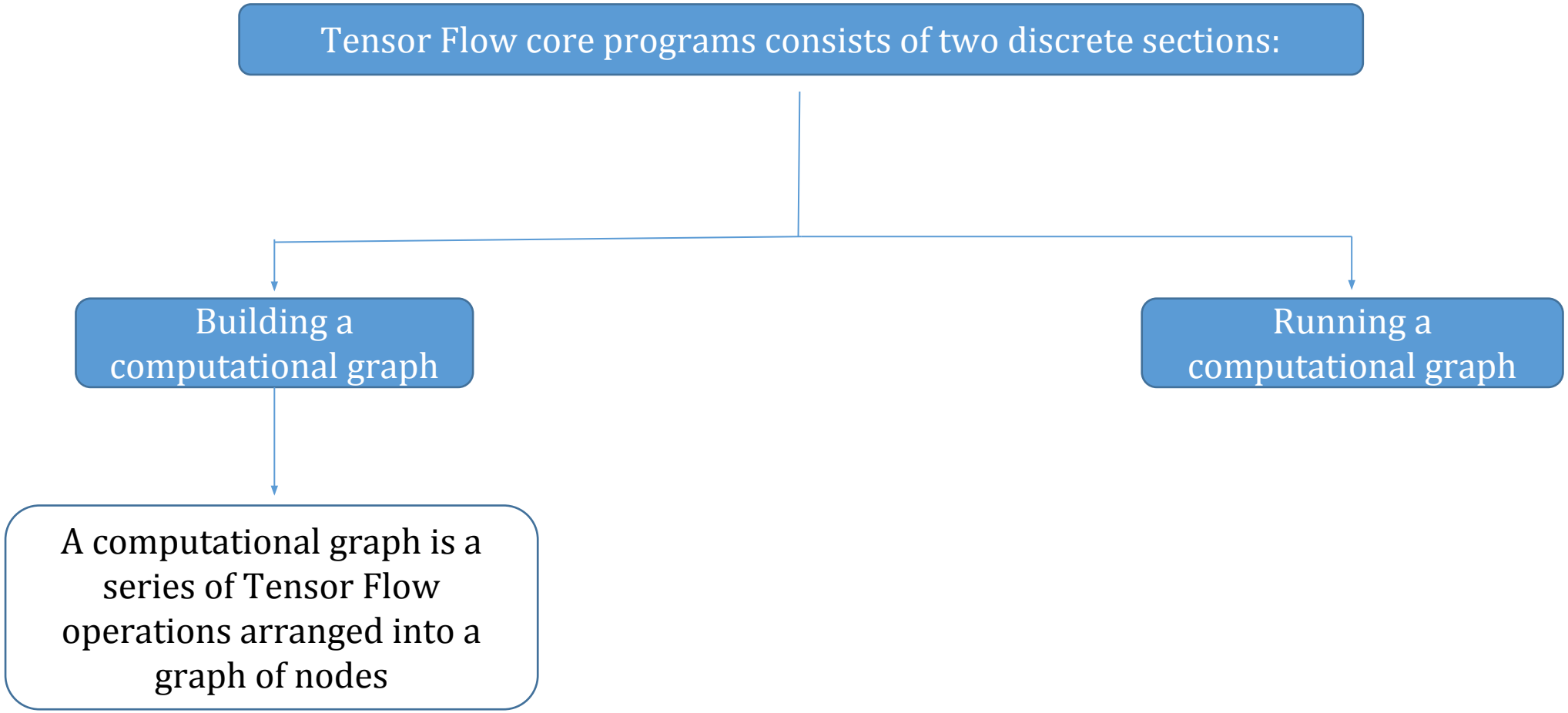
What is Tensor Flow?

- Tensor Flow is a python library used to implement deep networks.
- In Tensor Flow, computation is approached as a dataflow graph..

3.2	-1.4	5.1	…
-1.0	-2	-2.4	…
…	…	…	…
…	…	…	…



Tensor Flow Code-Basics



Tensor Flow Building And Running A Graph

Building a computational graph

```
Import tensorflow as tf

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

Constant nodes

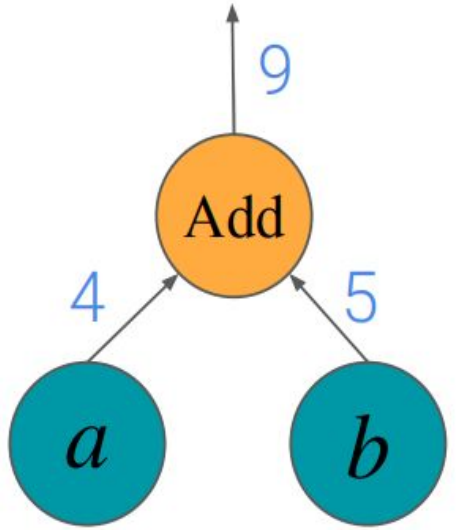
Running a computational graph

```
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.

Basic Code Structure

- View functions as computational graphs
- **First** build a computational graph, and Nodes are operators (ops), variables, and constants
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix ○ Etc.
- **TensorFlow = Tensor + Flow = Data + Flow**
- **Second**, use a session to execute operations in the grap



Basic Code Structure - Graphs

- **Constants** are fixed value tensors - not trainable
- **Variables** are tensors initialized in a session – trainable
- **Placeholders** are tensors of values that are unknown during the graph construction, but passed as input during a session
- **Ops** are functions on tensors

Constants

Constant

Placeholders

Variable

One type of node is a constant. It takes no inputs and it outputs a value it stores internally

```
Import tensorflow as tf

node1=tf.constant(3.0,tf.float32)
node=2.tfconstant(4.0)
print(node1,node2)
```

Constant nodes

Placeholder

A graph can be parameterized to accept external input, known as placeholders. A placeholder is a promise to provide a value later

Constant

Placeholders

Variable

Constant nodes

```
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]}))
```

Feed values to the placeholders a and b

Variable

To make the model traiable, we need to be able to modify the graph to get new outputs with same input. Variables allow us to add trainable parameters to a graph.

- Constant
- Placeholders
- Variable

Variables

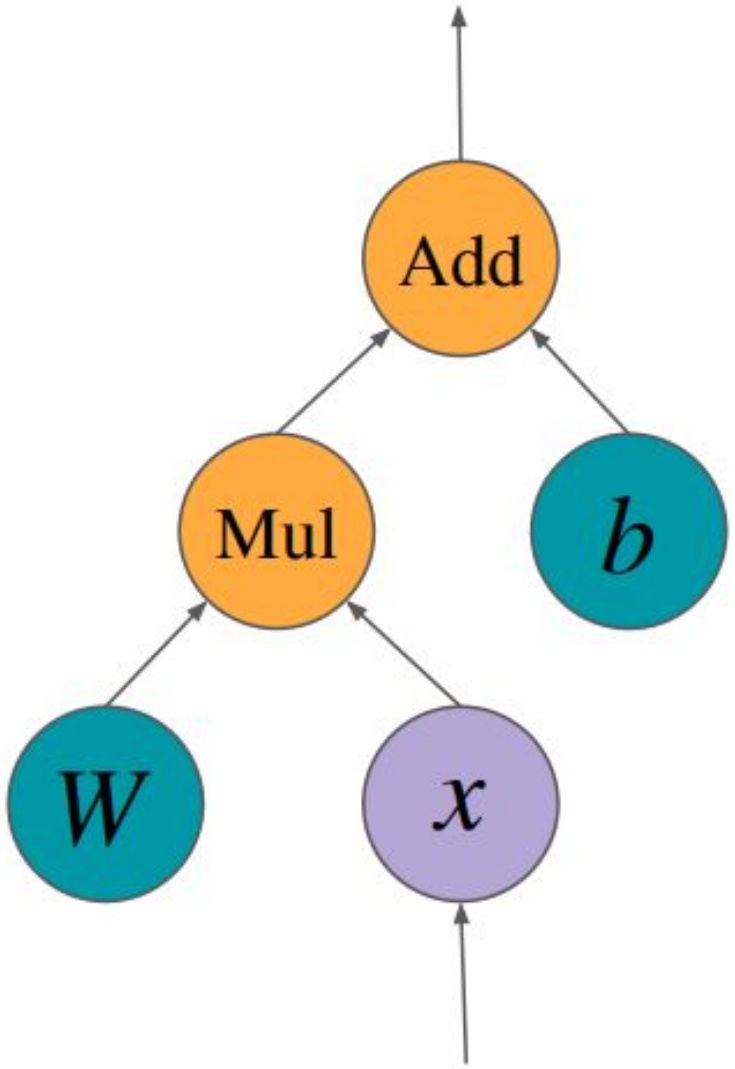
```
Import tensorflow as tf
W = tf.Variable([.3], tf.float32)
B= tf.Variable([-.3], tf.float32)

linear_model=W* x + b
init = tf.global_variables_initializer()
sess = tf.session()
sess.run(init)
Print(sess.run(linear_model,{x[1,2,4]}))
```

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

Basic Code Structure - Graphs

$$\hat{y} = Wx + b$$

variable ops placeholder variable



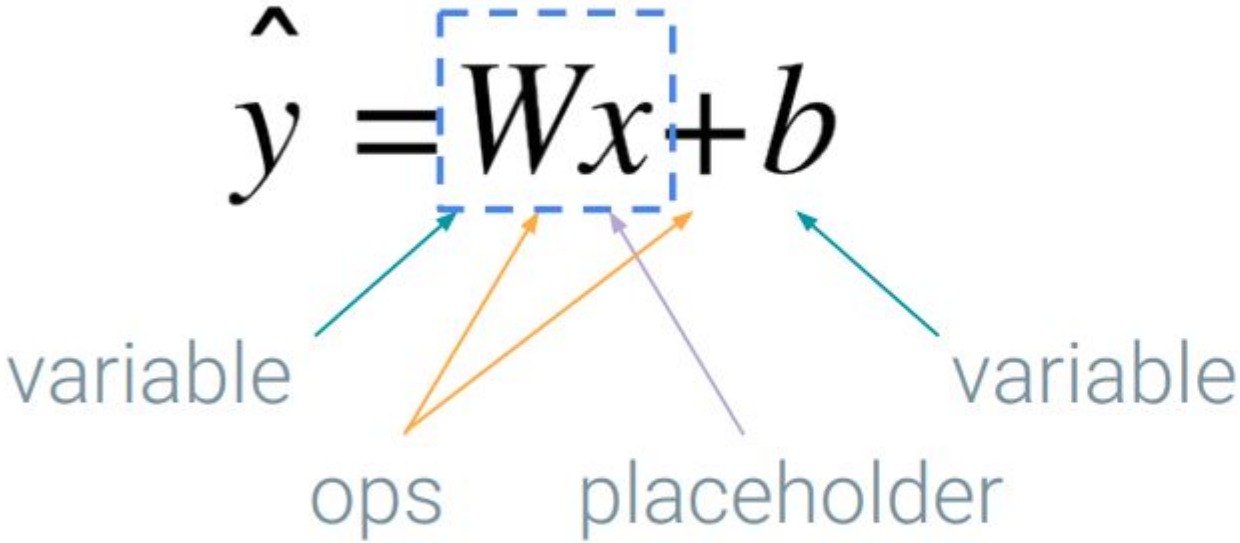
Basic Code Structure - Sessions

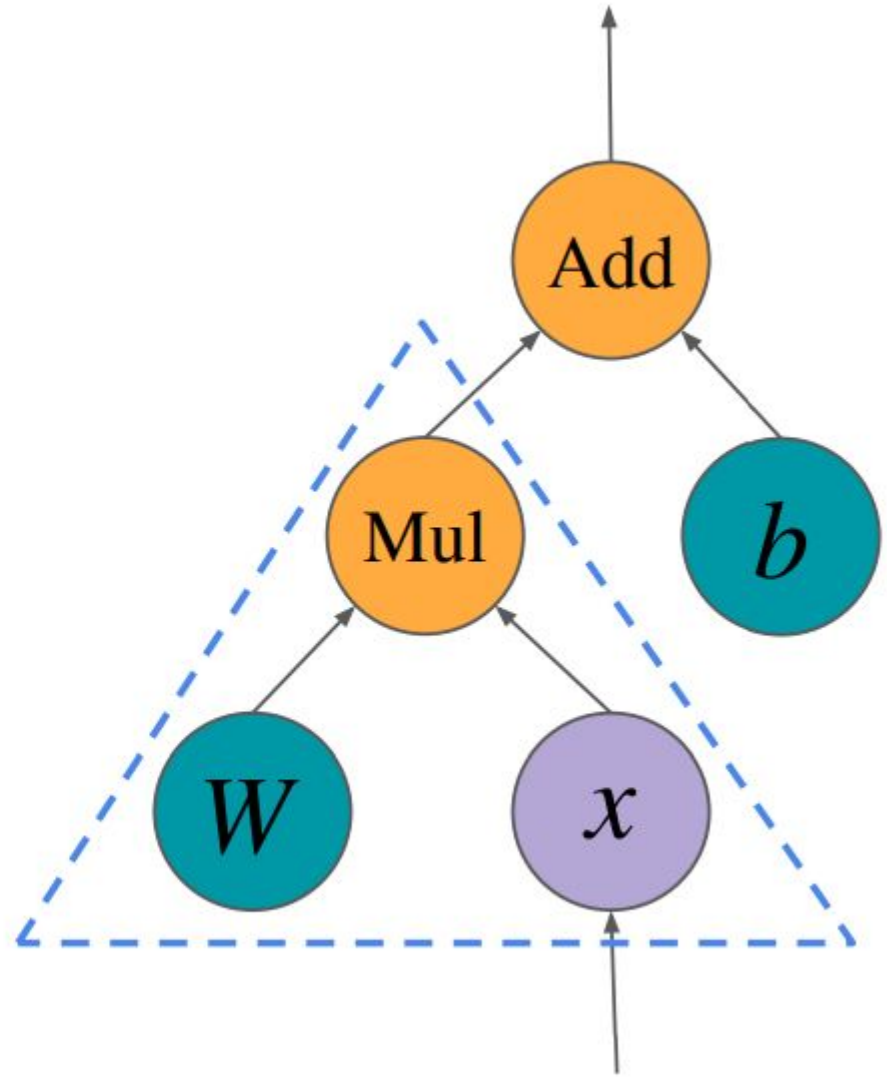
- Session is the runtime environment of a graph, where operations are executed, and tensors are evaluated

```
>>> import tensorflow as tf
>>> a = tf.constant(4)
>>> b = tf.constant(5)
>>> add_op = tf.add(a, b)
>>> print(add_op)
Tensor("Add:0", shape=(), dtype=int32)
```

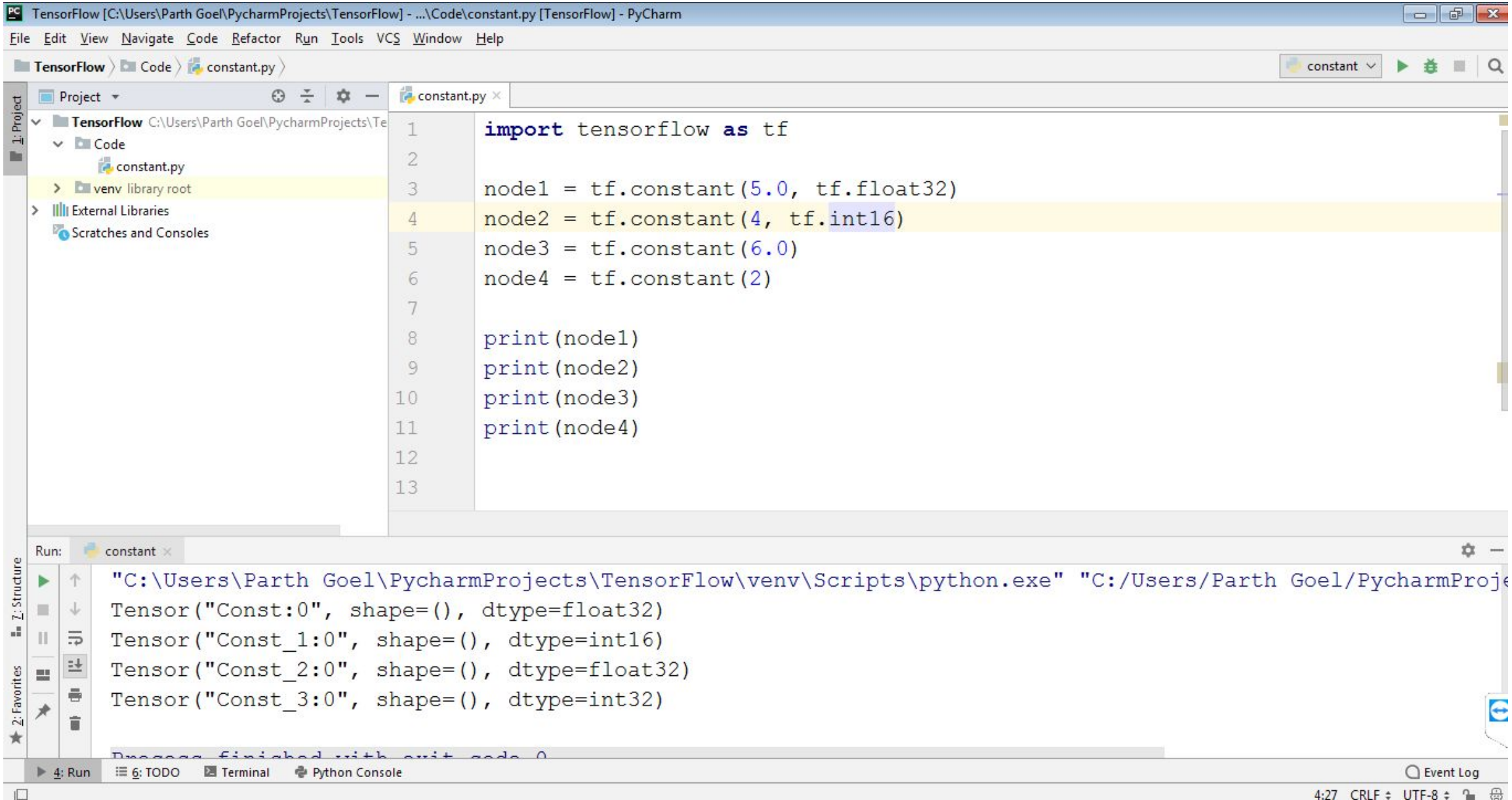
```
>>> import tensorflow as tf
>>> a = tf.constant(4)
>>> b = tf.constant(5)
>>> add_op = tf.add(a, b)
>>> with tf.Session() as session:
...     print(session.run(add_op))
...
9
```

- a.eval() is equivalent to session.run(a), but in general, “eval” is limited to executions of a single op and ops that returns a value
- Upon op execution, only the subgraph required for calculating its value is evaluated

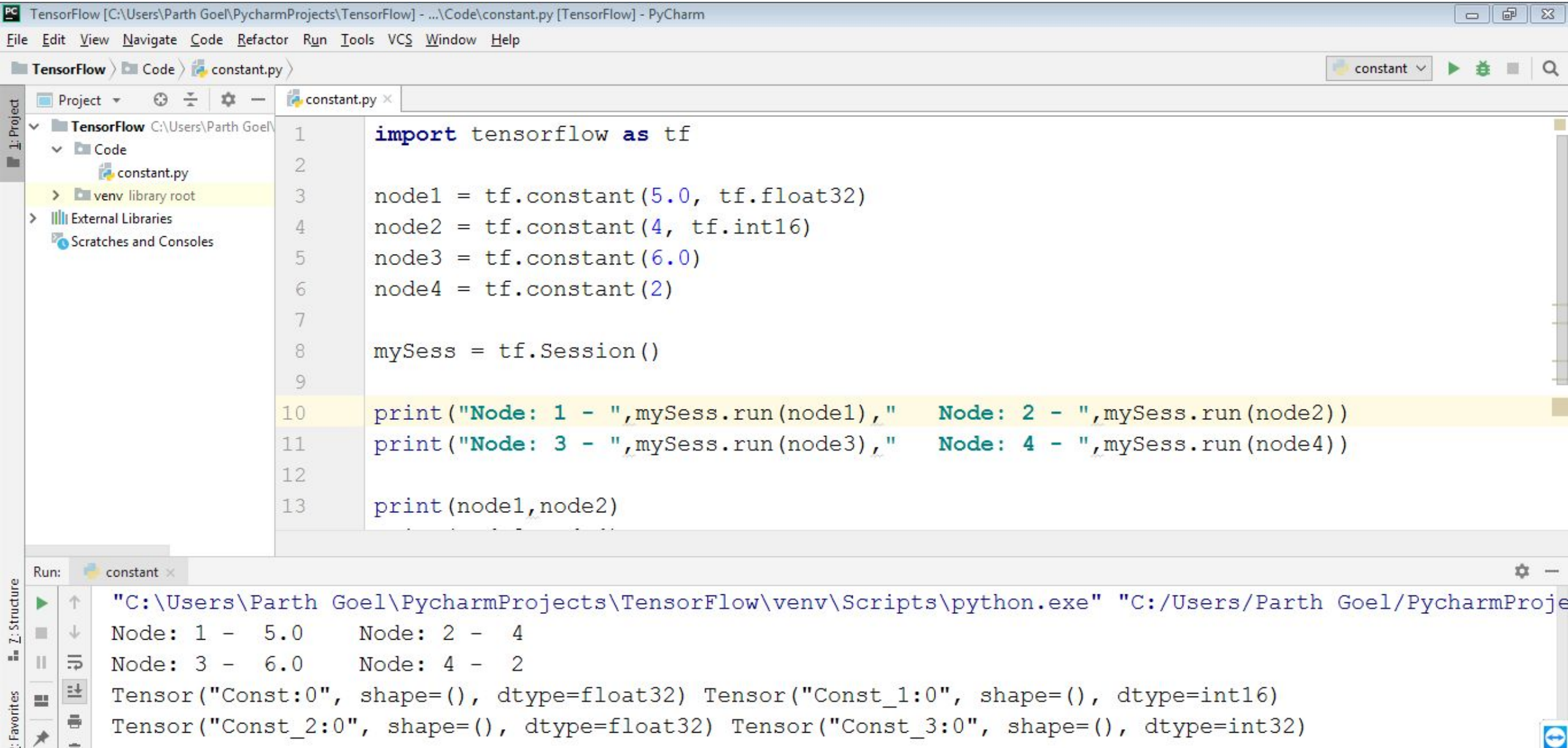

$\hat{y} = Wx + b$
variable
ops
placeholder
variable


Add
Mul
b
W
x

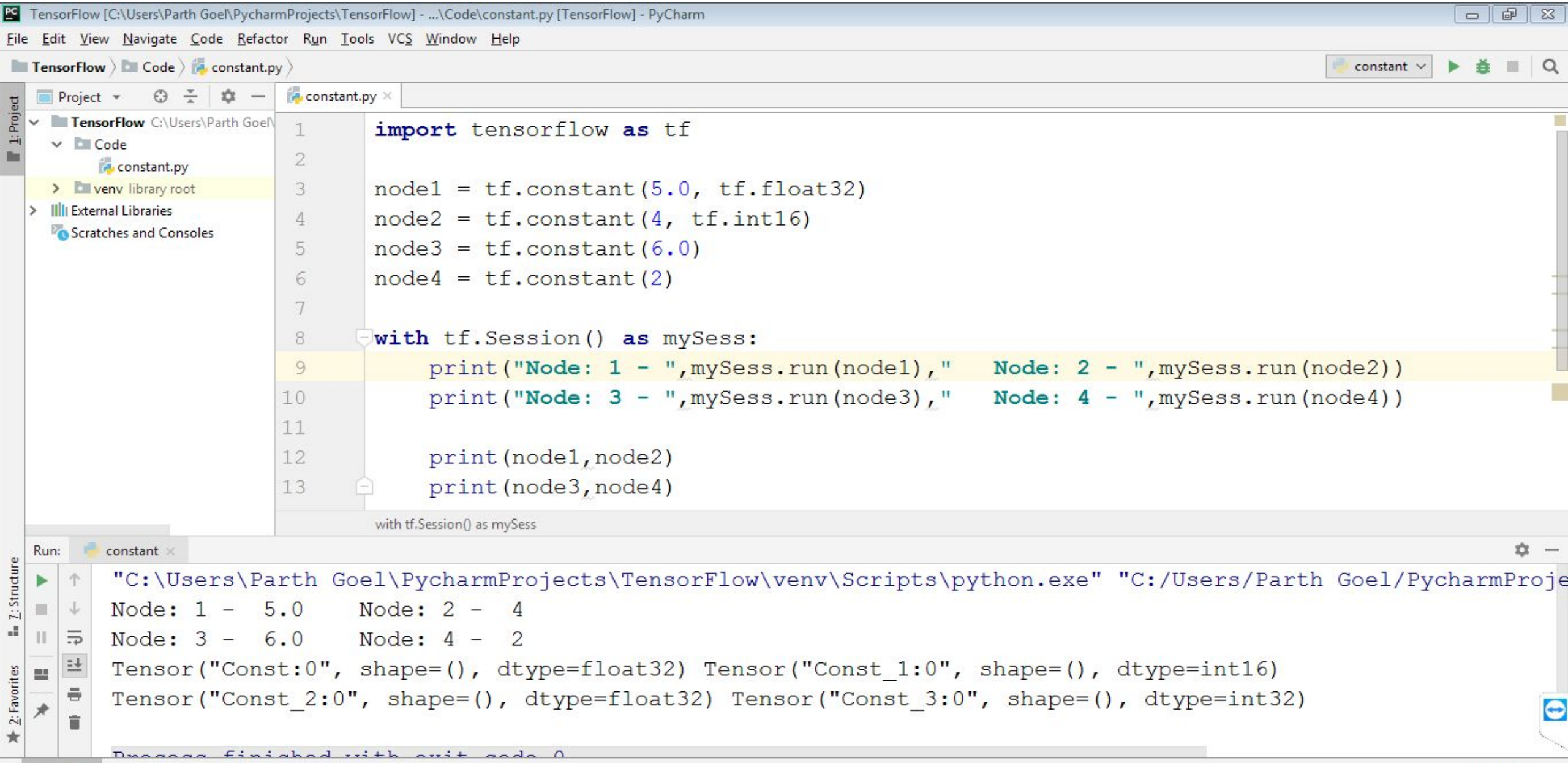
Abstract Tensor code



Method 1 - To Execute The Graph

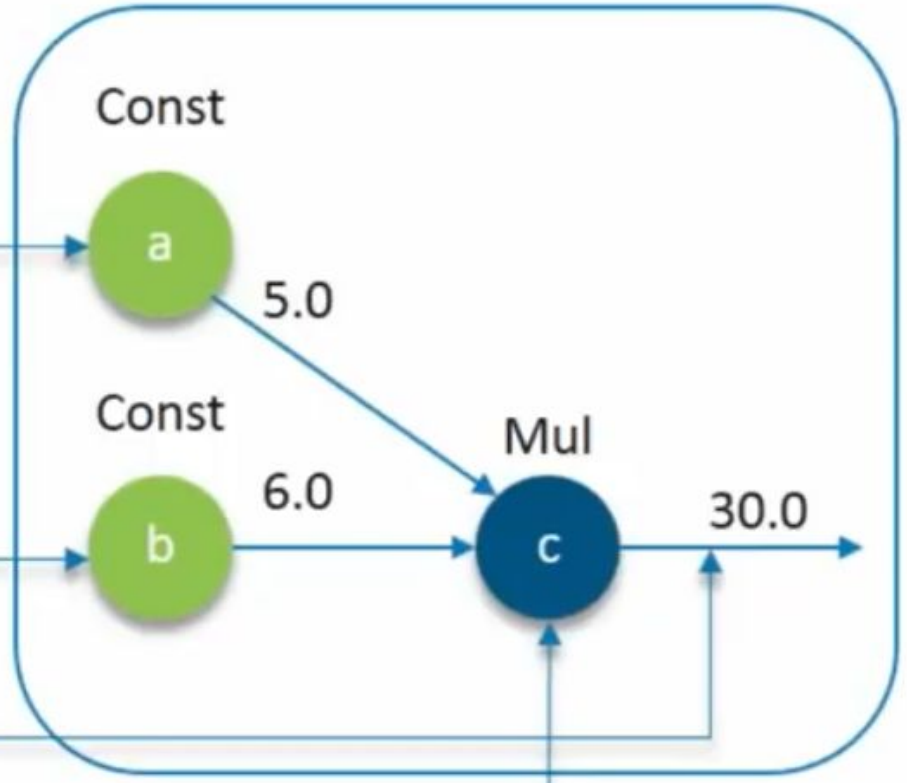


Method 2 - To Execute The Graph

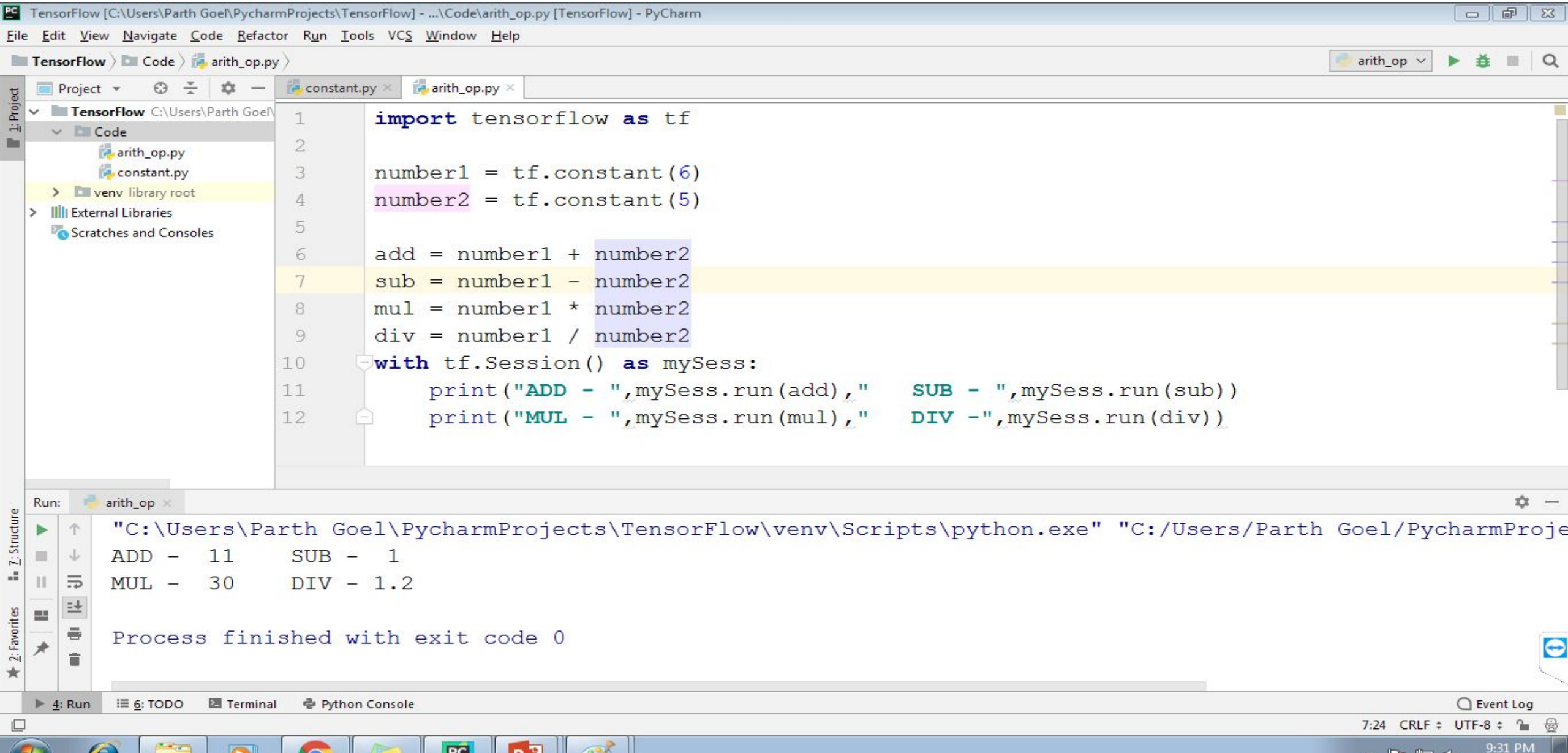


Tensorflow Example

```
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))
```

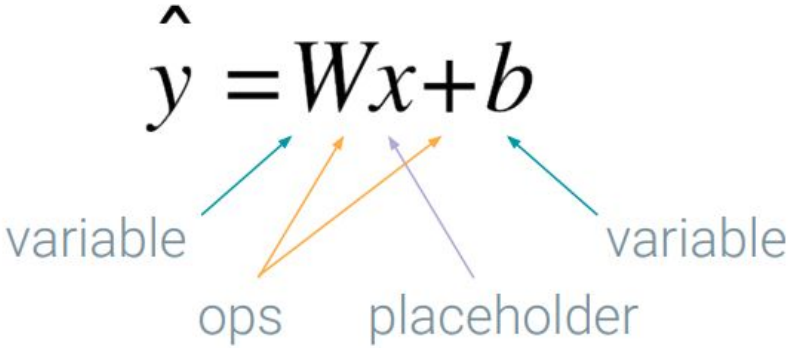


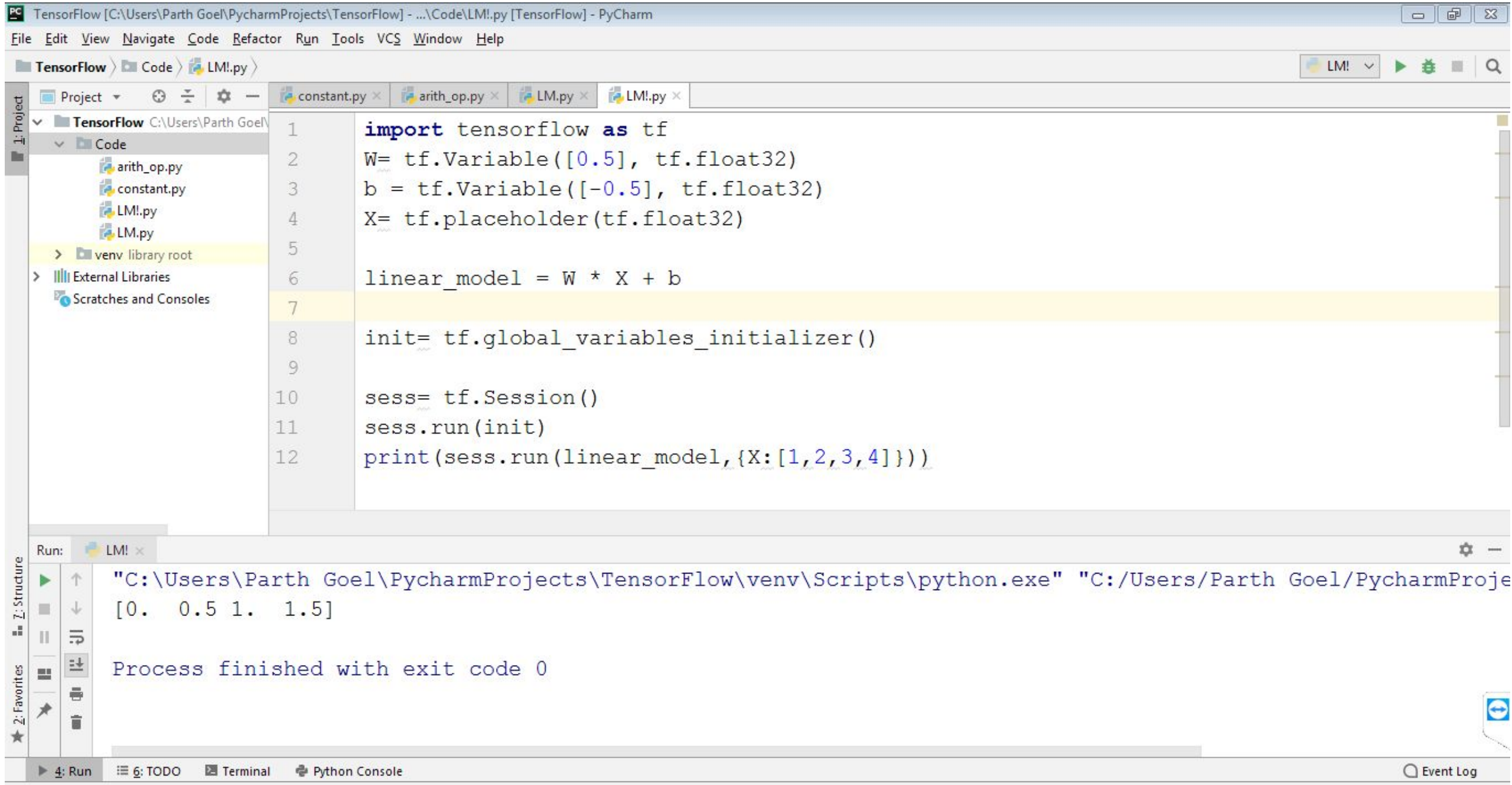
Arithmetic Operations Example



Simple linear Model

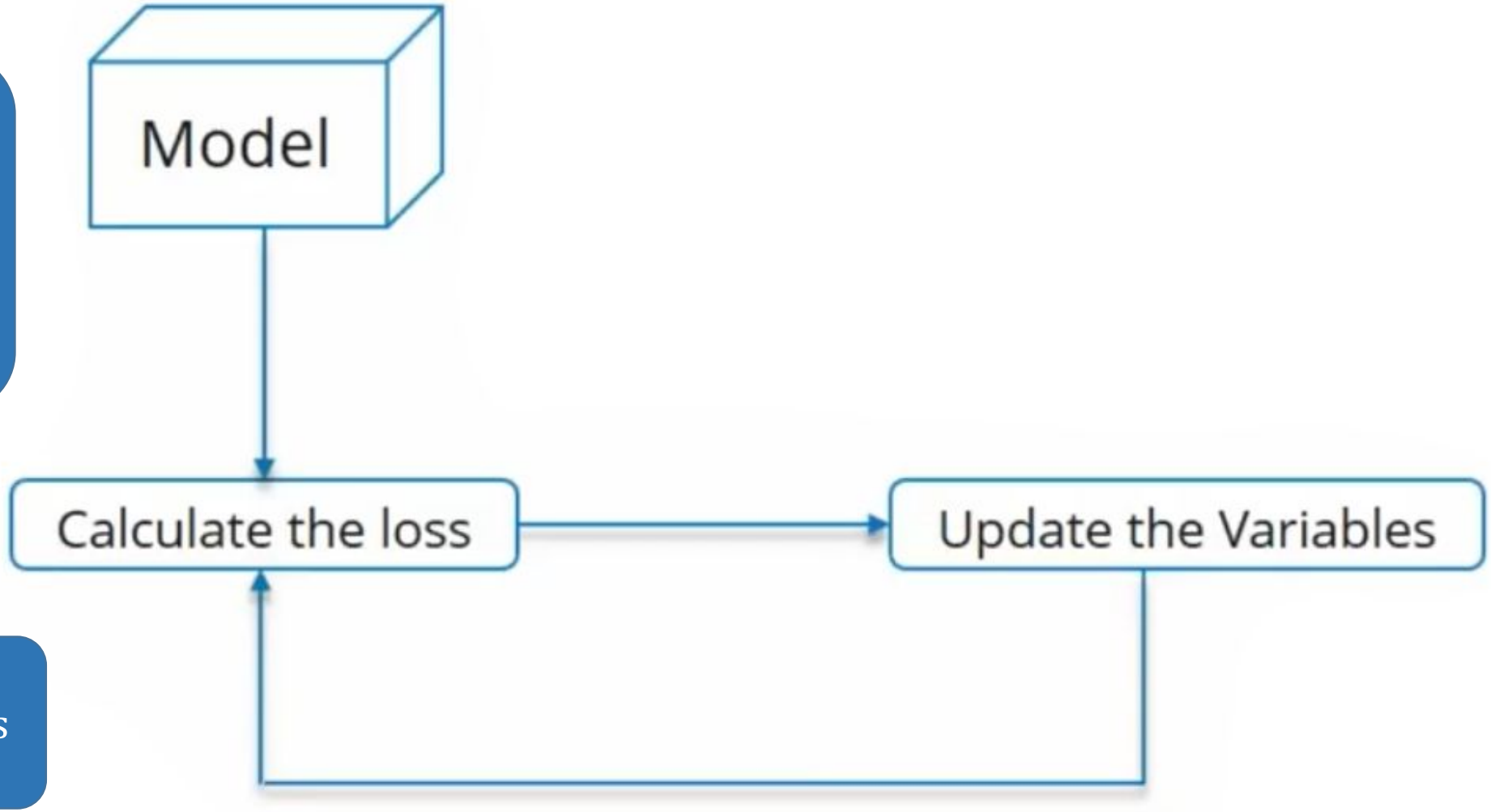
```
import tensorflow as tf
W= tf.Variable([0.5], tf.float32)
b = tf.Variable([-0.5], tf.float32)
X= tf.placeholder(tf.float32)

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]}))
```




TensorFlow [C:\Users\Parth Goel\PycharmProjects\TensorFlow] - ...\Code\LM!.py [TensorFlow] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
TensorFlow Code LM!.py
LM!
Project
TensorFlow C:\Users\Parth Goel\
Code
arith_op.py
constant.py
LM!.py
LM.py
venv library root
External Libraries
Scratches and Consoles
constant.py
arith_op.py
LM.py
LM!.py
1 import tensorflow as tf
2 W= tf.Variable([0.5], tf.float32)
3 b = tf.Variable([-0.5], tf.float32)
4 X= tf.placeholder(tf.float32)
5
6 linear_model = W * X + b
7
8 init= tf.global_variables_initializer()
9
10 sess= tf.Session()
11 sess.run(init)
12 print(sess.run(linear_model,{X:[1,2,3,4]}))
Run: LM!
"C:\Users\Parth Goel\PycharmProjects\TensorFlow\venv\Scripts\python.exe" "C:/Users/Parth Goel/PycharmProje
[0. 0.5 1. 1.5]
Process finished with exit code 0
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Terminal
Python Console
Event Log

How To Increase The efficiency Of The Model?

A loss function measures how far apart the current model is from the provided data.



Repeat the process until the loss becomes very small.

Calculating The Loss

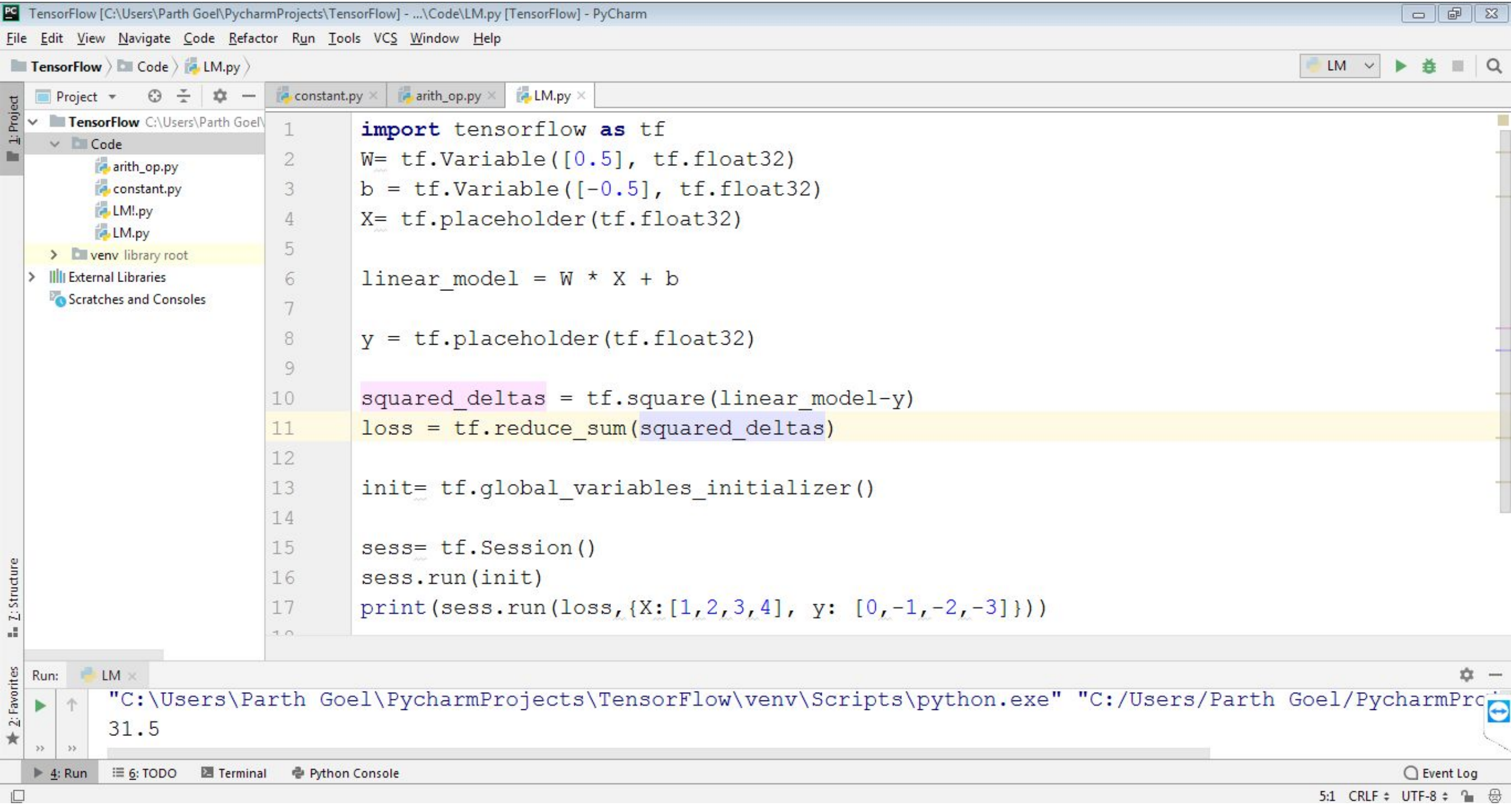
In order to understand how good the model is, we should know the loss/error

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

Squrad_deltas =
tf.square(linear_model -y)
loss=tf.reduce_sum(squared_delta
s)
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

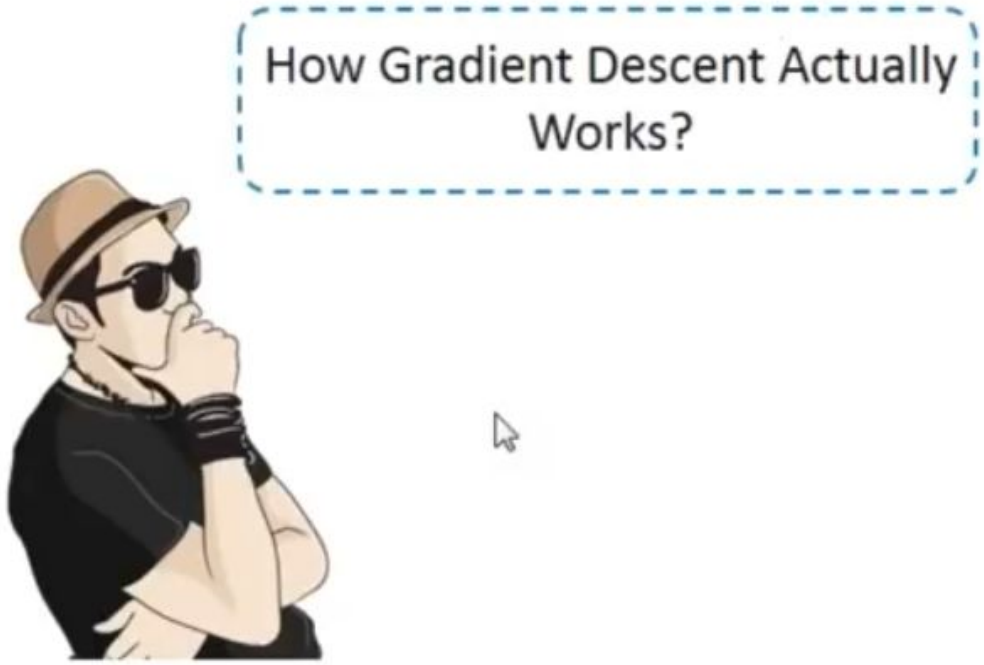
We'll use a standard loss model for linear regression. (linear_model–y) creates a vector where each element is the corresponding example's error delta.

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


TensorFlow [C:\Users\Parth Goel\PycharmProjects\TensorFlow] - ...\Code\LM.py [TensorFlow] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
TensorFlow Code LM.py
LM
Project
constant.py
arith_op.py
LM.py
TensorFlow C:\Users\Parth Goel\
Code
arith_op.py
constant.py
LM!.py
LM.py
venv library root
External Libraries
Scratches and Consoles
1 import tensorflow as tf
2 W= tf.Variable([0.5], tf.float32)
3 b = tf.Variable([-0.5], tf.float32)
4 X= tf.placeholder(tf.float32)
5
6 linear_model = W * X + b
7
8 y = tf.placeholder(tf.float32)
9
10 squared_deltas = tf.square(linear_model-y)
11 loss = tf.reduce_sum(squared_deltas)
12
13 init= tf.global_variables_initializer()
14
15 sess= tf.Session()
16 sess.run(init)
17 print(sess.run(loss,{X:[1,2,3,4], y: [0,-1,-2,-3]}))
Run: LM
"C:\Users\Parth Goel\PycharmProjects\TensorFlow\venv\Scripts\python.exe" "C:/Users/Parth Goel/PycharmPro
31.5
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Terminal
Python Console
Event Log
5:1 CRLF UTF-8

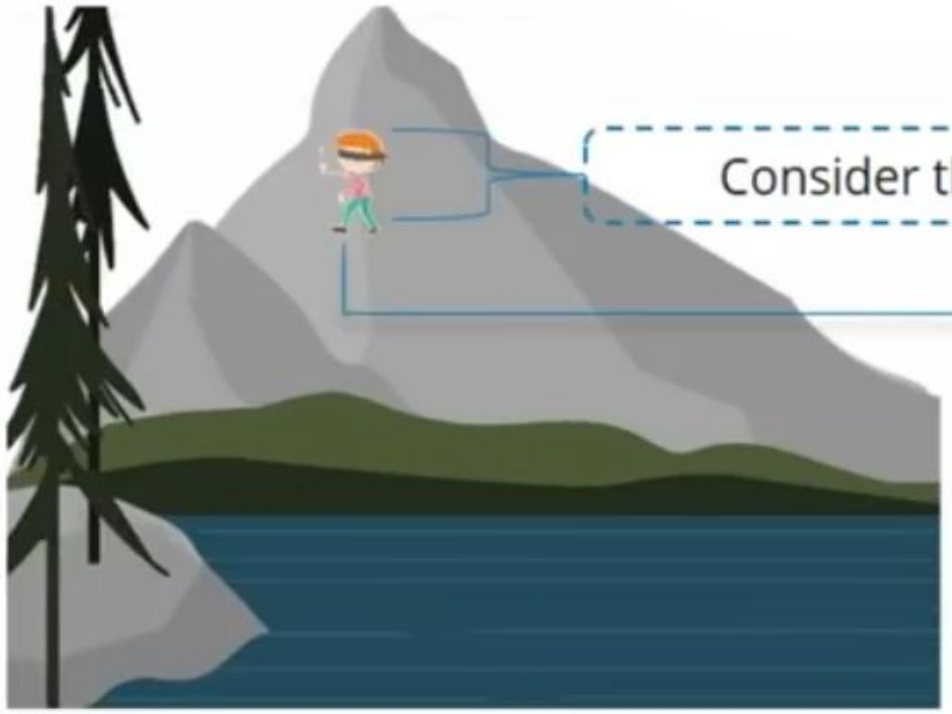
Reducing The Loss

Optimizer modifies each variable according to the magnitude of the derivative of a loss with respect to that variable. Here we will use Gradient descent optimizer.



Reducing The Loss

- Suppose you are at the top of a mountain and you have to reach a lake which is at the lowest point of the mountain(aka valley).
- A twist is that you are blindfolded and you have zero visibility to see where you are headed. So, what approach will take to reach the lake?



Consider the length of the step as learning rate

Consider the position of the hiker as weight

Consider the process of climbing down the mountain as cost function/loss function

Gradient Descent

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

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

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, $\Delta \mathbf{w}$ is a vector that contains the weight updates of each weight coefficient w, which are computed as follows:

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = -\eta \sum_i (\text{target}^{(i)} - \text{output}^{(i)})(-x_j^{(i)}) = \eta \sum_i (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)}.$$

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, Consider the example below:

```
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]))
```

TensorFlow [C:\Users\Parth Goel\PycharmProjects\TensorFlow] - ...\Code\LM_GD.py [TensorFlow] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

TensorFlow > Code > LM_GD.py

LM_GD

Project

TensorFlow C:\Users\Parth Goel\

Code

arith_op.py

constant.py

LM!.py

LM.py

LM_GD.py

venv library root

External Libraries

Scratches and Consoles

constant.py | arith_op.py | LM.py | LM_GD.py

```
1  import tensorflow as tf
2  W= tf.Variable([0.5], tf.float32)
3  b = tf.Variable([-0.5], tf.float32)
4  X= tf.placeholder(tf.float32)
5
6  linear_model = W * X + b
7  y = tf.placeholder(tf.float32)
8
9  squared_deltas = tf.square(linear_model-y) #Loss
10 loss = tf.reduce_sum(squared_deltas)
11
12 optimizer = tf.train.GradientDescentOptimizer(0.01) #Train
13 train = optimizer.minimize(loss)
14
15 init= tf.global_variables_initializer()
16 sess= tf.Session()
17 sess.run(init)
18
19 for i in  range(1000):
20       sess.run(train,{X:[1,2,3,4], y:[0,-1,-2,-3]})
21 print(sess.run([W,b]))
```

Run: LM_GD

```
"C:\Users\Parth Goel\PycharmProjects\TensorFlow\venv\Scripts\python.exe" "C:/Users/Parth Goel/PycharmProjects/TensorFlow/Code
[array([-0.9999964], dtype=float32), array([0.9999894], dtype=float32)]
```

4: Run | 6: TODO | Terminal | Python Console | Event Log

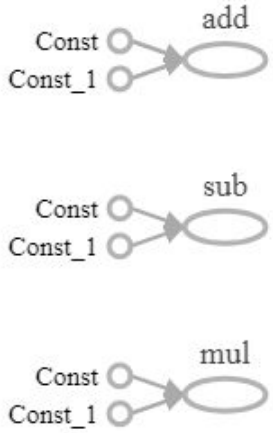
1: Project | 7: Structure | 2: Favorites

Graph Visualization

- For visualizing Tensorflow graphs, we use Tensor Board.
- The first argument when creating the file Writer is an output **directory name**, which will be created if it doesn't exist.

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

truediv




```
tensoboard –logdir = "path_to_the_graph"
```

TensorBoard runs as a local wed app, on port 6006.
(this is default port.)

How To Run Tensorboard

- In PyCharm
  - Go to VIEW -> Tool Windows -> Terminal
  - >>>python venv\Lib\site-packages\tensorboard\main.py --logdir=<dir. name>

  (Make sure that python3.5 path is set in environment variable.)

Inbox (6) - parthgoel.ce@charus | TensorBoard | Downloads

localhost:6006/#graphs&run=.

TensorBoard

GRAPHS

INACTIVE

Search nodes. Regexes sup...

Fit to screen

Download PNG

Run (1)

Session runs (0)

Upload Choose File

Trace inputs

Close legend.

Graph (* = expandable)

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

add

Operation: Add

Attributes (1)

T {"type":"DT_INT32"}

Inputs (2)

Const scalar

Const_1 scalar

Outputs (0)

Add to main graph

Inbox (6) - parthgoel.ce@charus | TensorBoard | Downloads

localhost:6006/#graphs&run=.

TensorBoard GRAPHS INACTIVE

Search nodes. Regexes sup...

Fit to screen

Download PNG

Run (1)

Session runs (0)

Upload Choose File

Trace inputs

Close legend.

Graph (* = expandable)

Namespace* ?

OpNode ?

Unconnected series* ?

Connected series* ?

Constant ?

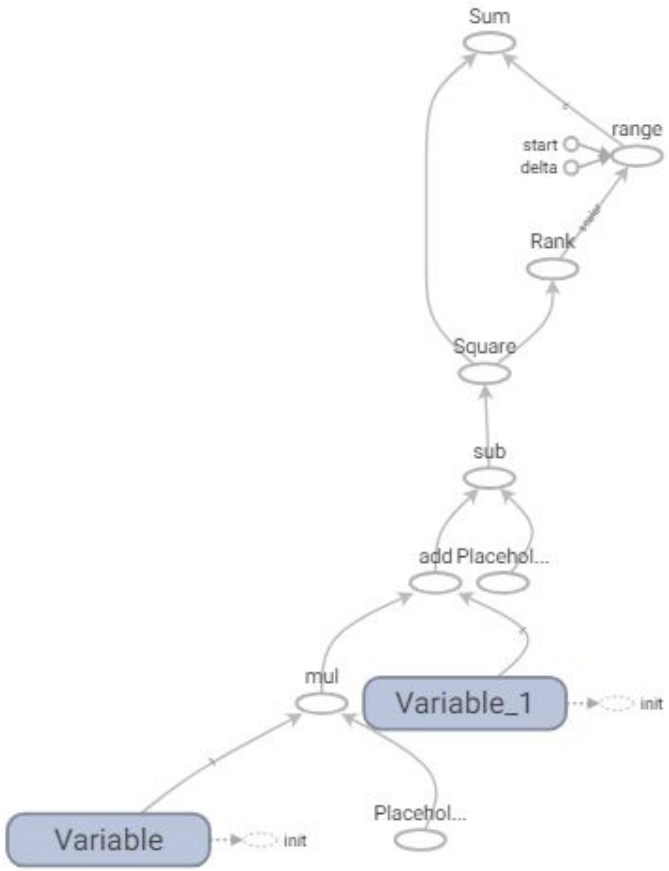
Summary ?

Dataflow edge ?

Control dependency edge ?

Reference edge ?

Main Graph



Auxiliary Nodes


```
import tensorflow as tf
W= tf.Variable([0.5], tf.float32)
b = tf.Variable([-0.5], tf.float32)
X= tf.placeholder(tf.float32)

linear_model = W * X + b
init= tf.global_variables_initializer()

sess= tf.Session()
sess.run(init)
File_Writer =
tf.summary.FileWriter('graph_LM',sess.graph)

print(sess.run(linear_model,{X:[1,2,3,4]}))
```

TensorFlow Cheat Sheet

About

TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. TensorFlow was originally developed for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Skflow

Scikit Flow provides a set of high level model classes that you can use to easily integrate with your existing Scikit-learn pipeline code. Scikit Flow is a simplified interface for TensorFlow, to get people started on predictive analytics and data mining. Scikit Flow has been merged into TensorFlow since version 0.8 and now called TensorFlow Learn.

Keras

Keras is a minimalist, highly modular neural networks library, written in Python and capable of running on top of either TensorFlow or Theano

Installation

How to install new package in Python:

```
pip install <package-name>
```

Example: `pip install requests`

How to install tensorflow?

device = **cpu/gpu**

python_version = **cp27/cp34**

```
sudo pip install
https://storage.googleapis.com/
tensorflow/linux/$device/tensorflow-
0.8.0-$python_version-none-linux_x86
_64.whl
```

How to install Skflow

```
pip install sklearn
```

How to install Keras

```
pip install keras
```

update ~/.keras/keras.json - replace “theano” by “tensorflow”

Helpers

Python helper

Important functions

`type(object)`

Get object type

`help(object)`

Get help for object (list of available methods, attributes, signatures and so on)

`dir(object)`

Get list of object attributes (fields, functions)

`str(object)`

Transform an object to string

`object?`

Shows documentations about the object

`globals()`

Return the dictionary containing the current scope's global variables.

`locals()`

Update and return a dictionary containing the current scope's local variables.

`id(object)`

Return the identity of an object. This is guaranteed to be unique among simultaneously existing objects.

```
import __builtin__
dir(__builtin__)
```

Other built-in functions

TensorFlow

Main classes

```
tf.Graph()
tf.Operation()
tf.Tensor()
tf.Session()
```

Some useful functions

```
tf.get_default_session()
tf.get_default_graph()
tf.reset_default_graph()
ops.reset_default_graph()
tf.device(“/cpu:0”)
tf.name_scope(value)
tf.convert_to_tensor(value)
```

TensorFlow Optimizers

```
GradientDescentOptimizer
AdadeltaOptimizer
AdagradOptimizer
MomentumOptimizer
AdamOptimizer
FtrlOptimizer
RMSPropOptimizer
```

Reduction

`reduce_sum`
`reduce_prod`
`reduce_min`
`reduce_max`
`reduce_mean`
`reduce_all`
`reduce_any`
`accumulate_n`

Activation functions

`tf.nn?`

relu
relu6
elu
softplus
softsign
dropout
bias_add
sigmoid
tanh
sigmoid_cross_entropy_with_logits
softmax
log_softmax
softmax_cross_entropy_with_logits
sparse_softmax_cross_entropy_with_logits
weighted_cross_entropy_with_logits
etc.

Skflow

Main classes

TensorFlowClassifier
TensorFlowRegressor
TensorFlowDNNClassifier
TensorFlowDNNRegressor
TensorFlowLinearClassifier
TensorFlowLinearRegressor
TensorFlowRNNClassifier
TensorFlowRNNRegressor

TensorFlowEstimator

Each classifier and regressor have following fields
n_classes=0 (Regressor), n_classes are expected to be input (Classifiers)
batch_size=32,
steps=200, // except TensorFlowRNNClassifier - there is 50
optimizer='Adagrad',
learning_rate=0.1,

The Machine Learning Life Cycle

1. **Define Project Objectives**
   - ☐ Specify business problem
   - ☐ Acquire subject matter expertise
   - ☐ Define unit of analysis and prediction target
   - ☐ Prioritize modeling criteria
   - ☐ Consider risks and success criteria
   - ☐ Decide whether to continue
2. **Acquire & Explore Data**
   - ☐ Find appropriate data
   - ☐ Merge data into single table
   - ☐ Conduct exploratory data analysis
   - ☐ Find and remove any target leakage
   - ☐ Feature engineering
3. **Model Data**
   - ☐ Variable selection
   - ☐ Build candidate models
   - ☐ Model validation and selection
4. **Interpret & Communicate**
   - ☐ Interpret model
   - ☐ Communicate model insights
5. **Implement, Document & Maintain**
   - ☐ Set up batch or API prediction system
   - ☐ Document modeling process for reproducibility
   - ☐ Create model monitoring and maintenance plan

Image Source: https://www.datarobot.com/wiki/machine-learning-life-cycle/

Parameter and Hyperparameter

Parameter - Weight and Bias

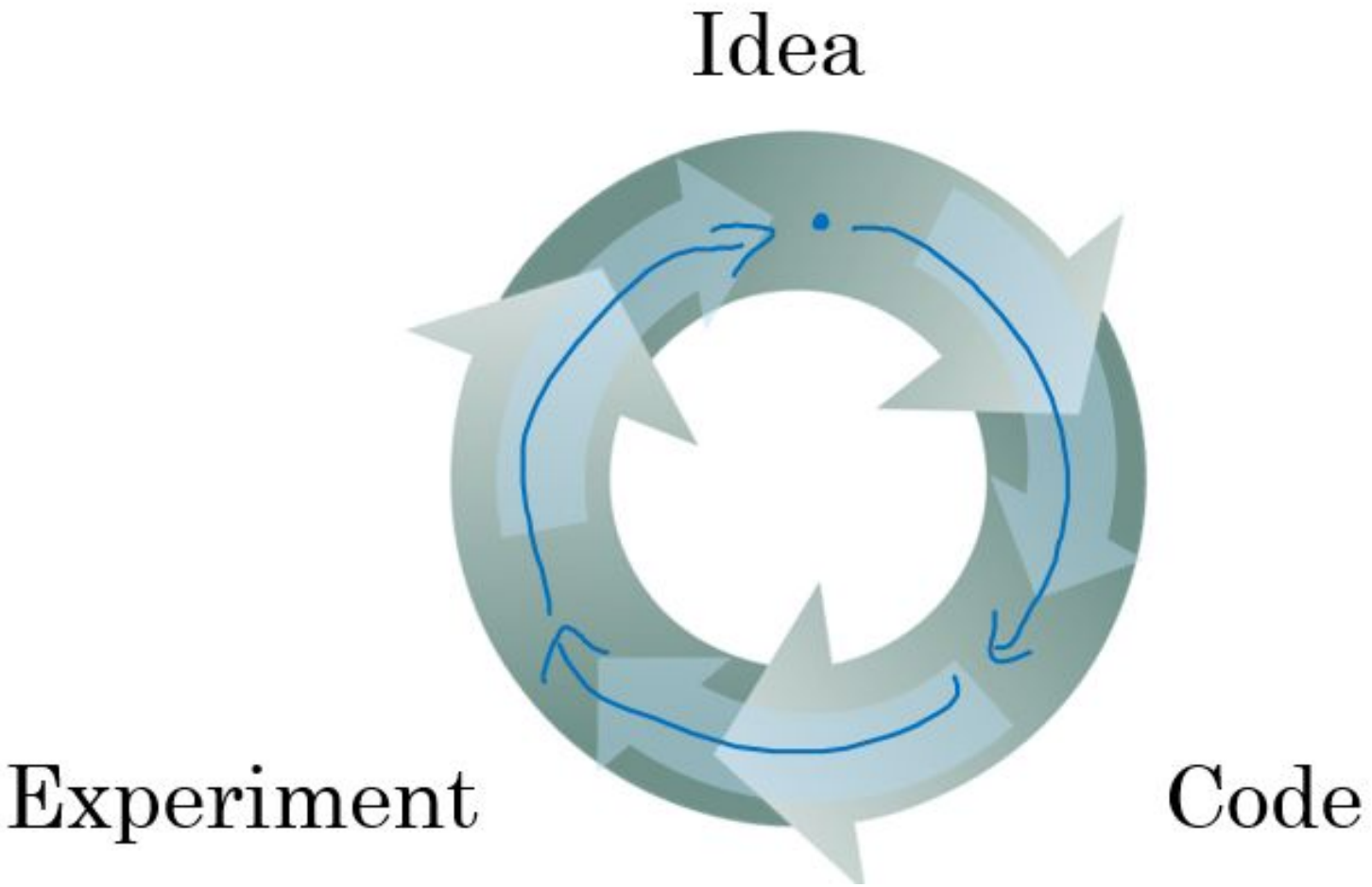
Hyperparameter (parameters that control the algorithm):

1) No of hidden layers - Try and error
2) No of neurons in each hidden layer - Try and error
3) Learning rate. (The most important parameter) – 0.1, 0.01, 0.001, 0.0001, .00001
4) Activation functions – Sigmoid, Relu, Leaky Relu, Tanh
5) Number of iteration – Try and error
6) Epoch – Try and error
7) Batch size – 4, 8, 16, 32, 64, 128… (Power of 2)
8) Regularization – Dropout, L1, L2
9) Regularization Rate and Dropout lembda
10) Normalizing inputs – Min-max, mean, Z-Score
11) Weights and Bias Initialization – Zero, Random, He
12) optimization algorithm – SGD, ADAM, RMSProp
13) Learning rate decay
14) momentum – 0.9

Iterations, Epoch and Batch Size

- **batch size** = the number of training examples in one forward/backward pass. The higher the batch size, the more memory space you'll need.
- number of **iterations** = number of passes, each pass using [batch size] number of examples. To be clear, one pass = one forward pass + one backward pass (we do not count the forward pass and backward pass as two different passes).
- one **epoch** = one forward pass and one backward pass of *all* the training examples.
- **Example**: if you have 5000 training examples, and your batch size is 32, then it will take 157 iterations to complete 1 epoch.
- **5000/32 = 156.25 =157 iterations (156 x 32 + 1 x 8)**
- **If you set epoch = 20, 20 x 157 = 3140 iterations**

Empirical Process



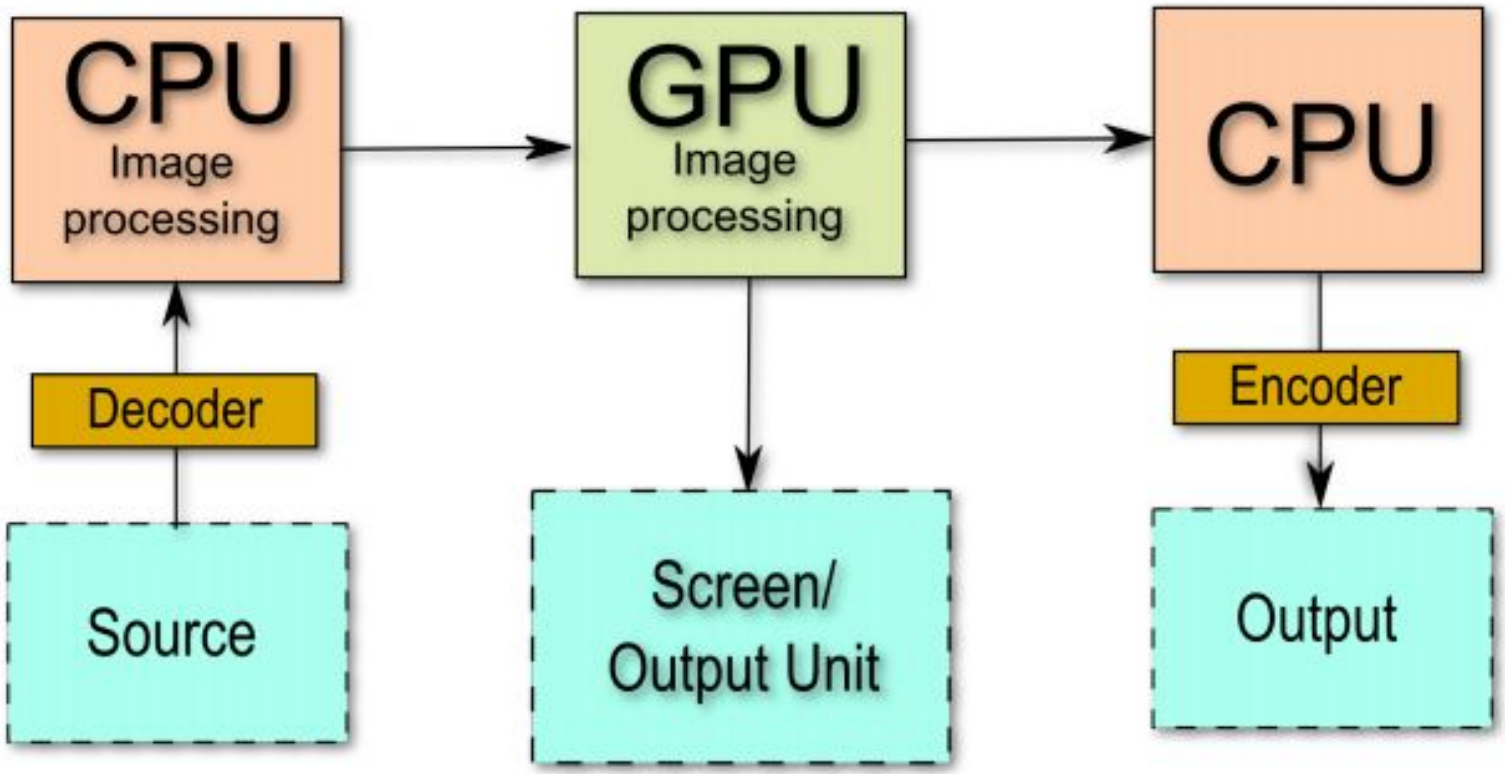
Practical issues with neural networks

- Data augmentation
- Data normalization
- Architecture optimization
- Loss function
- Weight initialization
- Learning rate and its decay schedule
- Overfitting: regularization, early stopping
- Momentum

Research issues in Deep Learning

- Huge amount of data
  - Overfitting in Neural Networks
  - Hyperparameter Optimization
- High-performance Hardware Requirement
- Essentially – It is a Blackbox
- Lack of Flexibility, Scalability and Multitasking
- Generative Models – learn and then generate new images
- Video Processing and Analytics
- Deep Learning in Embedded System
- Generalization of algorithms – (Partial Solution: Transfer Learning)
- CPU to GPU Data Transfer Time

A typical workflow of images/video processing



Deep Learning Library

TensorFlow™ torch DEEPLEARNING4J OpenCV

Keras theano YOLO Microsoft CNTK

mxnet Caffe Chainer scikits learn

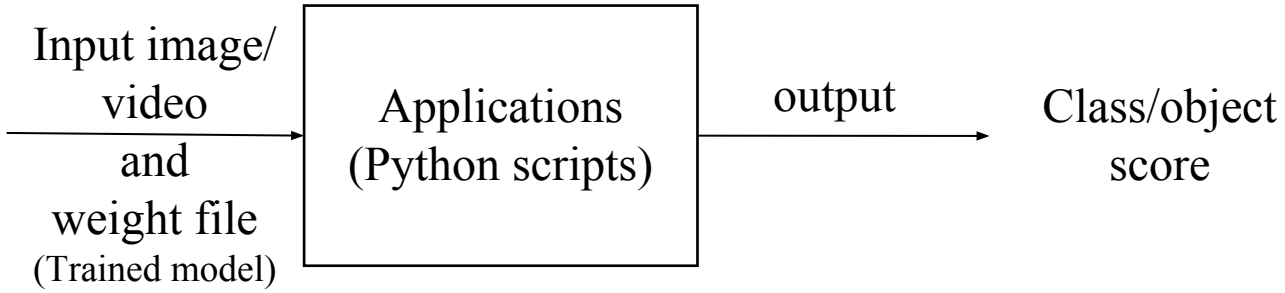
Deep Learning Cloud Services

Microsoft Azure amazon web services™ Google Cloud Platform Google Colab Cirrascale CLOUD SERVICES FLOYD Paperspace

Comparison of Deep Learning Library

Software	Creator	Software license	Open source	Platform	Written in	Interface	CUDA support
TensorFlow	Google Brainteam	Apache 2.0	Yes	Linux, Mac OS, Windows	C++,Python	Python, (C/C++public API only for executing graphs)	Yes
Keras	François Chollet	MIT license	Yes	Linux, Mac OS, Windows	Python	Python	Yes
Deeplearning4j	Adam Gibson	Apache 2.0	Yes	Linux, Mac OS, Windows, Android (Cross-platform)	C, C++	Java, Scala	Yes
Caffe	Yangqing Jia	BSD License	Yes	Linux, Mac OS, Windows	C++	Python, Matlab	Yes
Theano	Université de Montréal	BSD License	Yes	Cross-platform	Python	Python	Yes
Torch	Ronan Collobert,Kavukcuoglu, Clement	BSD License	Yes	Linux, Mac OS, Windows, Android, iOS	C, Lua	Lua, C, utility library for C++/OpenCL	Yes
Microsoft Cognitive Toolkit - CNTK	Microsoft Research	MIT license	Yes	Windows, Linux	C++	Python, C++, Command line, BrainScript(.NET on roadmap)	Yes

Schematic Diagram

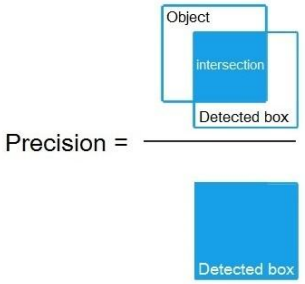


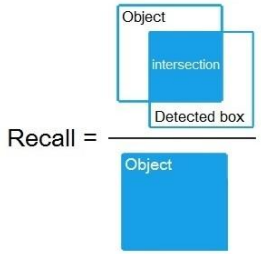
Evaluation Parameters

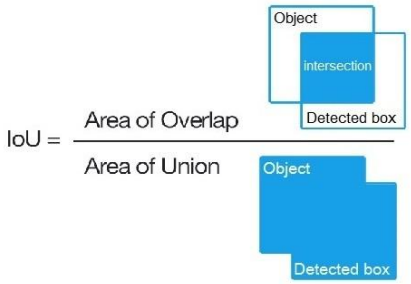
		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

- Accuracy – Confusion matrix
- Precision
- Recall
- F1 Score (try to maximize - to create a balanced classification model)
- mAP – IOU (For Object Detection)
- Loss/Cost – Minimize the difference between actual and predicated
- Entropy

$$recall = \frac{true\ positives}{true\ positives\ + false\ negatives} \qquad precision = \frac{true\ positives}{true\ positives + false\ positives}$$







$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

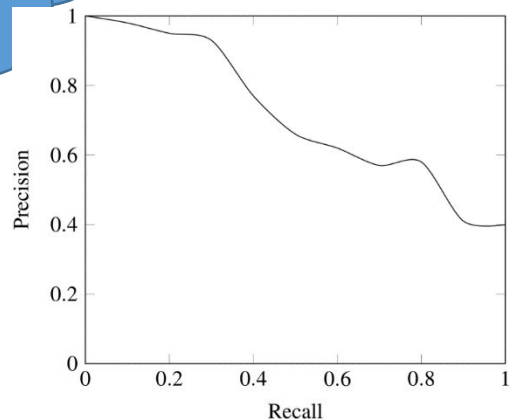
Dear, Do you remember all wedding anniversary surprises from me?

They are celebrating 6th wedding anniversary.

If you can recall all **5** gifts correctly, then, your recall ratio is **1.0** (**100%**). If you can recall 4 events correctly, your recall ratio is **0.8** (**80%**).

After some days, again she asked same question……

Most Dangerous Question!!!



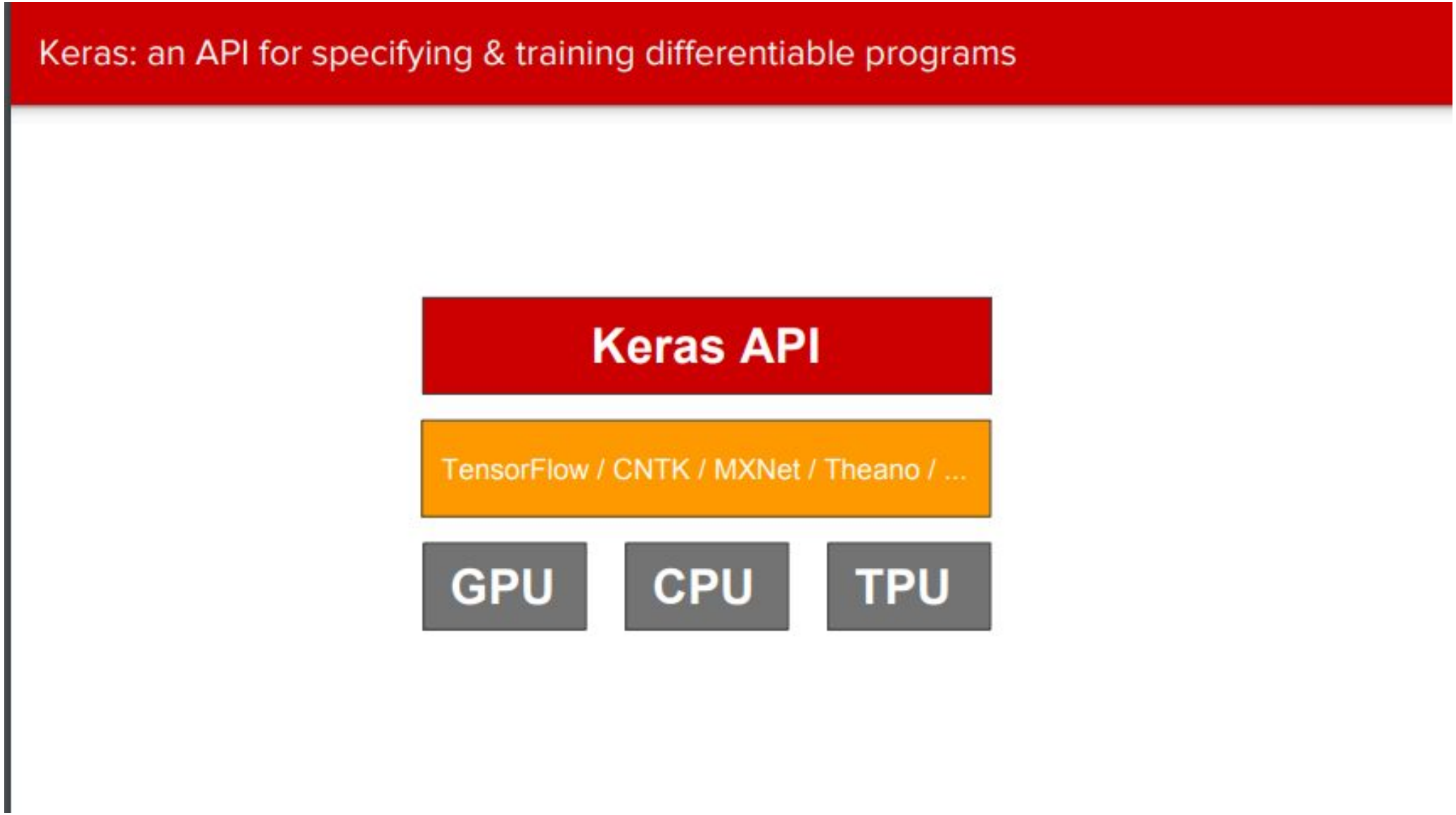
For example, you answers 15 times, 10 times are correct and 5 times are wrong. This means you can recall all events but it's not so precise.

That means: (15 answers: 10 correct answers, 5 wrong answers), you get **100%** recall but your precision is only **66.67%** (10 / 15).

Online DEMO

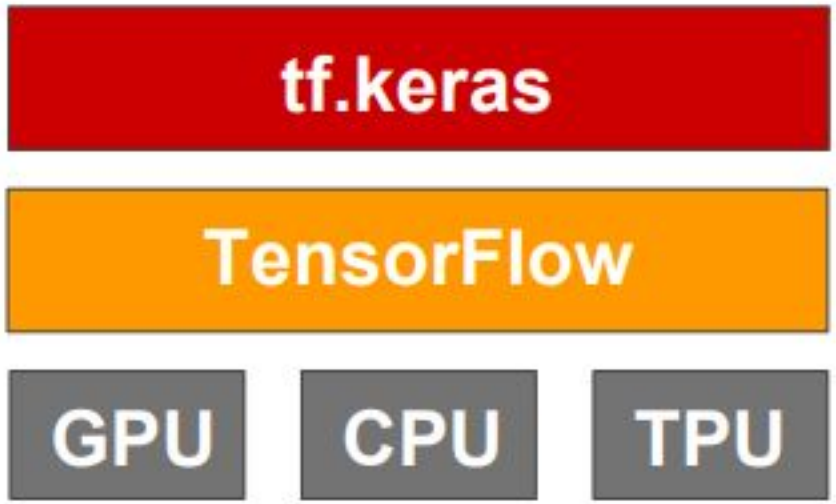
- https://playground.tensorflow.org

What is Keras?



Keras is the official high-level API of TensorFlow

- tensorflow.keras (tf.keras) module
- Part of core TensorFlow since v1.4
- Full Keras API
- Better optimized for TF
- Better integration with TF-specific features
  - Estimator API
  - Eager execution
  - etc.



Who makes Keras? Contributors and backers

633 contributors

What's special about Keras?

- A focus on user experience.
- Large adoption in the industry and research community.
- Multi-backend, multi-platform.
- Easy productization of models.

Keras vs Tensorflow

Parameters	Keras	Tensorflow
Type	High-Level API Wrapper	Low-Level API
Complexity	Easy to use if you Python language	You need to learn the syntax of using some of Tensorflow function
Purpose	Rapid deployment for making model with standard layers	Allows you to make an arbitrary computational graph or model layers
Tools	Uses other API debug tool such as TFDBG	You can use Tensorboard visualization tools
Community	Large active communities	Large active communities and widely shared resources

Case Study: Classification of Iris Dataset

Iris Dataset

Collected by **Ronald Fisher in 1936**

- Available in **UCI machine learning repository** (https://archive.ics.uci.edu/ml/datasets/Iris)
- contains measurements for **150 iris flowers** from three different species. **Each class has 50 observations.**

Iris Versicolor

Iris Setosa

Iris Virginica

Samples
(instances, observations)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

in cm

Features
(attributes, measurements, dimensions)

Class labels
(targets)

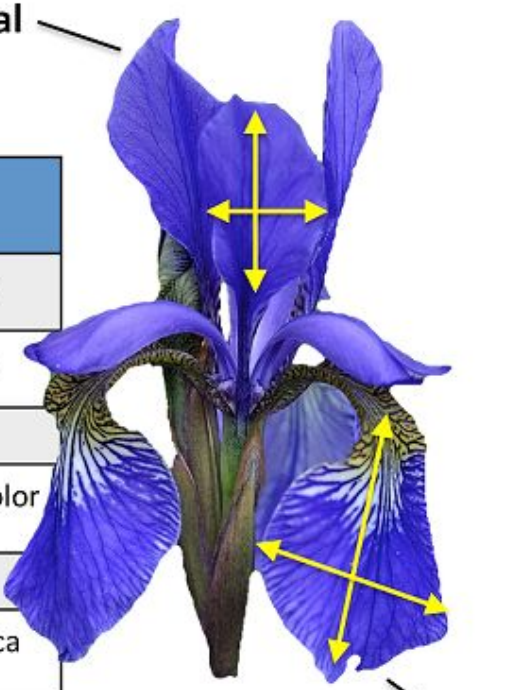
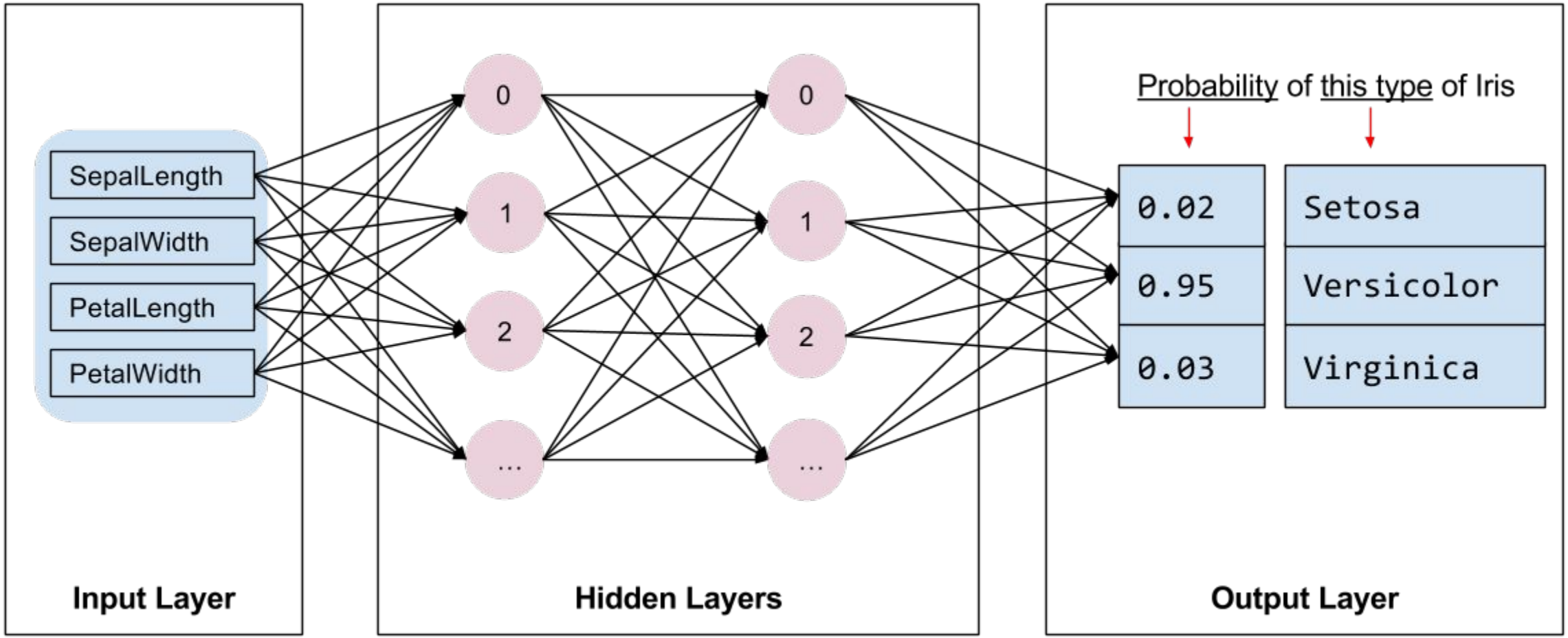


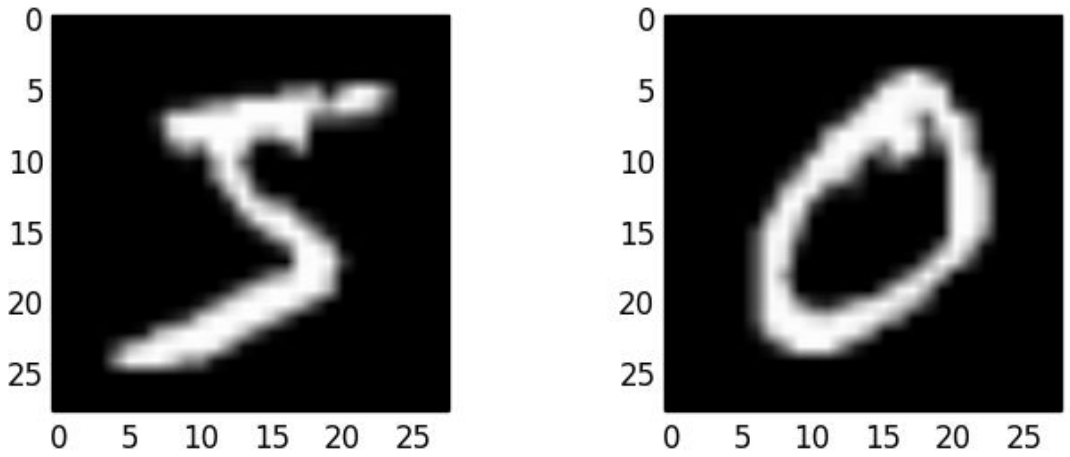
Image Source:

NN Model for Iris classification

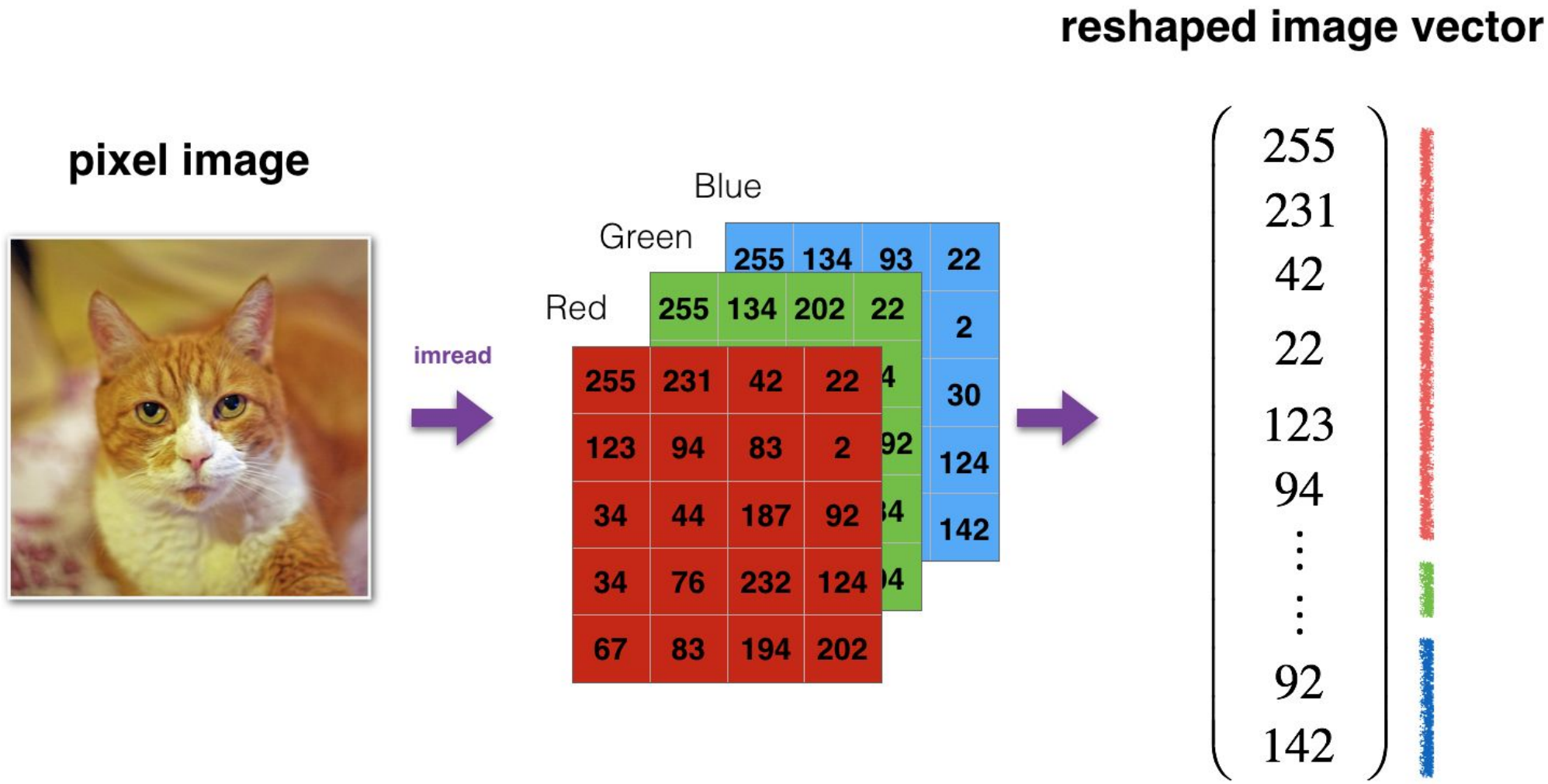


MNIST Dataset

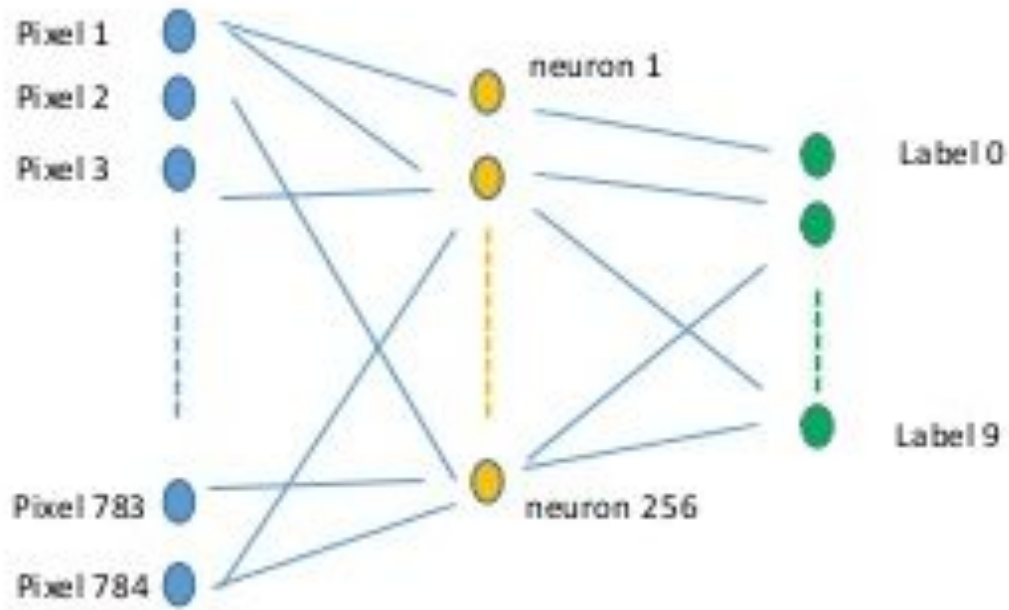
- The MNIST database of **handwritten digits**, available from this page, has a **training set of 60,000** examples, and a **test set of 10,000** examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.
- Each Image Size – 28 x 28
- Color Channel – 1 (Black and White)



Input


pixel image
imread
Blue
Green
Red
255 134 93 22
255 134 202 22
2
255 231 42 22
30
123 94 83 2
124
34 44 187 92
142
34 76 232 124
67 83 194 202
reshaped image vector
255
231
42
22
123
94
92
142

Treat image as a vector. It has length 784 (28by28), the number of pixels. One hidden layer (fully connected)



NVIDIA Titan Xp

(Grant by Nvidia Graphics Pvt. Ltd.)

❖ Configuration

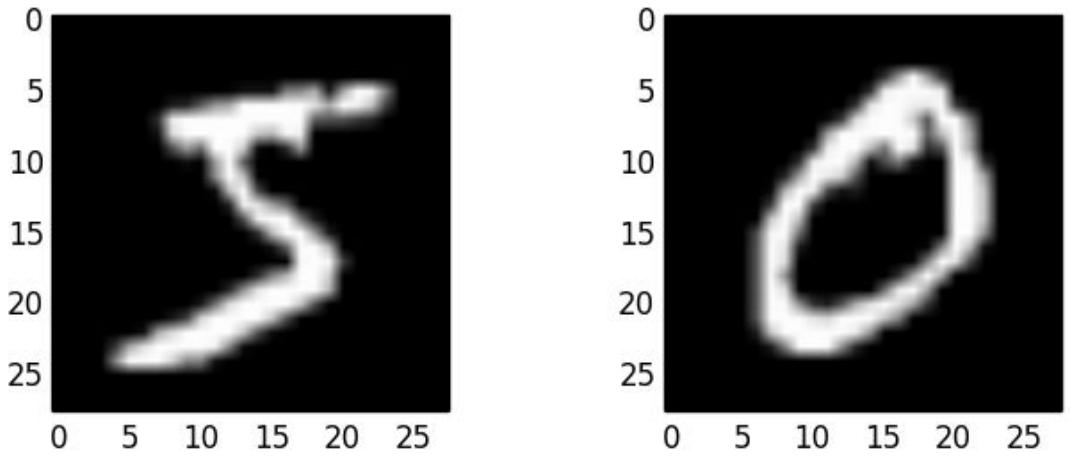
- GPU Architecture – Pascal
- NVIDIA Cuda Cores – 3840
- Memory Size – 12 GB GDDR5X
- Memory Speed - 11.4 Gbps
- Nvidia Cuda Compute Capblity – 6.1
- VR Ready

Work Station (Donated by CHARUSAT)

- **Configuration**
  - 32 GB RAM
  - Core i7 8th Generation CPU
  - 2 TB HDD
  - ASRock Mother Board Z series
  - Cooler Master Cabinet
  - 19’’ FHD LED
  - Logitech Keyboard and Mouse

MNIST Dataset

- The MNIST database of **handwritten digits**, available from this page, has a **training set of 60,000** examples, and a **test set of 10,000** examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.
- Each Image Size – 28 x 28
- Color Channel – 1 (Black and White)



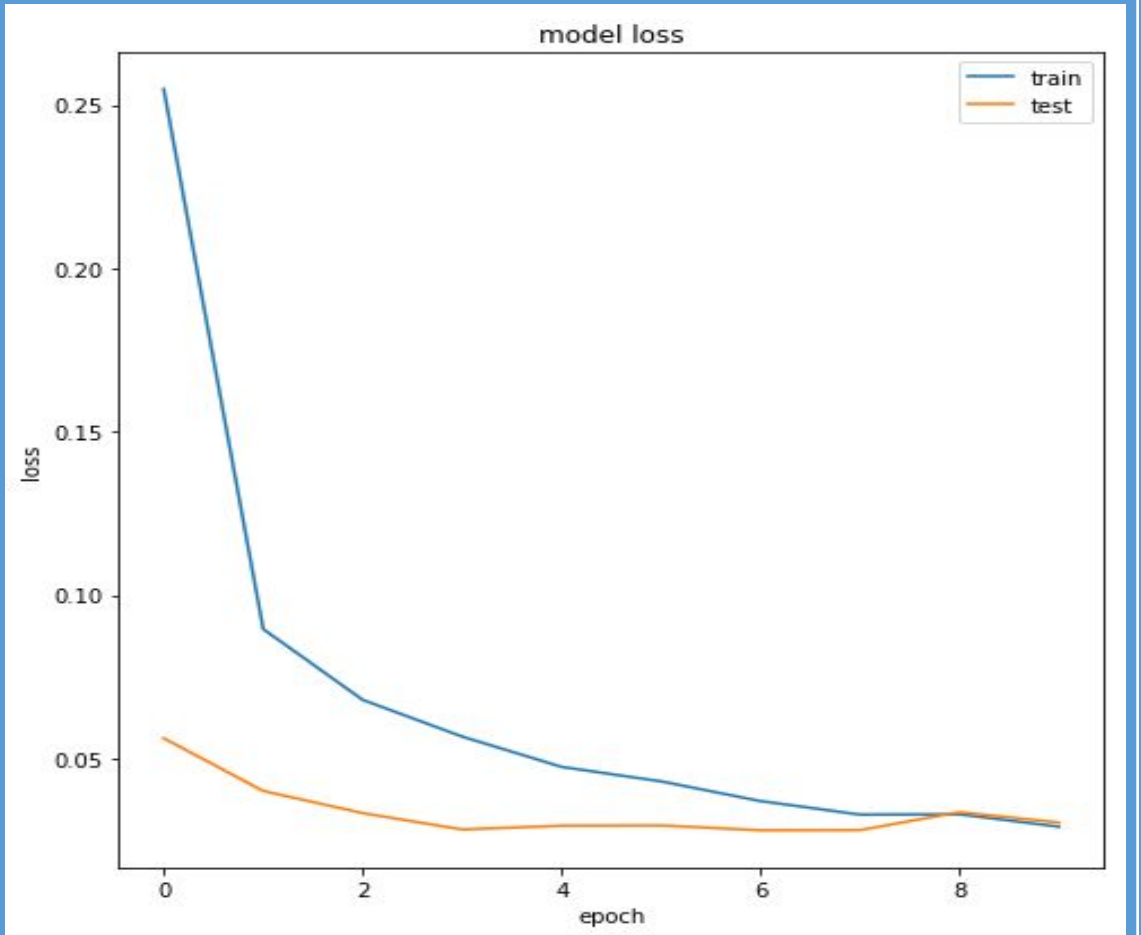
CNN Model

Implementation using Keras and Tensorflow as backend

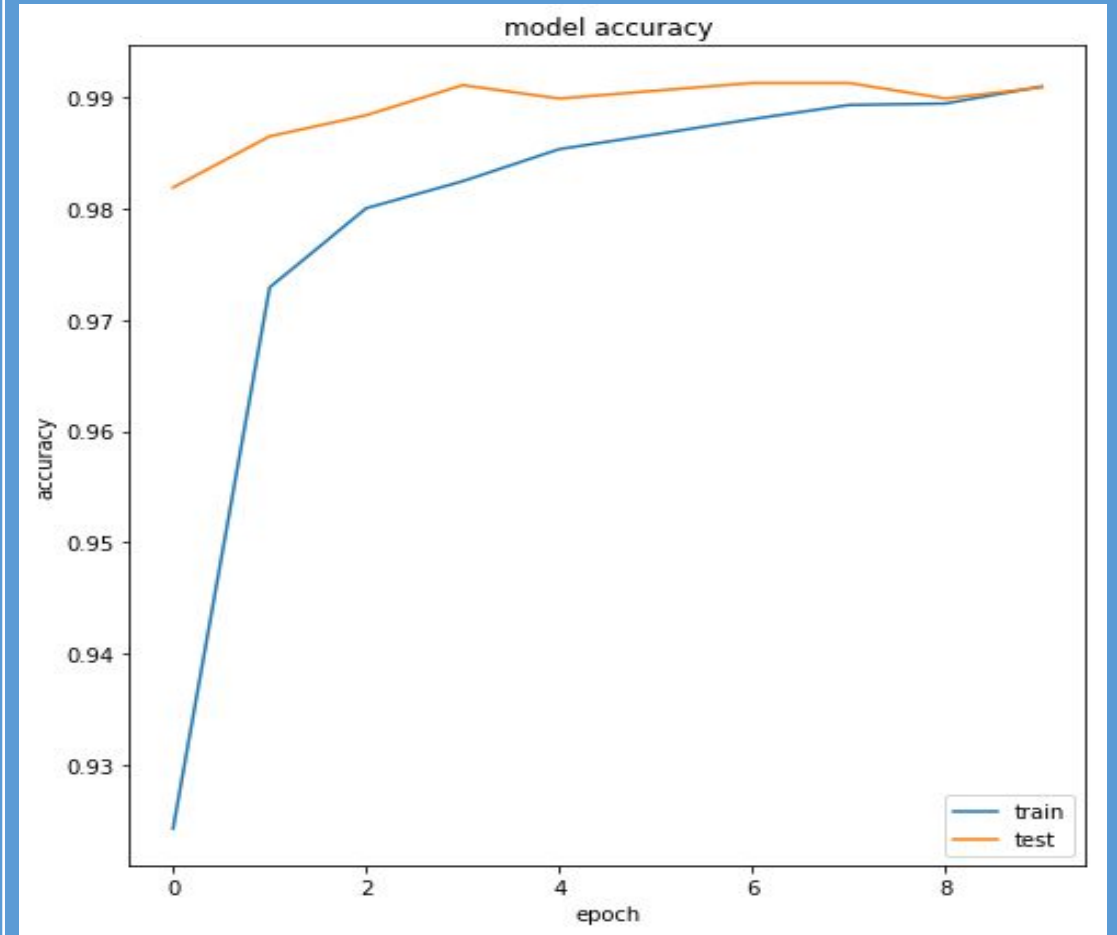
Layer (type)	Output Shape	Param #
conv2d_17 (Conv2D)	(None, 26, 26, 32)	320
conv2d_18 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_9 (MaxPooling)	(None, 12, 12, 64)	0
dropout_17 (Dropout)	(None, 12, 12, 64)	0
flatten_9 (Flatten)	(None, 9216)	0
dense_17 (Dense)	(None, 128)	1179776
dropout_18 (Dropout)	(None, 128)	0
dense_18 (Dense)	(None, 10)	1290

Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0

Results in CPU of CNN Model

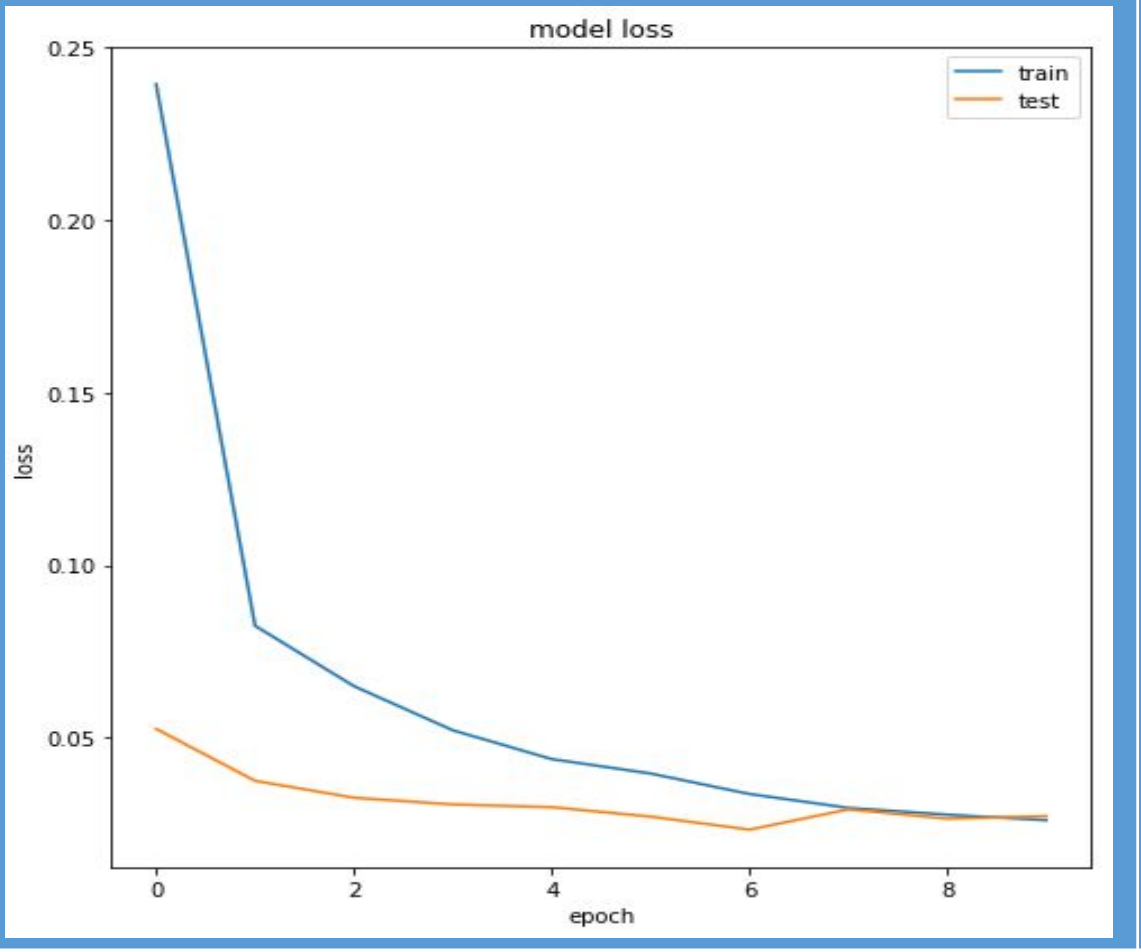


Model Loss vs. Epoch in CPU

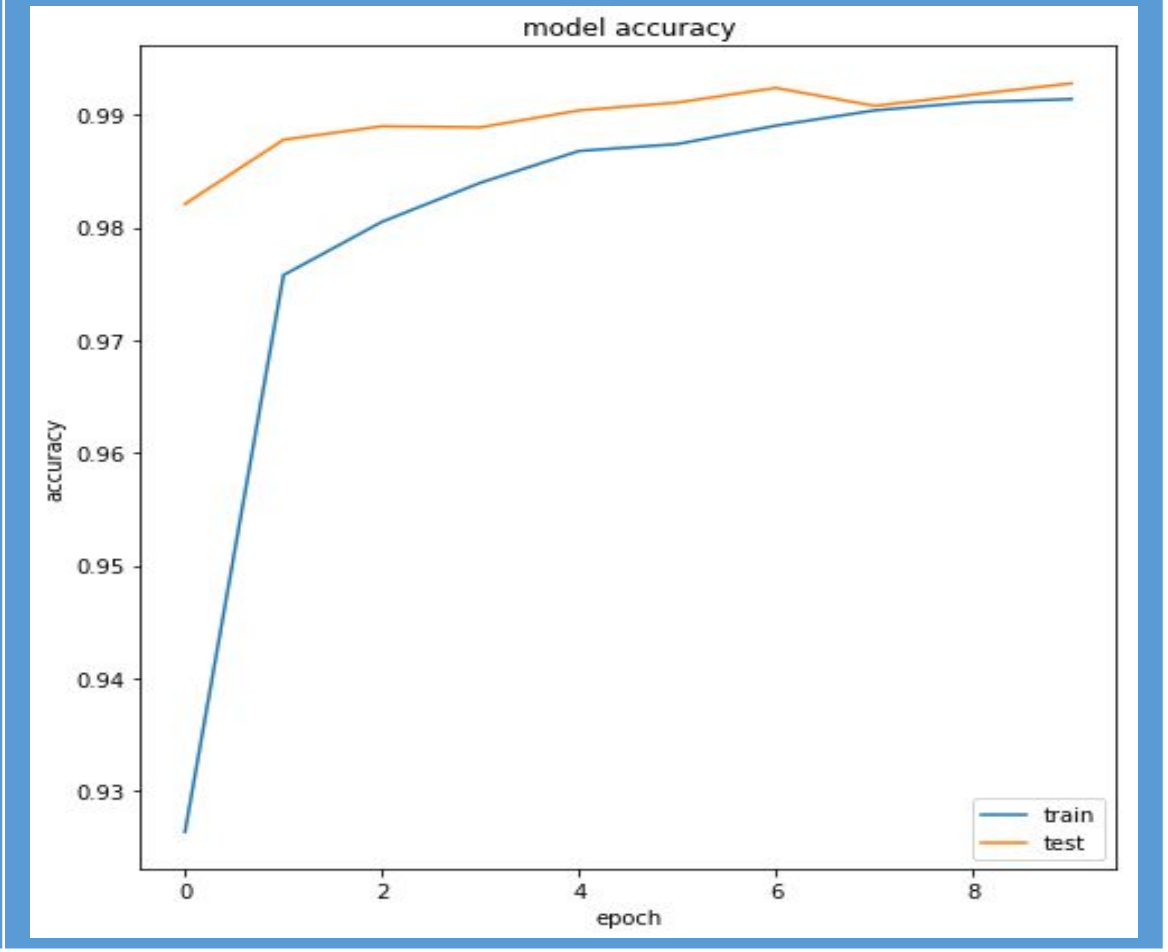


Model Accuracy vs. Epoch in CPU

Results in GPU of CNN Model

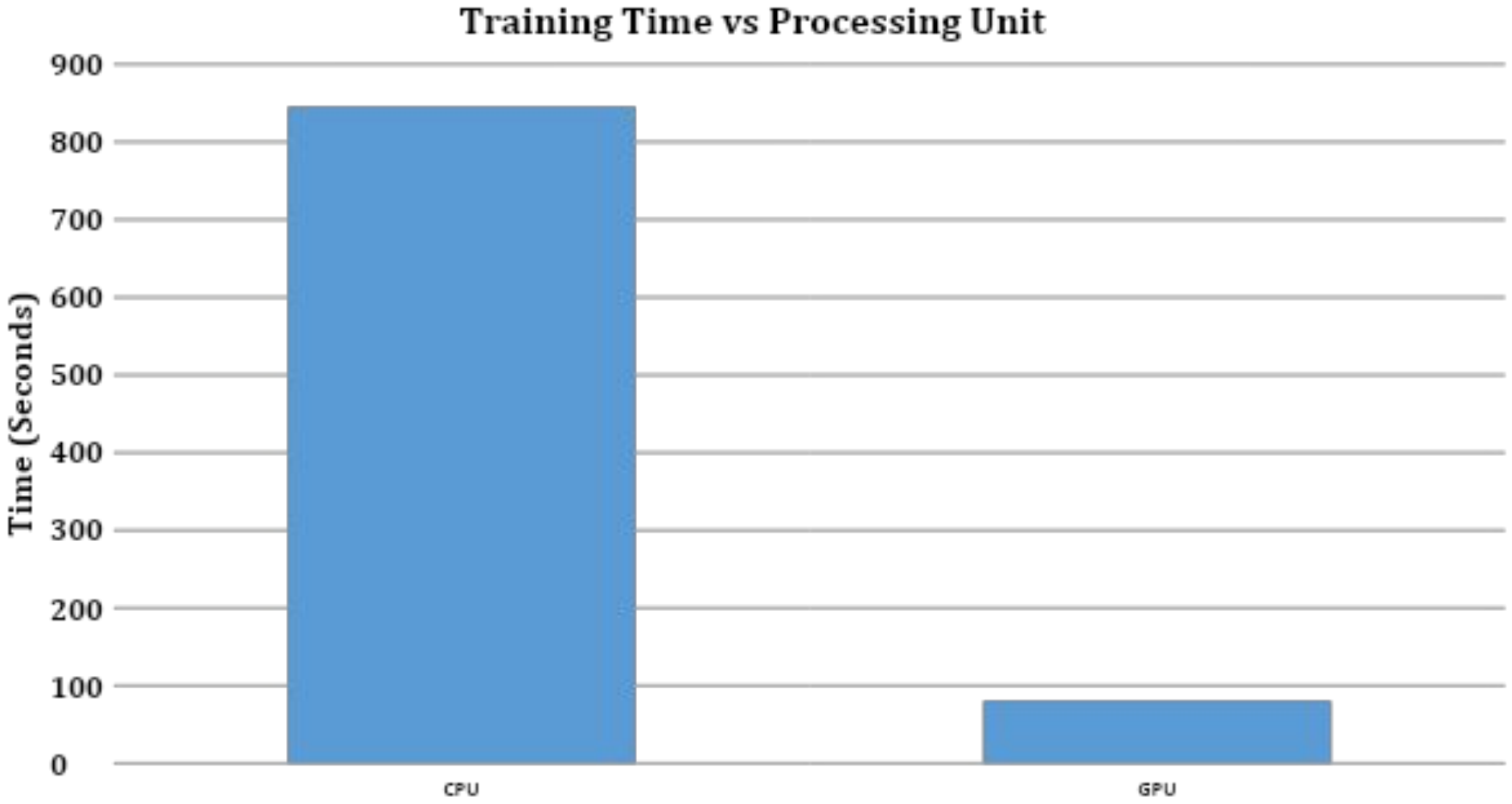


Model Loss vs. Epoch in GPU



Model Accuracy vs. Epoch in GPU

Comparison of training time CNN model in CPU and GPU



ML/DL Conferences/Journals

- **CVPR** - Conference on Computer Vision and Pattern Recognition
- **NIPS** - Neural Information processing systems
- **ICML** - International conference on machine learning
- **ECCV** - European Conference on Computer Vision
- **ICCV** - International Conference on Computer Vision
- **ECML** - European Conference on Machine Learning
- **IJCAI** - International Joint Conference on Artificial Intelligence
- **JML** - Journal of Machine Learning
- IEEE Transactions on Neural Networks
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- Neural Networks
- Artificial Intelligence
- Pattern Recognition
- Expert Systems with Applications
- IEEE Transactions on Knowledge and Data Engineering
- Neural Computing and Applications
- IEEE Transactions on Audio, Speech and Language Processing
- International Journal of Neural Systems

Reading Materials

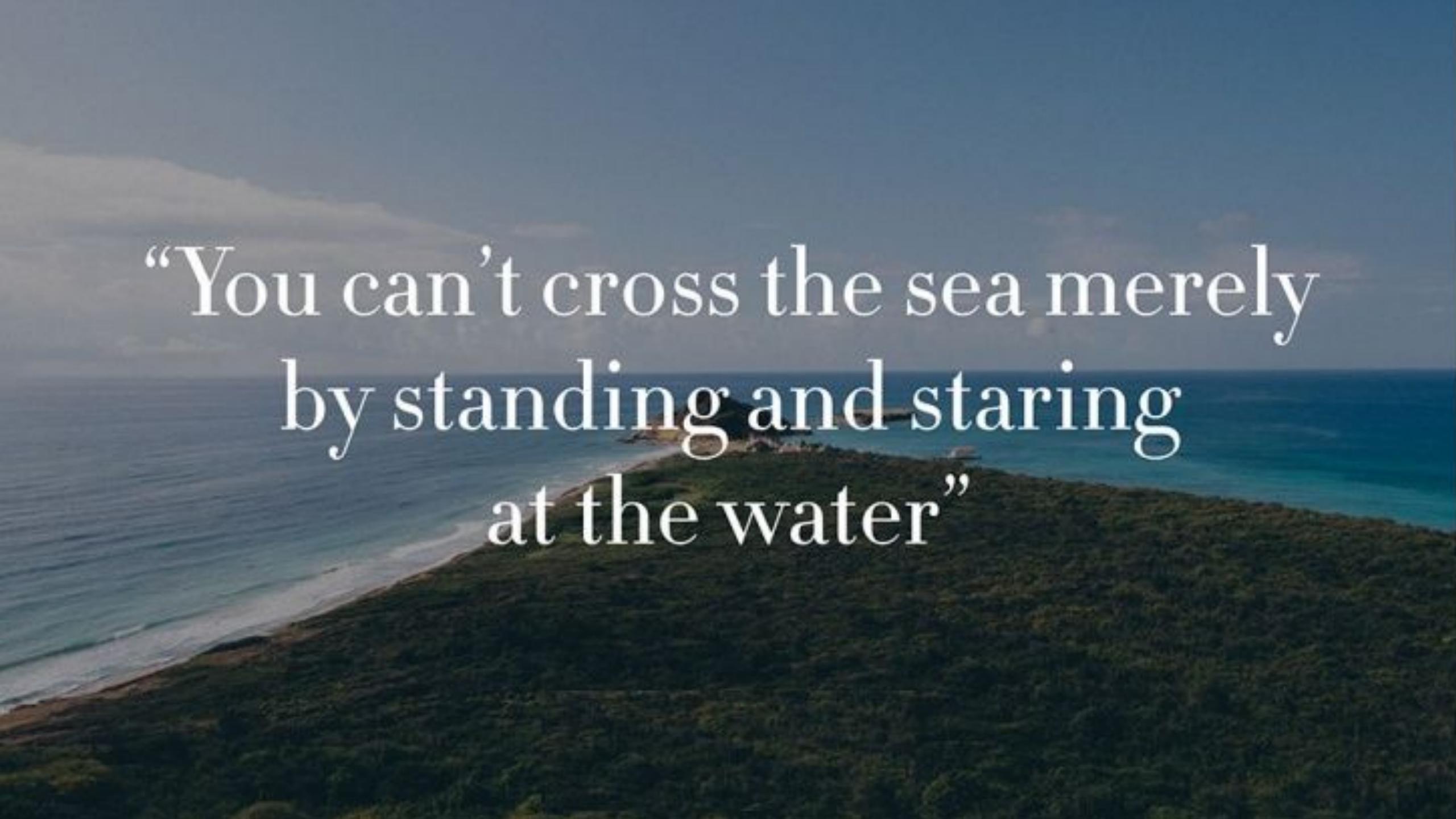
- **Text Book:** https://www.deeplearningbook.org/
- https://www.coursera.org/learn/machine-learning
- deeplearning.ai
- http://cs231n.stanford.edu/
- https://github.com/floodsung/Deep-Learning-Papers-Reading-Roadmap
- https://github.com/terryum/awesome-deep-learning-papers

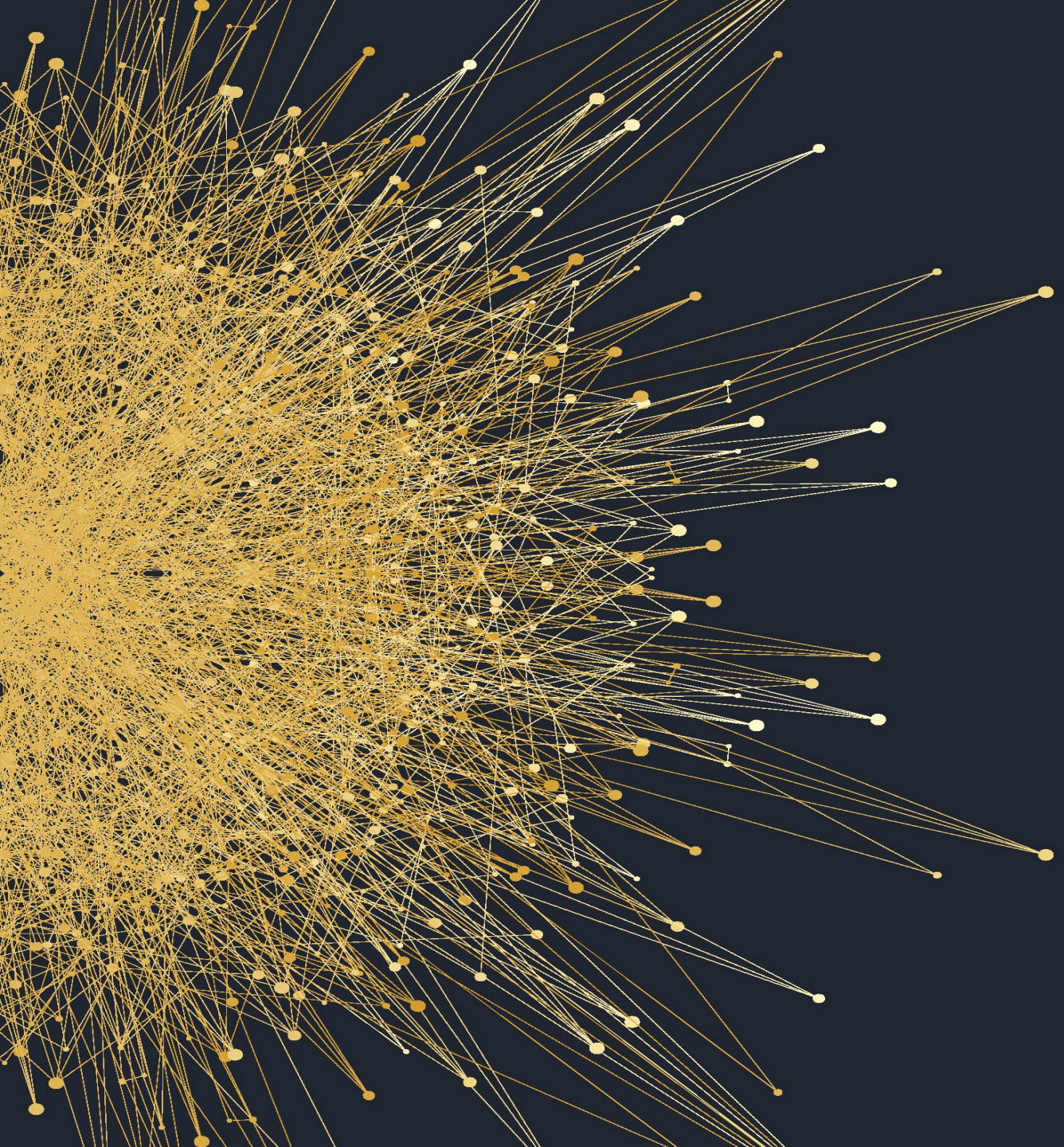
Reading Materials

- https://www.tensorflow.org/api_docs/python/
- https://www.toptal.com/machine-learning/tensorflow-machine-learning-tutorial
- https://www.matroid.com/dlwithtf/chap1-2.pdf
- http://web.donga.ac.kr/yjko/usefulthings/TensorFlow-Basic-Concept_Ko.pdf
- https://cs224d.stanford.edu/lectures/CS224d-Lecture7.pdf

References

- Abadi, Martín, et al. “Tensorflow: Large-scale machine learning on heterogeneous distributed systems.” *arXiv preprint arXiv:1603.04467* (2016).
- https://github.com/tensorflow/tensorflow
- https://www.youtube.com/watch?v=yX8KuPZCAMo
- https://www.youtube.com/watch?v=E8n_k6HNAgs&t=319s


“You can’t cross the sea merely
by standing and staring
at the water”



Thank you!

Any Question ?