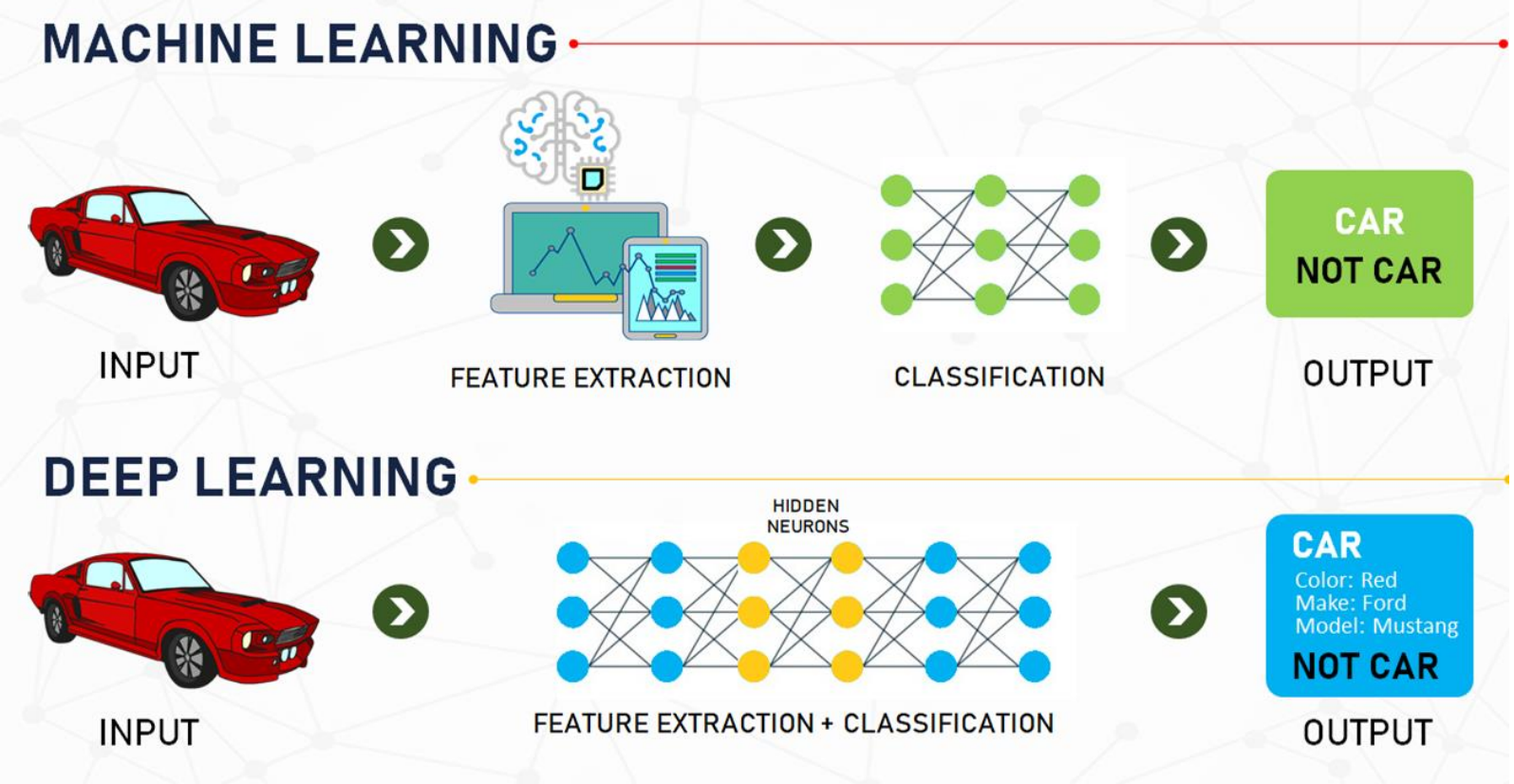
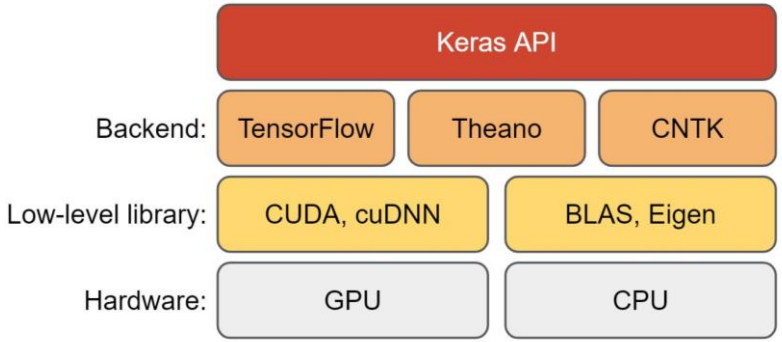
Deeplearning & Tensorflow

SHAILESH S

Deeplearning



Deeplearning Technology Stack



Caffe2 · Microsoft CNTK · TensorFlow · theano · Chainer · PYTORCH · dmlc mxnet · K

Keras

Keras is an effective high-level neural network Application Programming Interface (API) written in Python

PyTorch

- Pytorch is a relatively new deep learning framework based on Torch.
- Developed by Facebook's AI research group
- Pytorch has a reputation for simplicity, ease of use, flexibility, efficient memory usage, and dynamic computational graphs.

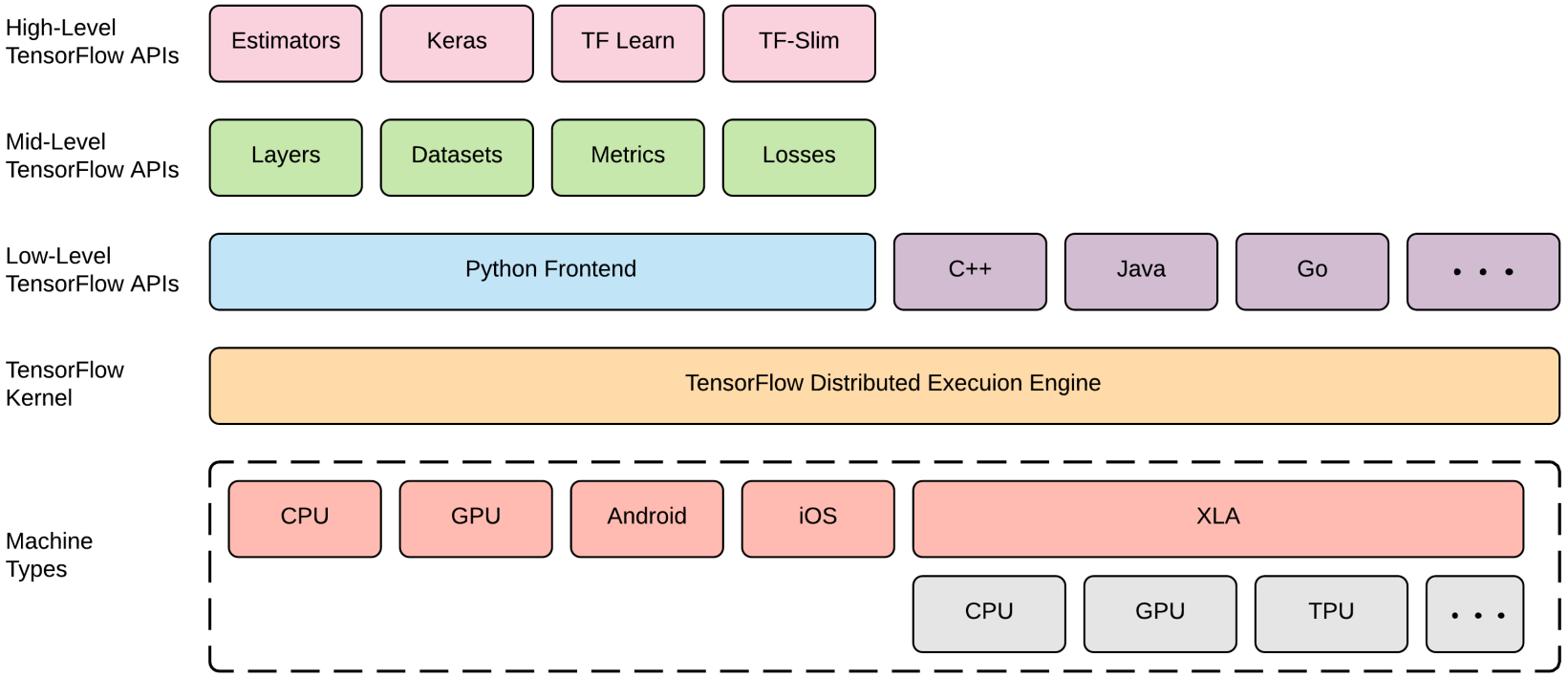
Tensorflow

- TensorFlow is an end-to-end open-source deep learning framework developed by Google
- It is known for scalable production and deployment options, multiple abstraction levels, and support for different platforms, such as Android

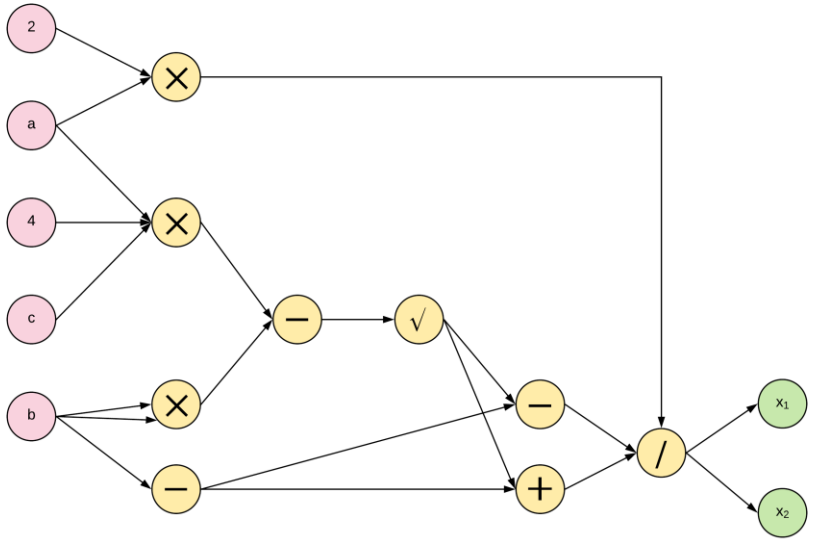
Theano

- Theano is popular deep learning libraries, an open-source project
- Theano was developed by the Universite de Montreal
- It lets programmers define, evaluate, and optimize mathematical expressions, including multi-dimensional arrays and matrix-valued expressions

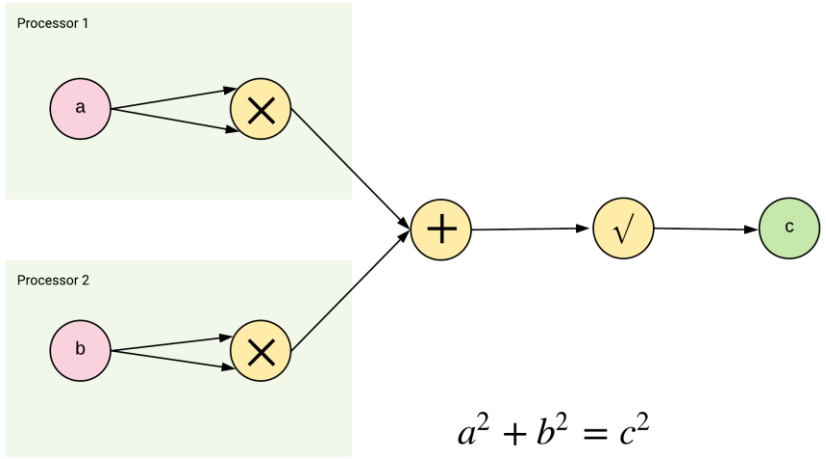
Tensorflow


High-Level TensorFlow APIs
Estimators
Keras
TF Learn
TF-Slim
Mid-Level TensorFlow APIs
Layers
Datasets
Metrics
Losses
Low-Level TensorFlow APIs
Python Frontend
C++
Java
Go
. . .
TensorFlow Kernel
TensorFlow Distributed Execuion Engine
Machine Types
CPU
GPU
Android
iOS
XLA
CPU
GPU
TPU
. . .

Tensorflow Computational Graphs



$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



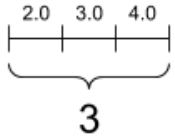
$$a^2 + b^2 = c^2$$

Tensors

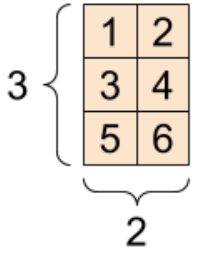
A scalar, shape: [ ]

4

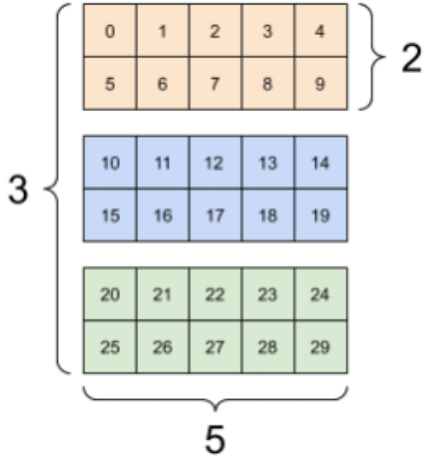
A vector, shape: [3]

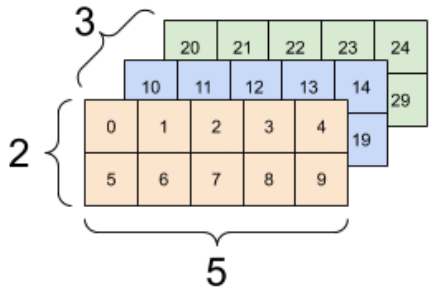


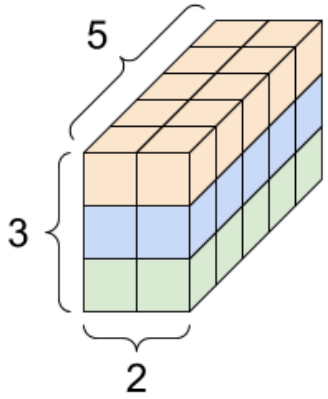
A matrix, shape: [3, 2]



A 3-axis tensor, shape: [3, 2, 5]

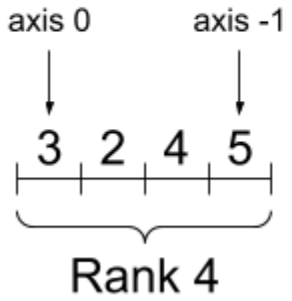


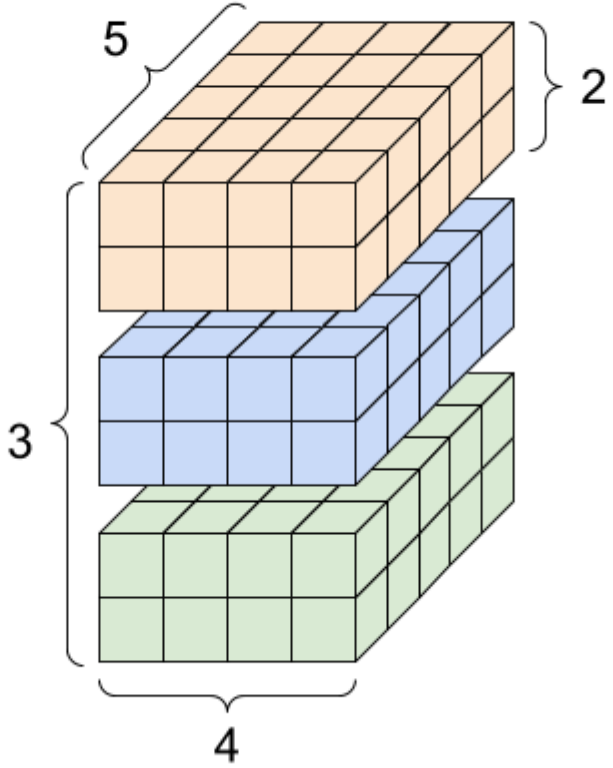




Tensors

A rank-4 tensor, shape: [3, 2, 4, 5]





Tensors

```
# This will be an int32 tensor by default; see "dtypes" below.

rank_0_tensor = tf.constant(4)
print(rank_0_tensor)


# Let's make this a float tensor.

rank_1_tensor = tf.constant([2.0, 3.0, 4.0])
print(rank_1_tensor)

# If you want to be specific, you can set the dtype (see below) at creation time

rank_2_tensor = tf.constant([[1, 2],
                             [3, 4],
                             [5, 6]], dtype=tf.float16)
print(rank_2_tensor)
```

Tensors

```
# There can be an arbitrary number of
# axes (sometimes called "dimensions")

rank_3_tensor = tf.constant([
  [[0, 1, 2, 3, 4],
   [5, 6, 7, 8, 9]],
  [[10, 11, 12, 13, 14],
   [15, 16, 17, 18, 19]],
  [[20, 21, 22, 23, 24],
   [25, 26, 27, 28, 29]],])

print(rank_3_tensor)
```

Variables

```
my_tensor = tf.constant([[1.0, 2.0], [3.0, 4.0]])

my_variable = tf.Variable(my_tensor)

# Variables can be all kinds of types, just like tensors

bool_variable = tf.Variable([False, False, False, True])

complex_variable = tf.Variable([5 + 4j, 6 + 1j])
```

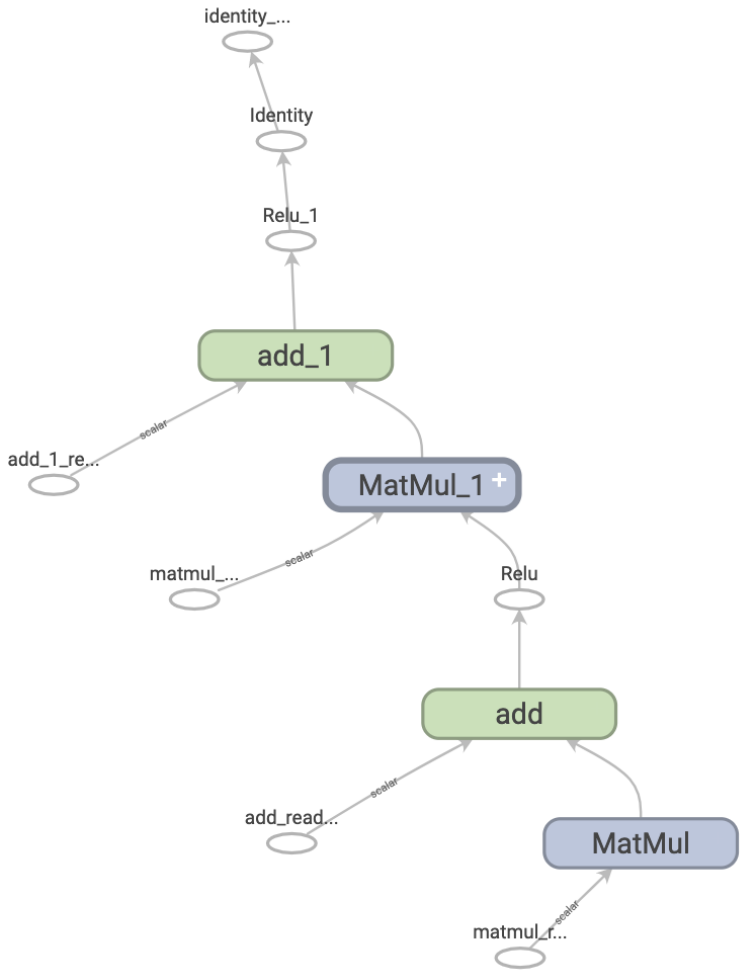
Tensorflow Graphs

Eagerly Execution means TensorFlow operations are executed by Python, operation by operation, and returning results back to Python.

Graph execution means that tensor computations are executed as a *TensorFlow graph*, sometimes referred to as a tf.Graph or simply a "graph."

While eager execution has several unique advantages, graph execution enables portability outside Python and tends to offer better performance

Graphs are data structures that contain a set of tf.Operation objects, which represent units of computation; and tf.Tensor objects, which represent the units of data that flow between operations. They are defined in a tf.Graph context.



THANK YOU