TensorFlow & Keras

Deep Learning Framework

theano

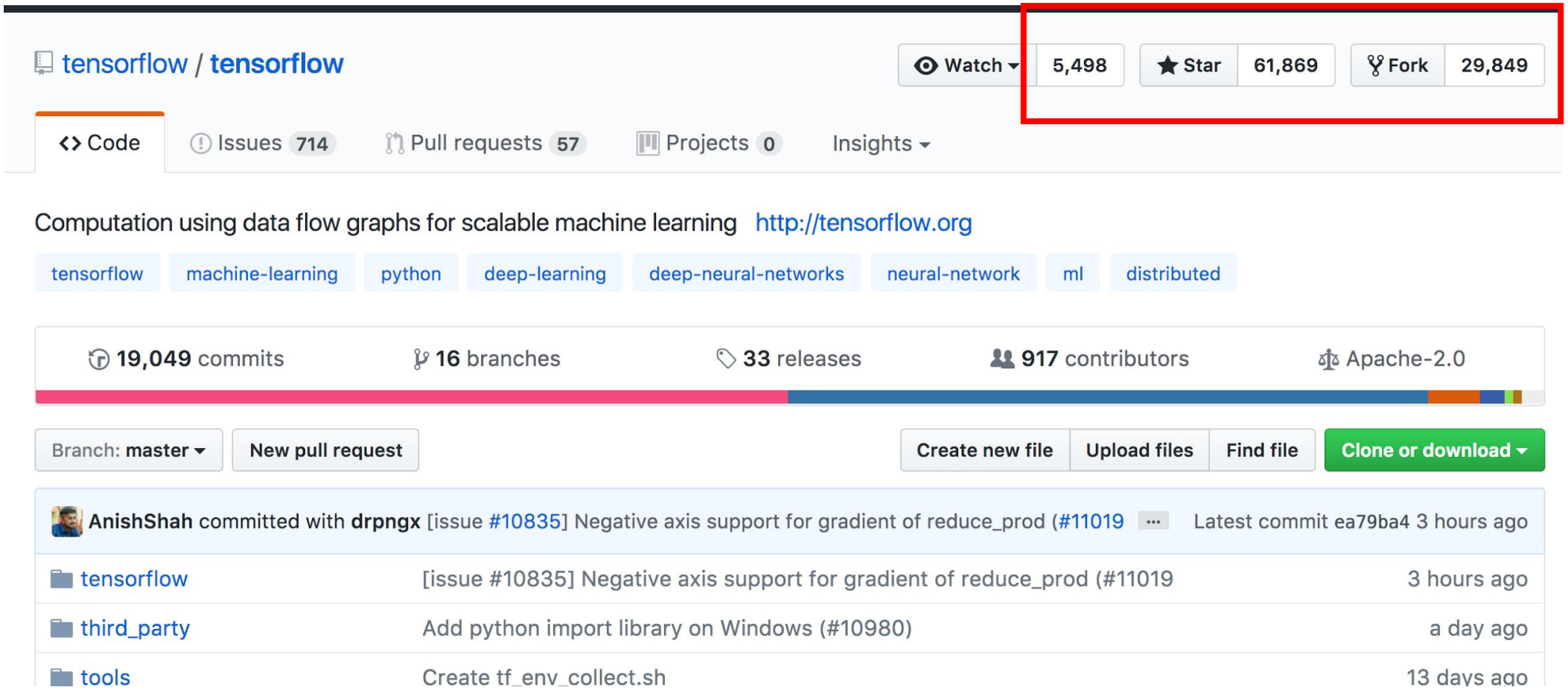
Caffe

Overview: TensorFlow

What is TensorFlow?

- URL: https://www.tensorflow.org/
- Released under the open source license on November 9, 2015
- Current version 1.2
- Open source software library for numerical computation using data flow graphs
- Originally developed by Google Brain Team to conduct machine learning and deep neural networks research
- General enough to be applicable in a wide variety of other domains as well
- TensorFlow provides an extensive suite of functions and classes that allow users to build various models from scratch.

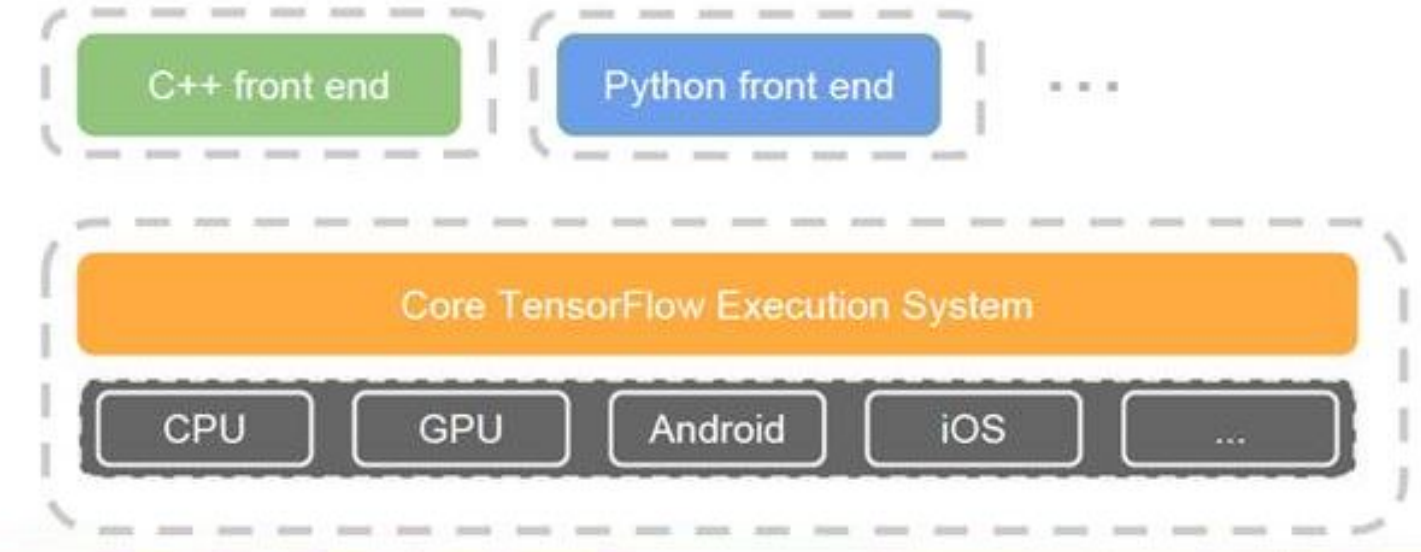
Most popular on Github



https://github.com/tensorflow/tensorflow

TensorFlow architecture

- Core in C++
  - Low overhead
- Different front ends for specifying/driving the computation
  - Python, C++, R and many more

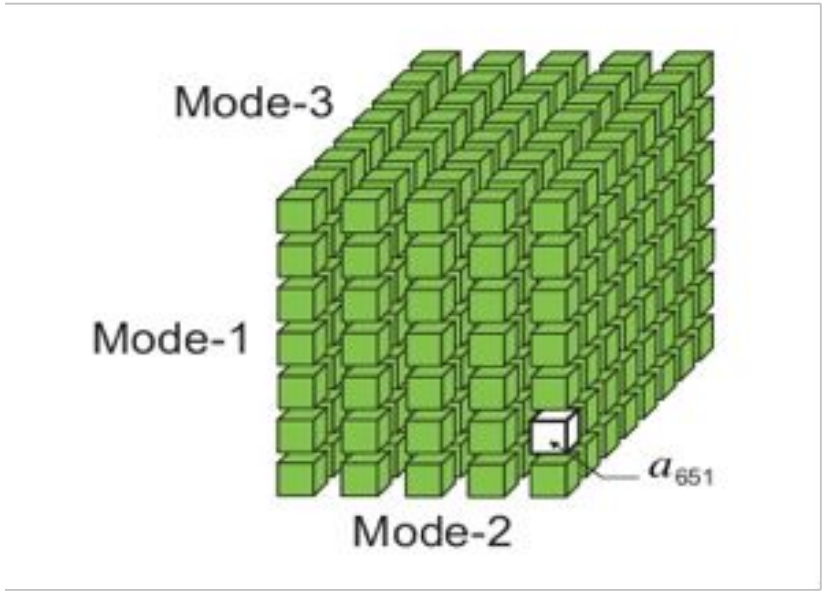


CPU - GPU

- In TensorFlow, the supported device types are CPU and GPU. They are represented as strings. For example:
  - `"/cpu:0"`: The CPU of your machine.
  - `"/gpu:0"`: The GPU of your machine, if you have one.
  - `"/gpu:1"`: The second GPU of your machine, etc.

```
# Creates a graph.
with tf.device('/gpu:2'):
  a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
  b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
  c = tf.matmul(a, b)
# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
# Runs the op.
print(sess.run(c))
```

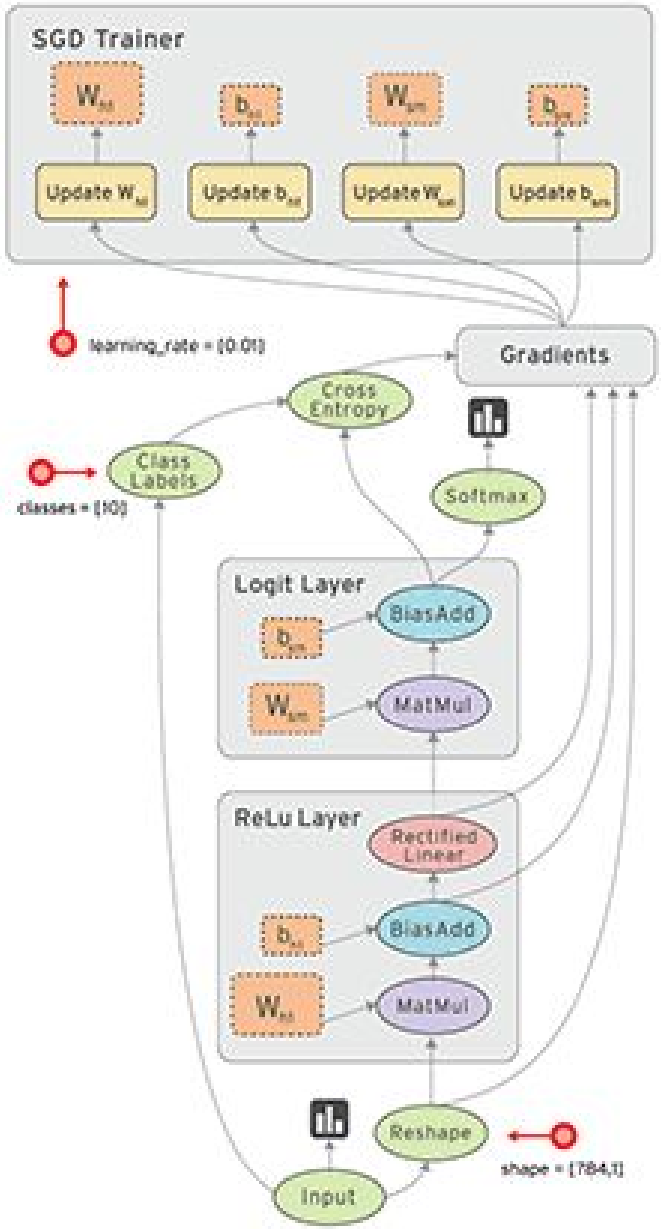
What is a Tensor?



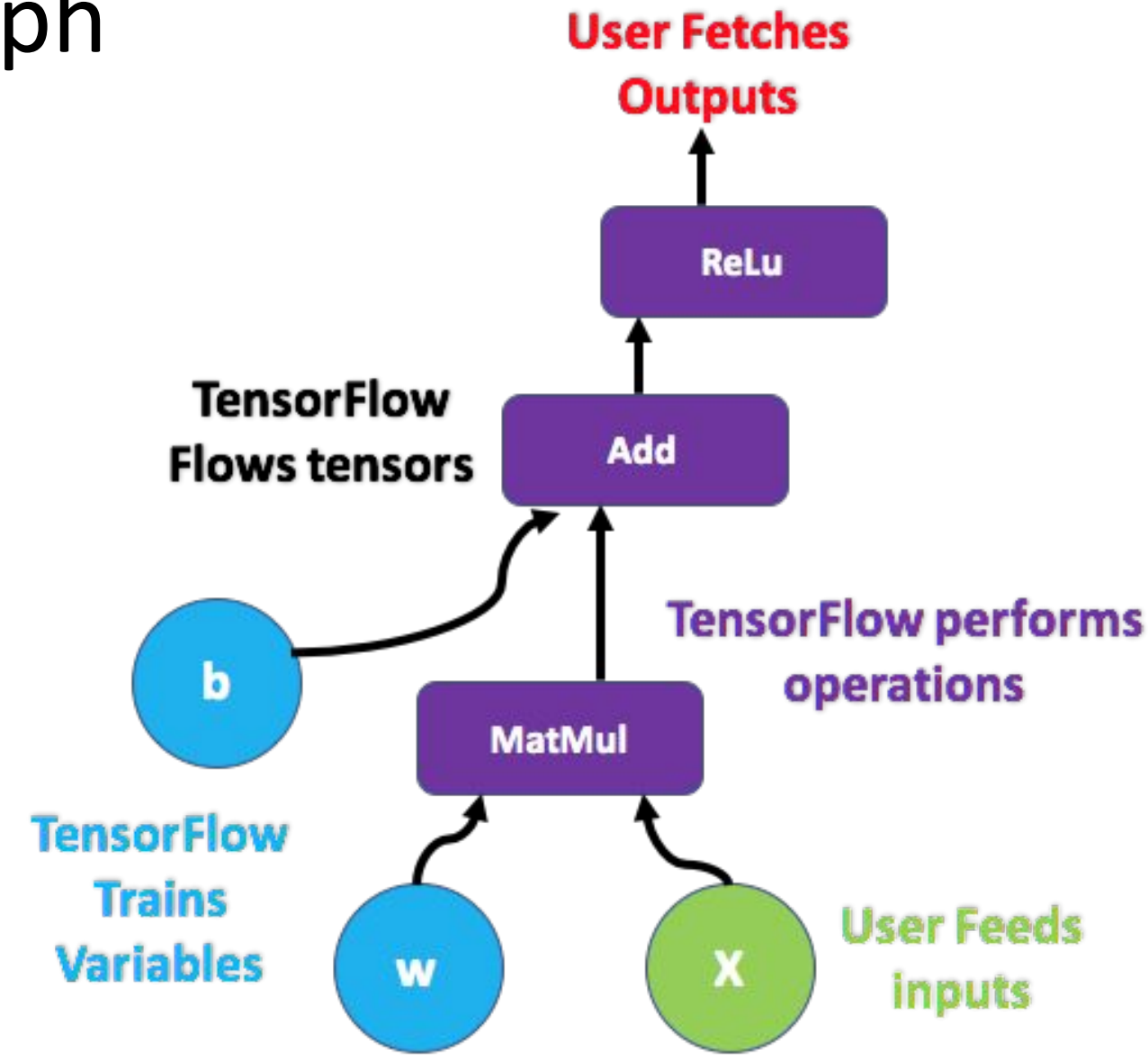
Dimensions	Example	Terminology
1	0 1 2	Vector
2	0 1 2 3 4 5 6 7 8	Matrix
3	0 1 2 3 4 5 6 7 8	3D Array (3rd order Tensor)
N	... ...	ND Array

Why it is called TensorFlow?

- TensorFlow is based on computation data flow graph
- TensorFlow separates definition of computations from their execution

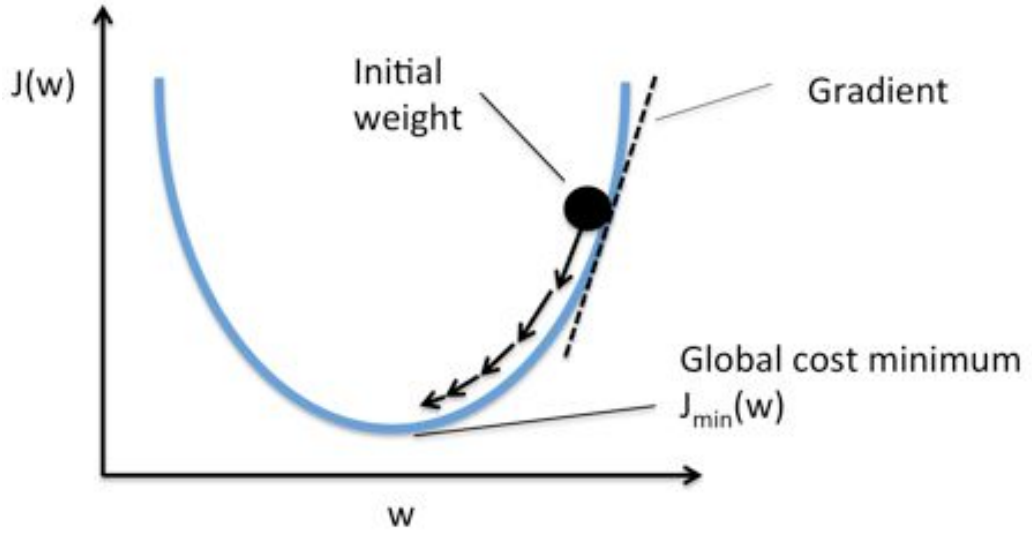


TensorFlow Graph

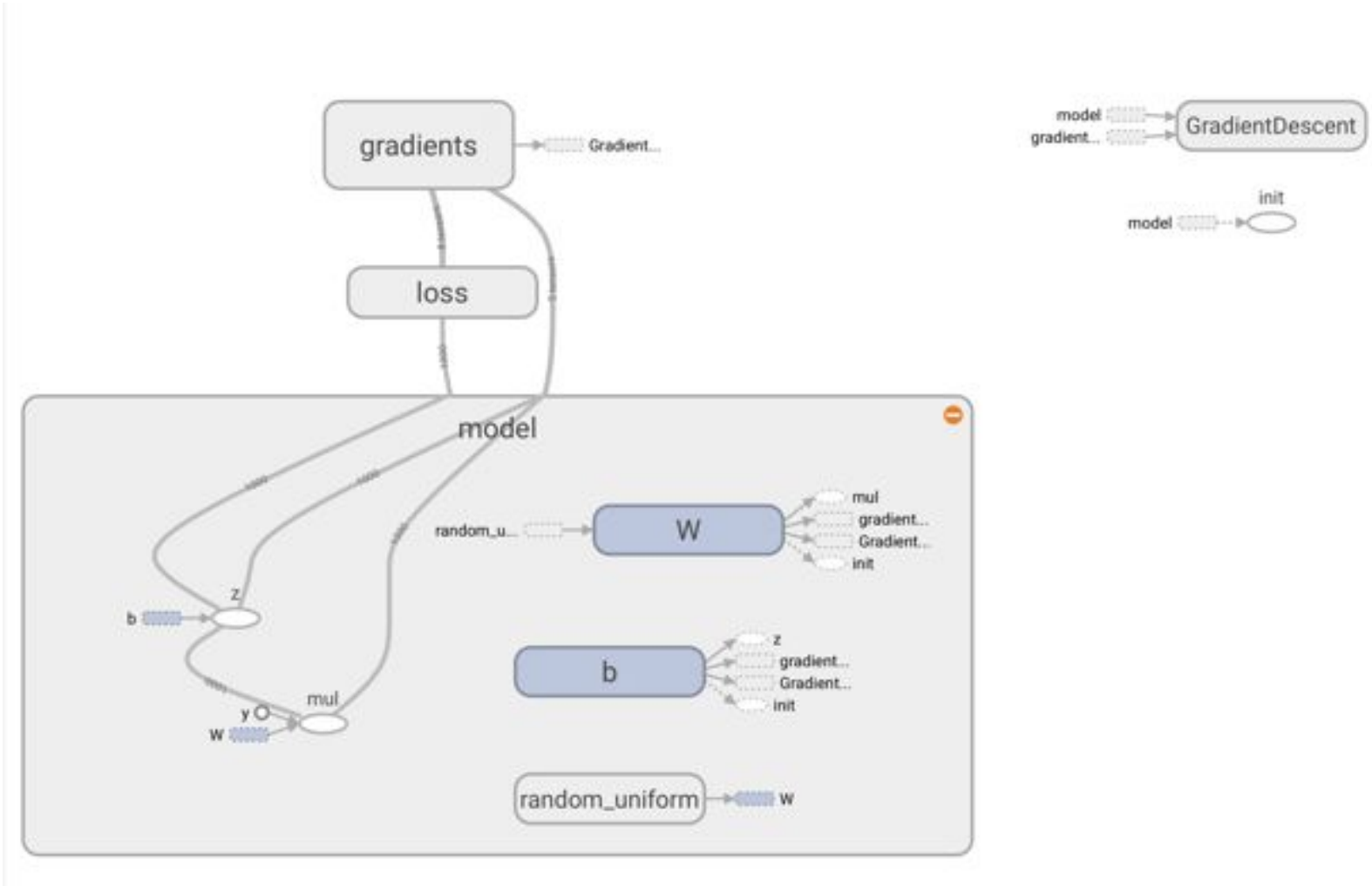


Learn Parameters: Optimization

- The Optimizer base class provides methods to compute gradients for a loss and apply gradients to variables.
- A collection of subclasses implement classic optimization algorithms such as GradientDescent and Adagrad.
- TensorFlow provides functions to compute the derivatives for a given TensorFlow computation graph, adding operations to the graph.

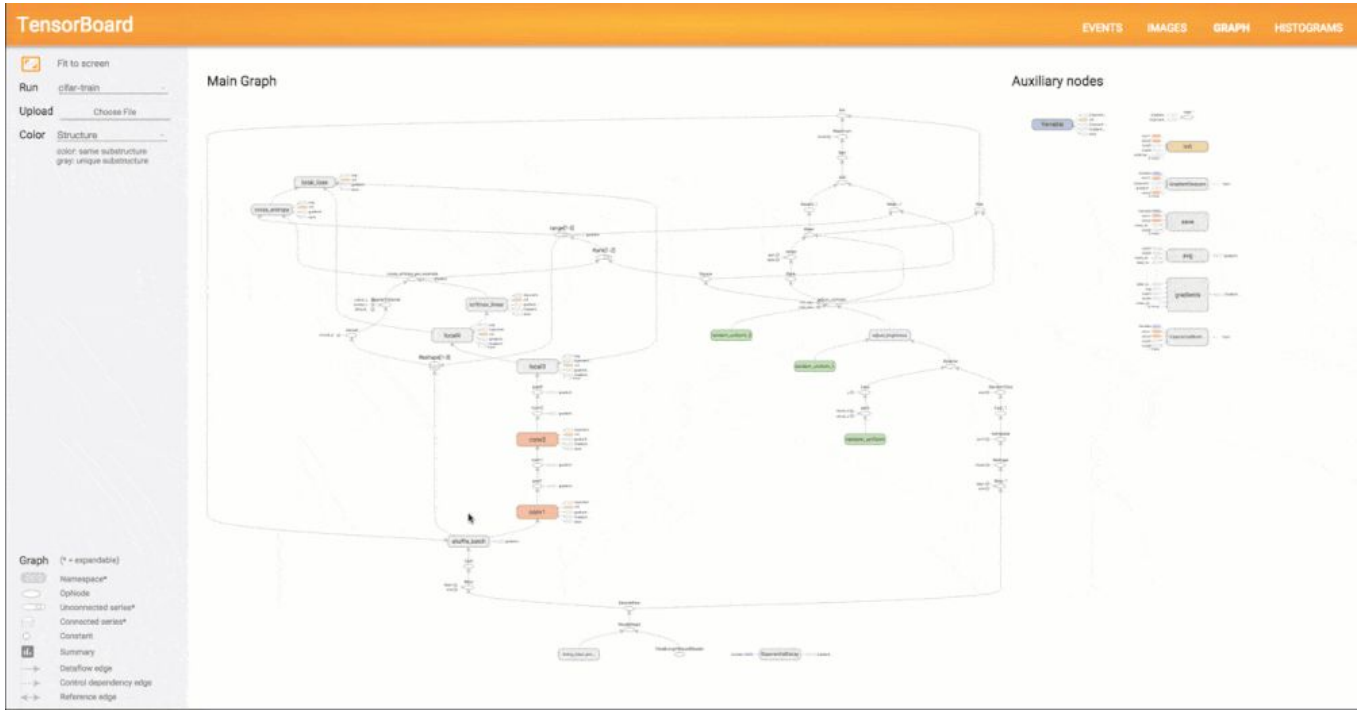


Learn Parameters: Optimization



TensorBoard

- Visualize your TensorFlow graph
- Plot quantitative metrics about the execution of your graph
- Show additional data like images that pass through it



TensorFlow Models

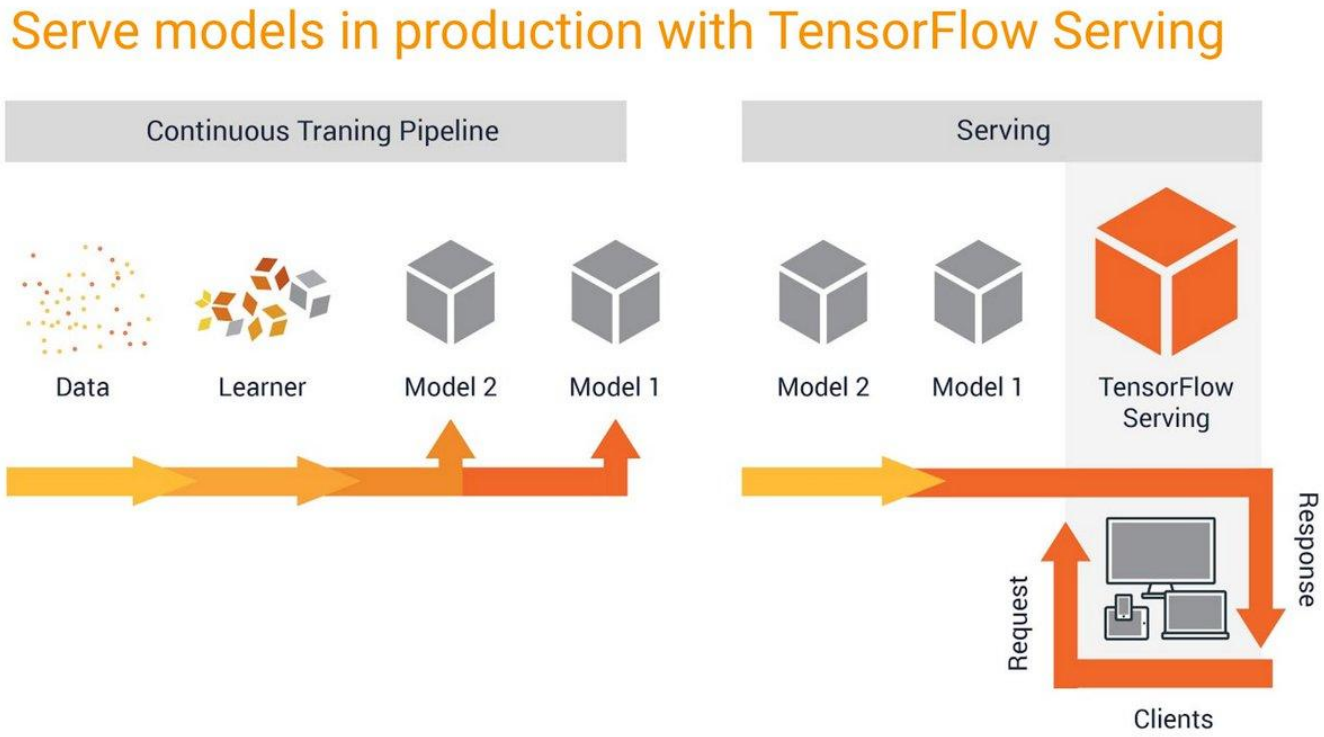
https://github.com/tensorflow/models

Models

- adversarial_crypto: protecting communications with adversarial neural cryptography.
- adversarial_text: semi-supervised sequence learning with adversarial training.
- attention_ocr: a model for real-world image text extraction.
- autoencoder: various autoencoders.
- cognitive_mapping_and_planning: implementation of a spatial memory based mapping and planning architecture for visual navigation.
- compression: compressing and decompressing images using a pre-trained Residual GRU network.
- differential_privacy: privacy-preserving student models from multiple teachers.
- domain_adaptation: domain separation networks.
- im2txt: image-to-text neural network for image captioning.
- inception: deep convolutional networks for computer vision.

TensorFlow Serving

- Flexible, high-performance serving system for machine learning models, designed for production environments.
- Easy to deploy new algorithms and experiments, while keeping the same server architecture and APIs



Code snippet

```
import tensorflow as tf

# build a linear model where y = w * x + b

w = tf.Variable([0.2], tf.float32, name='weight')
b = tf.Variable([0.3], tf.float32, name='bias')

X = tf.placeholder(tf.float32, name="X")
Y = tf.placeholder(tf.float32, name='Y')

# the training values for x and y
x = ([2.,3.,4.,5.])
y = ([-1.,-2.,-3.,-4.])

# define the linear model
linear_model = w*X+b

# define the loss function
square_delta = tf.square(linear_model - Y)
loss = tf.reduce_sum(square_delta)

#set the learning rate and training epoch
learning_rate = 0.01
training_epoch = 1000
```

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

# start a session
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)

    for i in range(training_epoch):
        sess.run(train, feed_dict={X:x,Y:y})

    # evaluate training accuracy
    curr_w, curr_b, curr_loss  = sess.run([w, b, loss], {X:x,Y:y})
    print('w: %f b: %f loss: %f '%(curr_w, curr_b, curr_loss))
```

TensorFlow: Installation

- `pip3 install tensorflow`

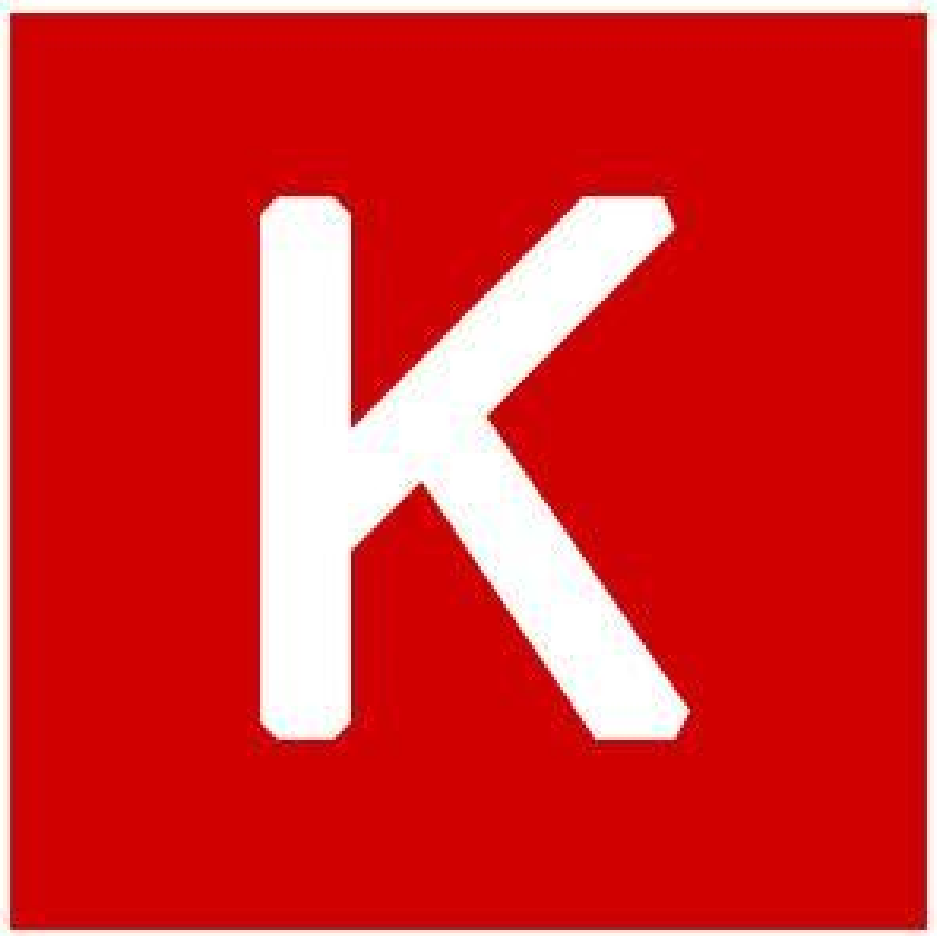
or

- Anaconda: `conda install -c conda-forge tensorflow`

Overview: Keras

Keras

- https://keras.io/
- Minimalist, highly modular neural networks library
- Written in Python
- Capable of running on top of either TensorFlow/Theano and CNTK
- Developed with a focus on enabling fast experimentation



Guiding Principles

- Modularity
  - A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible
- Minimalism
  - Each module should be kept short and simple
- Easy extensibility
  - New modules can be easily added and extended
- Python
  - Supports Python

General Design

- General idea is to based on layers and their input/output
  - Prepare your inputs and output tensors
  - Create first layer to handle input tensor
  - Create output layer to handle targets
  - Build virtually any model you like in between

Layers

- Keras has a number of pre-built layers. Notable examples include:
  - Regular dense, MLP type

    keras.layers.core.Dense(units, activation=**None**, use_bias=**True**, kernel_initializer='glorot_uniform', bias_initializer='zeros', kernel_regularizer=**None**, bias_regularizer=**None**, activity_regularizer=**None**, kernel_constraint=**None**, bias_constraint=**None**)
  - Recurrent layers, LTSM, GRU, etc

    keras.layers.recurrent.Recurrent(return_sequences=**False**, return_state=**False**, go_backwards=**False**, stateful=**False**, unroll=**False**, implementation=0)

Layers

- 1D Convolutional layers

  keras.layers.convolutional.Conv1D(filters, kernel_size, strides=1, padding='valid', dilation_rate=1, activation=**None**, use_bias=**True**, kernel_initializer='glorot_uniform', bias_initializer='zeros', kernel_regularizer=**None**, bias_regularizer=**None**, activity_regularizer=**None**, kernel_constraint=**None**, bias_constraint=**None**)

- 2D Convolutional layers

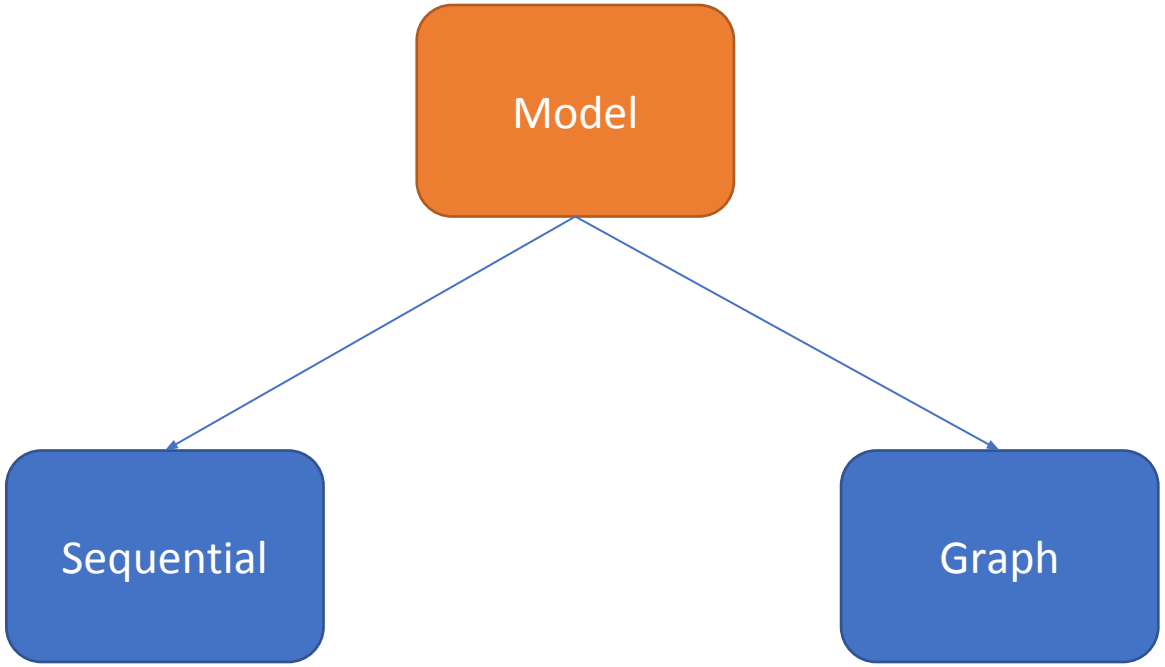
  keras.layers.convolutional.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid', data_format=**None**, dilation_rate=(1, 1), activation=**None**, use_bias=**True**, kernel_initializer='glorot_uniform', bias_initializer='zeros', kernel_regularizer=**None**, bias_regularizer=**None**, activity_regularizer=**None**, kernel_constraint=**None**, bias_constraint=**None**)

Why use Keras?

- Easy and fast prototyping (through total modularity, minimalism, and extensibility).
- Supports both convolutional networks and recurrent networks and combinations of the two.
- Supports arbitrary connectivity schemes (including multi-input and multi-output training).
- Runs seamlessly on CPU and GPU.

Keras Code examples

- The core data structure of Keras is a **model**
- Model → a way to organize layers



Code-snippet

```
import numpy as np
from keras.models import Sequential
from keras.layers.core import Activation, Dense
from keras.optimizers import SGD

X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0],[1],[1],[0]])

model = Sequential()
model.add(Dense(2, input_dim=2, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))

sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)

history = model.fit(X, y, nb_epoch=10000, batch_size=4)
```

Keras: Installation

- **Theano :**
  - pip3 install Theano
- **CNTK**
  - $ export CNTK_BINARY_URL=... [choose link here]
  - pip3 install $CNTK_BINARY_URL
- **TensorFlow :**
  - $ export TF_BINARY_URL=... [choose link here]
  - pip3 install $TF_BINARY_URL
- **Keras :**
  - pip3 install keras
- **To enable GPU computing :**
  - NVidia Cuda
  - CuDNN
  - CNMeM

Demo

Slides & Codes available from Github:

https://github.com/kuanhoong/Magic_DeepLearning