lec2-1-5-TF-videoCapture-Convolution-2018-2-1

Harry Li, Ph.D.
Feb. 6, 2018

Version 1.0

TF Data Representation

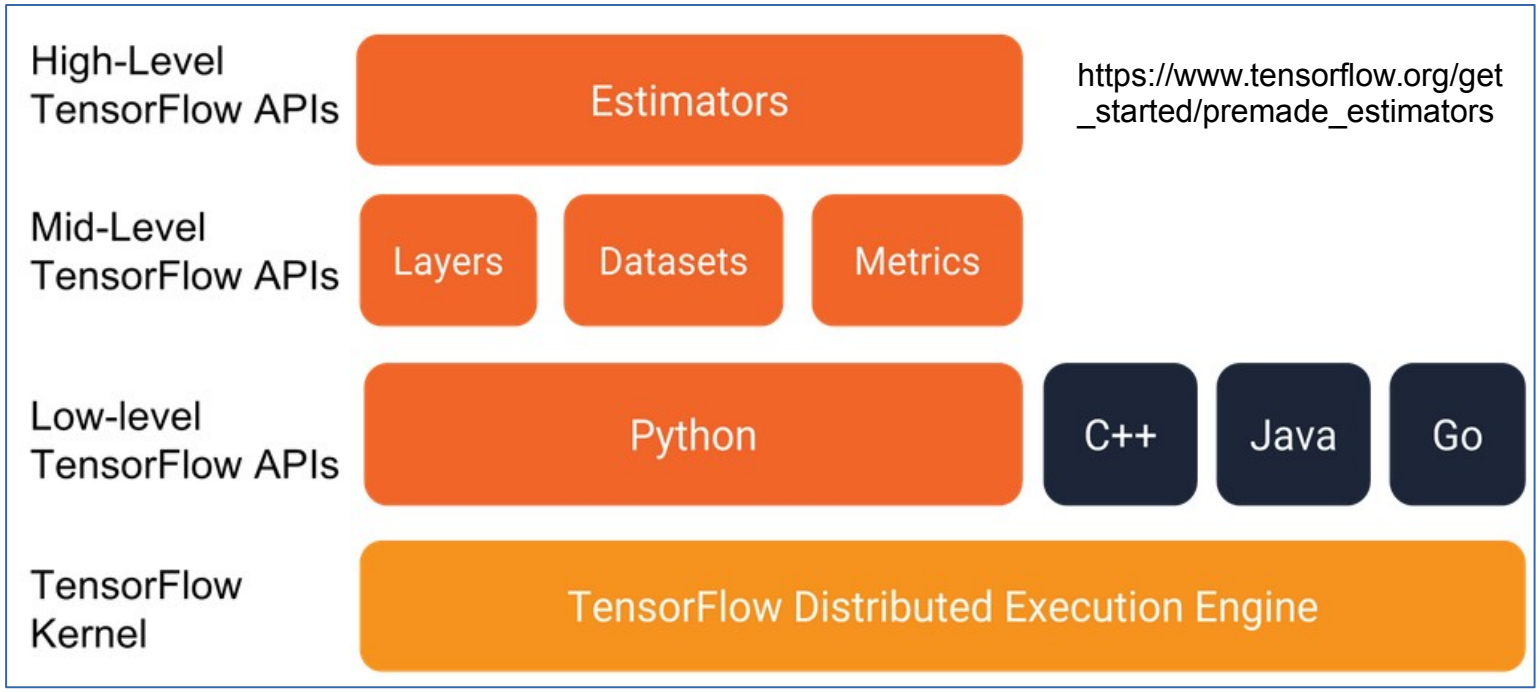
https://www.tensorflow.org/tutorials/

1. Linear Model Tutorial. Uses feature columns to feed a variety of data types to linear model, to solve a classification problem.

2. Deep Learning Tutorial. Builds on the above linear model tutorial, adding a deep feed-forward neural network component and a DNN-compatible data representation.

3. Vector Representations of Words, which demonstrates how to create an embedding for words.

4. Improving Linear Models Using Explicit Kernel Methods, which shows how to improve the quality of a linear model by using explicit kernel mappings.



https://www.tensorflow.org/get_started/premade_estimators

Harry Li, Ph.D.

Tensor Rank, Shape

https://www.tensorflow.org/tutorials/

First activate your TF as using command:

$source ~/tensorflow/bin/activate

Run the following lines to start

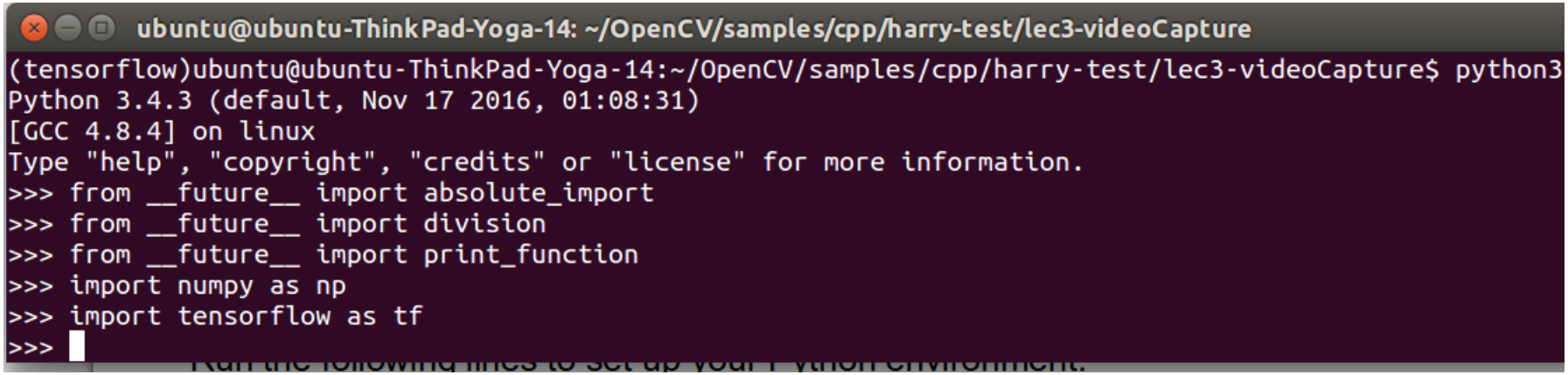
```
$python3

from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
import tensorflow as tf
```

Tensor ： The central unit of data which consists of a set of primitive values shaped into an array.

A tensor's rank: dimensions of the array.
A tensor's shape: a tuple of integers specifying the array's each dimension.



Define A Tensor As An Array

Example: 1D arrary

>>[1., 2., 3.] #a rank 1 tensor; a vector with shape [3

```
ubuntu@ubuntu-ThinkPad-Yoga-14: ~/OpenCV/sar
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license"
>>> from __future__ import absolute_import
>>> from __future__ import division
>>> from __future__ import print_function
>>> import numpy as np
>>> import tensorflow as tf
>>> [1., 2., 3.]
[1.0, 2.0, 3.0]
>>>
```

Example: 2D array (2x3) , e.g., two rows and three columns

>>[[1., 2., 3.], [4., 5., 6.]] #rank 2 tensor; a matrix shape [2, 3]

```
>>> [[1.,2.,3.],
... [4.,5.,6.]]
[[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]]
>>>
```

TensorBoard
TensorFlow provides a utility called TensorBoard. One of its capabilities is to visualize a computation graph.

https://www.tensorflow.org/programmers_guide/low_level_intro

Tensor (As An Array)

Example: 1D arrary

>>[1., 2., 3.] #a rank 1 tensor; a vector shape [3]

```
ubuntu@ubuntu-ThinkPad-Yoga-14: ~/OpenCV/sar
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license"
>>> from __future__ import absolute_import
>>> from __future__ import division
>>> from __future__ import print_function
>>> import numpy as np
>>> import tensorflow as tf
>>> [1., 2., 3.]
[1.0, 2.0, 3.0]
>>>
```

Note: these tensors are the same as python lists, arrays

Example: 2D array (2x3) , e.g., two rows and three columns

>>[[1., 2., 3.], [4., 5., 6.]] #rank 2 tensor; shape [2, 3]

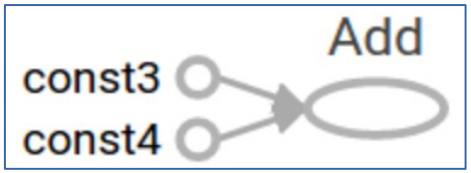
```
>>> [[1.,2.,3.],
... [4.,5.,6.]]
[[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]]
>>>
```

Math Ops On Tensors

https://www.tensorflow.org/programmers_guide/low_level_intro

Computational graph is a series of TensorFlow operations arranged into a graph, which compose of 2 types of objects.

1. Operations (or "ops"): The nodes of the graph. Operations describe calculations that consume and produce tensors.
2. Tensors: The edges in the graph. These represent the values that will flow through the graph. Most TensorFlow functions return tf.Tensors.



Example: addition

The Python function takes a tensor input, then perform addition and outputs the value passed to the constructor.

```
a = tf.constant(3.0, dtype=tf.float32)
b = tf.constant(4.0) # also tf.float32 implicitly
total = a + b
sess = tf.Session()
print(sess.run(total))
```

Create a session

Execute the session

```
>>> a = tf.constant(3.0, dtype=tf.float32)
>>> b = tf.constant(4.0) # also tf.float32 implicitly
>>> total = a + b
>>>
```

Define floating point constants

```
ubuntu@ubuntu-ThinkPad-Yoga-14: ~/OpenCV/samples
>>> sess = tf.Session()
2018-02-10 17:50:24.752413: I tensorflow/core/common_
u:0) -> (device: 0, name: GeForce 940M, pci bus id: 0
>>> print(sess.run(total))
7.0
>>>
```

Resulte

Harry Li, Ph.D.

Discrete 2D Convolution Without Boundary Condition

http://mourafiq.com/2016/08/10/playing-with-convolutions-in-tensorflow.html

$$M = \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} & m_{04} \\ m_{10} & m_{11} & m_{12} & m_{13} & m_{14} \\ m_{20} & m_{21} & m_{22} & m_{23} & m_{24} \\ m_{30} & m_{31} & m_{32} & m_{33} & m_{34} \\ m_{40} & m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix}$$

Note: this reference also gives LeNet computation, see the url link above.

and

$$c = \begin{pmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{pmatrix}$$

$$\begin{aligned} conv(M, c)[1, 1] = & m_{11} * c_{00} + m_{12} * c_{01} + m_{13} * c_{02} + \\ & m_{21} * c_{10} + m_{22} * c_{11} + m_{23} * c_{12} + \\ & m_{31} * c_{20} + m_{32} * c_{21} + m_{33} * c_{22} \end{aligned}$$

TF 2D Convolution for 2-by-2 Image and 1-by-1 Kernel

Tensorflow 2D convolution:
https://www.tensorflow.org/versions/r1.0/api_docs/python/tf/nn/conv2d

```
tf.nn.conv2d(input, filter, strides, padding, use_cudnn_on_gpu=None, data_format=None, name=None)
```

Example:

1 image, size 2x2, with 1 channel.

1 filter, size 1x1, and 1 channel (size is height x width x channels x number of filters).

```
input = tf.Variable(tf.random_normal([1,2,2,1]))
filter = tf.Variable(tf.random_normal([1,1,1,1]))

op = tf.nn.conv2d(input, filter, strides=[1, 1, 1, 1], padding='SAME')
init = tf.initialize_all_variables()
with tf.Session() as sess:
    sess.run(init)

    print("input")
    print(input.eval())
    print("filter")
    print(filter.eval())
    print("result")
    result = sess.run(op)
    print(result)
```

the result 2x2, 1 channel image (size 1x2x2x1, number of images x height x width x channels).

```
ubuntu@ubuntu-ThinkPad-Yoga-14: ~/OpenCV/s
2018-02-10 22:52:37.013796: I tensorflow/stream
84.98.0
2018-02-10 22:52:37.013803: I tensorflow/stream
 DSO: 384.98.0
input
[[[[ 0.50807762]
   [ 1.28140223]]

  [[ 1.10267854]
   [-0.96673602]]]]
filter
[[[[-0.98943055]]]]
result
[[[[-0.50270754]
   [-1.26785851]]

  [[-1.0910238 ]
   [ 0.95651817]]]]
```

https://stackoverflow.com/questions/34619177/what-does-tf-nn-conv2d-do-in-tensorflow

TF 2D Convolution for 4-by-4 Image and 3-by-3 Kernel

Tensorflow 2D convolution: https://www.tensorflow.org/versions/r1.0/api_docs/python/tf/nn/conv2d

```
tf.nn.conv2d(input, filter, strides, padding, use_cudnn_on_gpu=None, data_format=None, name=None)
```

calculates convolutions in batches, For an input it is [batch, in_height, in_width, in_channels]

Example:

Compute convolution with no padding, stride=1

$$input = \begin{pmatrix} 4 & 3 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 4 & 1 \\ 3 & 1 & 0 & 2 \end{pmatrix} \quad kernel = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The result:

$$\begin{pmatrix} 14 & 6 \\ 6 & 12 \end{pmatrix}$$

where:

14 = 4 * 1 + 3 * 0 + 1 * 1 + 2 * 2 + 1 * 1 + 0 * 0 + 1 * 0 + 2 * 0 + 4 * 1
6 = 3 * 1 + 1 * 0 + 0 * 1 + 1 * 2 + 0 * 1 + 1 * 0 + 2 * 0 + 4 * 0 + 1 * 1
6 = 2 * 1 + 1 * 0 + 0 * 1 + 1 * 2 + 2 * 1 + 4 * 0 + 3 * 0 + 1 * 0 + 0 * 1
12 = 1 * 1 + 0 * 0 + 1 * 1 + 2 * 2 + 4 * 1 + 1 * 0 + 1 * 0 + 0 * 0 + 2 * 1

```
2018-02-10 23:14:53.145858: I tensorflow
 DSO: 384.98.0
[[ 14.   6.]
 [  6.  12.]]
(tensorflow)ubuntu@ubuntu-ThinkPad-Yoga-
```

```
import tensorflow as tf
k = tf.constant([
    [1, 0, 1],
    [2, 1, 0],
    [0, 0, 1]
], dtype=tf.float32, name='k')
i = tf.constant([
    [4, 3, 1, 0],
    [2, 1, 0, 1],
    [1, 2, 4, 1],
    [3, 1, 0, 2]
], dtype=tf.float32, name='i')
kernel = tf.reshape(k, [3, 3, 1, 1], name='kernel')
image  = tf.reshape(i, [1, 4, 4, 1], name='image')
res = tf.squeeze(tf.nn.conv2d(image, kernel, [1, 1, 1, 1], "VALID"))
# VALID means no padding
with tf.Session() as sess:
   print (sess.run(res))
```

kernel is [filter_height, filter_width, in_channels, out_channels]

VALID means no padding

https://stackoverflow.com/questions/34619177/what-does-tf-nn-conv2d-do-in-tensorflow

Harry Li, Ph.D.