

# Deep Learning and Temporal Data Processing

Introduction to TensorFlow

---

Andrea Palazzi

June 21th, 2017

University of Modena and Reggio Emilia

**Purpose**

**Why TensorFlow**

**TensorFlow Basics**

**References**

# Purpose

---

This short tutorial aims at providing the very basics of TensorFlow.

The goal is to understand *what is* TensorFlow, *why do we need it* for deep learning, and *how does it work* from an high-level perspective.

Conversely, the next lectures will be more "hands-on": there we'll see actual code examples and we'll use our TF skills to tackle some easy task.

# Why TensorFlow

---

Open source software library for numerical computation using data flow graphs.



Why not *Theano* / *Torch* / *Caffe* / *Microsoft Cognitive Toolkit* / ... ?



- Python API
- Flexible enough for research, yet built with production use in mind
- Portable on heterogeneous systems, from mobile devices to large-scale distributed machines, and on a variety of OS (Android, Windows, iOS, ...).
- TensorBoard visualization has no rival.
- Large community and supported by Google.

There are a variety of good resources and tutorial to learn TensorFlow.

However, please keep in mind that TensorFlow is under heavy development and is constantly changing. In case of doubt, always refer to the official site:

<https://www.tensorflow.org>.

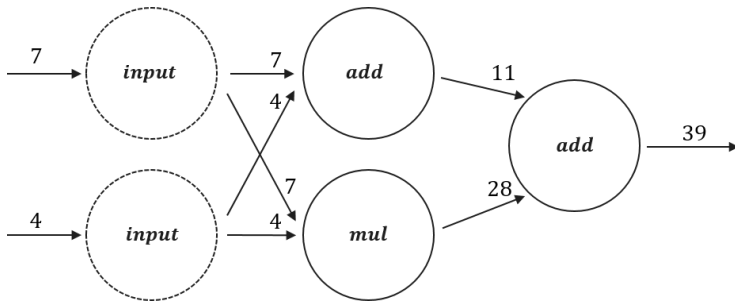


# TensorFlow Basics

---

All operations are encapsulated in a computational graph.

**Graph definition is totally separated from execution.**



Graph definition is totally separated from execution.

**So what?**

```
import tensorflow as tf

a = tf.add(7, 4)
b = tf.mul(7, 4)
result = tf.add(a, b)

print(result)
```

Graph definition is totally separated from execution.

**So what?**

```
import tensorflow as tf
```

```
a = tf.add(7, 4)
```

```
b = tf.mul(7, 4)
```

```
result = tf.add(a, b)
```

```
print(result)
```

```
# Output: "Tensor("Add_4:0", shape=(), dtype=int32)"
```

*What did you expect? :-)*

In this framework, we can think of **tensors** as **n-dimensional matrices**.

This allows to abstract over the precise data structure, e.g.:

**0-d** scalars

**1-d** vectors

**2-d** matrices

...

Tensors are represented as the **edges of the computational graph**.

In order to get a numerical result we have to **evaluate the symbolic graph**.

```
import tensorflow as tf

a = tf.add(7, 4)
b = tf.mul(7, 4)
result = tf.add(a, b)

sess = tf.Session()
print('Result: {}'.format(sess.run(result)))

# Output: "Result: 39"
```

A **Session** object encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

In order to feed value at execution time, TensorFlow provides a **placeholder** Operation.

Example:

```
x = tf.placeholder(tf.float32, shape=(1024, 1024))
y = tf.matmul(x, x)

with tf.Session() as sess:
    print(sess.run(y))    # ERROR: will fail because x was not fed.

    rand_array = np.random.rand(1024, 1024)
    print(sess.run(y, feed_dict={x: rand_array}))    # Will succeed.
```

Other advanced methods for feeding the computational graph exist, yet these are out of the scope these introductory lectures.

**Variables** are the core types we want to use when building a machine learning models. Indeed, variables' derivatives can be automatically computed by TensorFlow, which make possible easily implementing **gradient-based learning** strategies.

For this reason, learnable parameters in a machine learning models (e.g. weights and biases of a deep network) are implemented as trainable variables.

```
# Create a variable.  
w = tf.Variable(<initial-value>, name=<optional-name>)
```

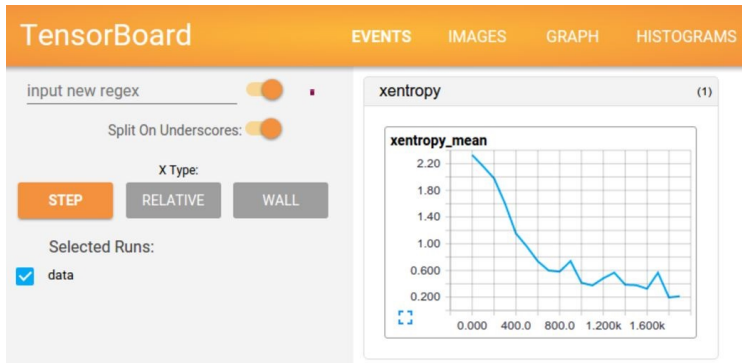


When you launch the graph, **variables have to be explicitly initialized** before you can run operations that use their value. To this purpose the convenience function `global_variables_initializer` is commonly used.

```
# Add an Op to initialize global variables.
init_op = tf.global_variables_initializer()

# Launch the graph in a session.
with tf.Session() as sess:
    # Run the Op that initializes global variables.
    sess.run(init_op)
    # ...you can now run any Op that uses variable values...
```

**TensorBoard** is a suite of visualization tools integrated with TensorFlow. You can use TensorBoard to visualize your TensorFlow graph, plot quantitative metrics about the execution of your graph, and show additional data like images that pass through it.



From a very high-level perspective, the lifecycle is the following:

- **tf.summary operations** can be attached to the nodes you'd like to collect summary data from.
- **tf.summary.FileWriter** object is in charge of writing generated logs to a certain directory `logdir`
- **TensorBoard** can be launched from command line as  
`tensorboard --logdir=<logdir>`. TensorBoard is now available through your web browser at `localhost:6006`.

Official tutorial here:

[https://www.tensorflow.org/get\\_started/summaries\\_and\\_tensorboard](https://www.tensorflow.org/get_started/summaries_and_tensorboard)

## References

---

- [1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng.

**TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.**

Software available from [tensorflow.org](https://www.tensorflow.org).