Introduction to TensorFlow

--how low-level tensorflow API works?

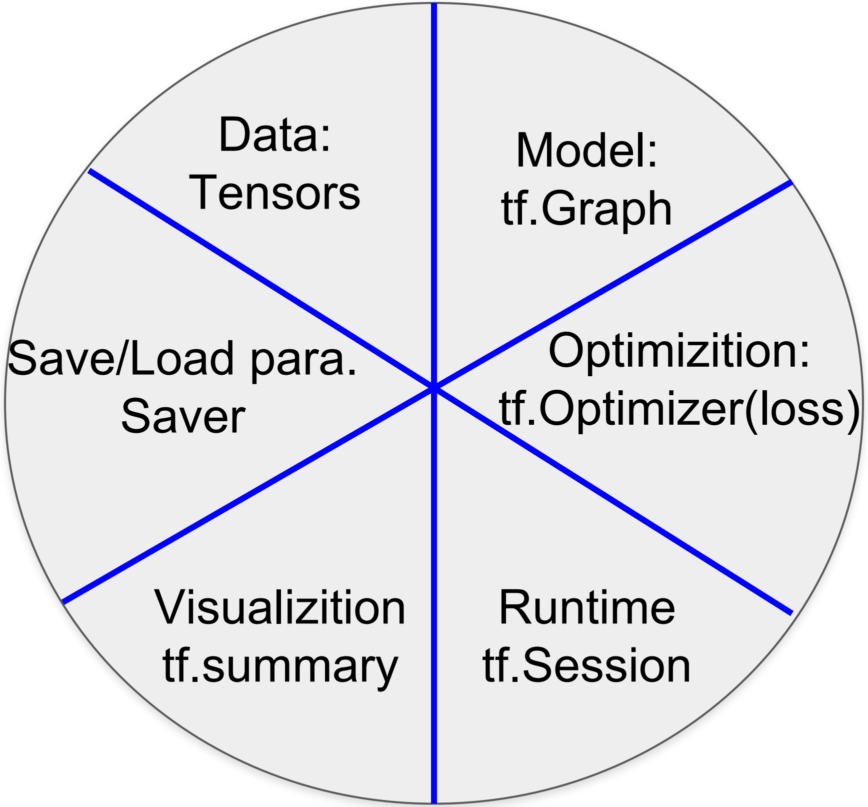
Lin Chen

23.05.2018

Institut für Photogrammetrie und GeoInformation

Leibniz
Universität
Hannover

Overview



Data:
Tensors

Model:
tf.Graph

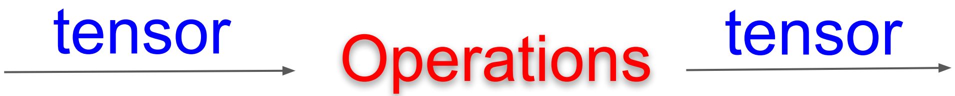
Optimizition:
tf.Optimizer(loss)

Runtime
tf.Session

Visualizition
tf.summary

Save/Load para.
Saver

Tensor ”Flow”



tensor

Operations

tensor

- Tensor: data
- Operations: processing steps (e.g. convolution) that use tensor as input and output tensor
- Basic piece of building blocks (neural network)

Tensors

Tensor: a generalization of vectors and matrices to potentially higher dimensions

Properties: datatype (float32, int32..), shape

Rank of tensors: # dimensions

- 0 Scalar (magnitude only)
- 1 Vector (magnitude and direction)
- 2 Matrix (table of numbers)
- 3 3-Tensor (cube of numbers)
- n n-Tensor (you get the idea)
- Shape partly known is allowed in tensor definition

Tensors

Variables: training parameters

Placeholders: input training mini-batch and ground truth data

Constant: e.g. superparameters

Tensors: Variables

Create [`tf.get_variable`] with name of shape, providing a name and shape

```
my_variable = tf.get_variable("my_variable", [1, 2, 3])
```

Needs a proper initializer when not restored from disk.

Tensors: Variables

Reuse

- Same scope name and set reuse=True in scope definition

```
with tf.variable_scope("model"):
    output1 = my_image_filter(input1)
with tf.variable_scope("model", reuse=True):
    output2 = my_image_filter(input2)
```

- In scope explicitly use `scope.reuse_variables()`

Tensors: Placeholder

A placeholder for a tensor that will always be fed.

```
tf.placeholder( dtype, shape=None, name=None)
```

example:

```
ex = tf.placeholder(tf.float32, shape=(1024, 1024))
```

Normally used to feed training data and ground truth (lables, depth..).

Graph

Basic types of objects:

- [Nodes] Operations (“ops”) :calculations that consume and produce tensors
- [Edges] Tensors: “values” flow through the graph. Handles to value

Computation Graph: A series of Tensorflow Operations arranged into a graph

- tf.Graph: define Graph structure
- Collections: associate a list of objects with key. tf.add_to_collection & tf.get_collection

Graph: model and optimization

Model definition:

- Placeholder for feeding data
- Build network structure and loss
- Assign a optimizer to loss

Optimizer

- Compute the gradients with `compute_gradients()`.
- Process the gradients as you wish.
- Apply the processed gradients with `apply_gradients()`.

Session

Encapsulates the state of Tensorflow runtime, and runs Tensorflow ops.

If Graph=.py file; Session = python executable

Graph are runned in sessions. [sess = tf.Session()]

- Create a session: sess = tf.Session()
- Feed data with dictionary feed_dict = {“name_placeholder”:data}
- Run operations as sess.run(“name_desired_operation”, feed_dict)

Tensorboard Summary

Save the computation graph to a TensorBoard summary file as follows:

```
writer = tf.summary.FileWriter('.')

writer.add_graph(tf.get_default_graph())
```

This Produce an event file with format:

```
events.out.tfevents.{timestamp}.{hostname}
```

Launch Tensorboard

```
tensorboard --logdir=path to write summary
```

Summary also support record tensors as histogram/scalar/image..

Save and restore trained weights

Add operations `saver = tf.train.Saver()` in graph building phase

Save variabels in a session (sess) by `saver.save(sess, model_save_dir)`

Restore saved variables in a session (sess) by `saver.restore(sess,model_save_dir)`

Referenece

[1] tensorflow programmer's guide, https://www.tensorflow.org/programmers_guide/

[2] tensorflow tutorials, https://www.tensorflow.org/tutorials/

[3] tf-lift, https://github.com/cvlab-epfl/tf-lift

[4] CIFAR-10 Dataset, https://www.cs.toronto.edu/~kriz/cifar.html