# TensorFlow

Tom Rochette <tom.rochette@coreteks.org>

March 22, 2020 — c745621c

### 0.1 Context

TensorFlow has rapidly grown in popularity due to the fact that is developed/supported by Google. As more and more developers move to the platform, it becomes essential to learn how it works and have a general idea of the various concepts it makes use of. This is a short article about some of these concepts.

### 0.2 Learned in this study

### 0.3 Things to explore

## 1 Overview

- Computations are represented as graphs
- Graphs are executed in the context of `Sessions`

## 2 Building a graph

- Start with ops that do not need any input (called `source ops`), such as `Constant`

## 3 Session

- Graphs are executed within a session (context)
  `session = tf.session()`
- Sessions are given one or many tensor to resolve
  `session.run([tensorA, tensorB])`
- Once we're done with a session, it should be closed
  `session.close()`

## 4 Tensors

A tensor is simply a multidimensional array of data. A scalar is a 0-D tensor, a vector is a 1-D tensor, a matrix is a 2-D tensor and anything over 3-D is called an n-D tensor.

**Rank:** The number of dimensions of a tensor.

| Rank | Math entity | Example |
|---|---|---|
| 0 | Scalar | s = 483 |
| 1 | Vector | v = [1.1, 2.2, 3.3] |
| 2 | Matrix | m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]] |
| 3 | 3-Tensor | t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]] |

**Shape:** A vector describing the number of elements at each point within a dimension.

| Rank | Shape | Dimension number | Example |
|---|---|---|---|
| 0 | [] | 0-D | A 0-D tensor. A scalar. |
| 1 | [D0] | 1-D | A 1-D tensor with shape [5] = [1, 2, 3, 4, 5]. |
| 2 | [D0, D1] | 2-D | A 2-D tensor with shape [3, 4] = [[1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 3, 4]]. |
| 3 | [D0, D1, D2] | 3-D | A 3-D tensor with shape [1, 4, 3] = [[[1, 2, 3], [1, 2, 3], [1, 2, 3], [1, 2, 3]]]. |
| n | [D0, D1, . . . , Dn] | n-D | A tensor with shape [D0, D1, . . . Dn]. |

**Type:** Type of the data contained within the tensor.

| Data type | Description |
|---|---|
| DT_FLOAT | 32 bits floating point. |
| DT_DOUBLE | 64 bits floating point. |
| DT_INT64 | 64 bits signed integer. |
| DT_INT32 | 32 bits signed integer. |
| DT_INT16 | 16 bits signed integer. |
| DT_INT8 | 8 bits signed integer. |
| DT_UINT8 | 8 bits unsigned integer. |
| DT_STRING | Variable length byte arrays. Each element of a Tensor is a byte array. |
| DT_BOOL | Boolean. |
| DT_COMPLEX64 | Complex number made of two 32 bits floating points: real and imaginary parts. |
| DT_QINT32 | 32 bits signed integer used in quantized Ops. |
| DT_QINT8 | 8 bits signed integer used in quantized Ops. |
| DT_QUINT8 | 8 bits unsigned integer used in quantized Ops. |

# 5 Variables

- Variables must be initialized (`tf.initialize_all_variables()`)
- Initialization is an operation, and thus must be executed within a session

# 6 Fetches

- All the ops needed to produce the values of requested tensors are run once (not once per requested tensor)

# 7 Feeds

- Temporarily replaces the output of an operation with a tensor value (act as a placeholder)
- The feed data is provided as an argument to a `session.run()` call
  `sess.run([output], feed_dict={input1:[7.], input2:[2.]})`

# 8 Operations/Functions of interest

## 8.1 CNN

- tf.nn.conv2d(input, kernel, strides, padding): apply a convolution using kernel
- tf.nn.relu(input): rectifier linear unit, every negative value is set to 0, and positive values are kept the same
- tf.sigmoid(input): returns a value in the range [0.0, 1.0]
- tf.tanh(input): returns a value in the range [-1.0, 1.0]
- tf.nn.dropout(input, keep_prob): set the output to 0.0 based on a given probability. The output is multiplied by 1/keep_prob in order to keep the expected sum unchanged
- tf.nn.max_pool(input, kernel, strides, padding): take the maximum value found within a certain kernel size
- tf.nn.avg_pool(input, kernel, strides, padding): averages out all the values at each depth found within a kernel size
- tf.nn.local_response_normalization

## 8.2 RNN

- tf.nn.rnn_cell.BasicRNNCell(num_neurons): declares a recurrent neural network cell
- tf.nn.dynamic_rnn(network, input): simulate the given RNN
- tf.nn.rnn_cell.LSTMCell(num_neurons): declares a long short-term memory neural network cell
- tf.nn.rnn_cell.GRUCell(num_neurons): declares a gated recurrent unit cell

# 9 CNN

- Used mostly to process high density matrices where the data surrounding a value is generally highly correlated with it
- Apply the convolution operator to a 2d matrix using a given kernel/filter

# 10 RNN

- Used to process sequential inputs (speech recognition, speech synthesis, connected handwriting recognition, time-series forecast, image caption generation, end-to-end translation)

# 11 See also

# 12 References


- https://www.tensorflow.org/
- https://medium.com/jim-fleming/loading-tensorflow-graphs-via-host-languages-be10fd81876f