High Level Computer Vision

Tensorflow Primer
Tensor

- A tensor is TensorFlow's basic data structure
- It has three parameters: rank, shape and type
  - rank - the number of dimensions
  - shape - the number of elements in each dimension
  - type - the data type of the tensor’s elements

<table>
<thead>
<tr>
<th>Rank</th>
<th>Shape</th>
<th>Dimension number</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[]</td>
<td>0-D</td>
<td>A 0-D tensor. A scalar.</td>
</tr>
<tr>
<td>1</td>
<td>[D0]</td>
<td>1-D</td>
<td>A 1-D tensor with shape [5].</td>
</tr>
<tr>
<td>2</td>
<td>[D0, D1]</td>
<td>2-D</td>
<td>A 2-D tensor with shape [3, 4].</td>
</tr>
<tr>
<td>3</td>
<td>[D0, D1, D2]</td>
<td>3-D</td>
<td>A 3-D tensor with shape [1, 4, 3].</td>
</tr>
<tr>
<td>n</td>
<td>[D0, D1, ... Dn-1]</td>
<td>n-D</td>
<td>A tensor with shape [D0, D1, ... Dn-1].</td>
</tr>
</tbody>
</table>
Operations

- An **Operation** is a node that takes zero or more **Tensor** objects as input, and produces zero or more **Tensor** objects as output.

- A **Computational Graph** is series of TensorFlow **Operations** arranged into a graph of nodes

- A TensorFlow Core programs as consisting of two sections:
  - **Building** the computational graph. (defining **Tensors** and **Operations**)
  - **Running** the computational graph. (evaluating the **Tensors** in a **Session**)

Example Tensor (tensor.py)

```python
from __future__ import print_function
import tensorflow as tf

node1 = tf.constant(3.0, tf.float32)
node2 = tf.constant(4.0)  # also tf.float32 implicitly
print("Node 1: ", node1)
print("Node 2: ", node2)

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

$>$ export GPU_ID=3
$>$ CUDA_VISIBLE_DEVICES=${GPU_ID} python tensor.py

```
Node 1:  Tensor("Const:0", shape=(), dtype=float32)
Node 2:  Tensor("Const_1:0", shape=(), dtype=float32)

[3.0, 4.0]
```
Example Operation(multiplication.py)

```python
import tensorflow as tf

a = tf.placeholder('int32')
b = tf.placeholder('int32')
y = tf.multiply(a, b)

session = tf.Session()

print(session.run(y, feed_dict = {a : 6, b: 7}))
```

> export GPU_ID=3
> CUDA_VISIBLE_DEVICES=${GPU_ID} python multiplication.py
> 42
```python
from __future__ import print_function
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

y = tf.placeholder(tf.float32)
squared_deltas = tf.square(linear_model - y)
loss = tf.reduce_sum(squared_deltas)

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

    print('Linear model: ', sess.run(linear_model, {x:[1,2,3,4]}))

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

Linear model: [ 0. 0.30000001 0.60000002 0.90000004]
Loss 23.66
Changing W and b (linear_model.py)

```python
# Changing W and b
fixW = tf.assign(W, [-1.])
fixb = tf.assign(b, [1.])

with tf.Session() as sess:
    sess.run([fixW, fixb])
    print('New loss: ', sess.run(loss, {x:[1,2,3,4], y:[0,-1,-2,-3]}))
```

New loss:  0.0
```python
# Finding optimal W and b
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for _ in range(1000):
        sess.run(train, {x: [1, 2, 3, 4], y: [0, -1, -2, -3]})

print('After training')
print('W: ', W.eval())
print('b: ', b.eval())
```

Before training
W: [ 0.30000001]
b: [ -0.30000001]

After training
W: [-0.9999969]
b: [ 0.99999082]
Start Tensorflow

- source /BS/hlcv-ss17/work/tensorflow-tutorial/path.sh
  - Sets up Python and Cuda paths

- source activate tensorflow
  - Activates the tensorflow environment in anaconda python
More tutorials

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