



TensorFlow

("TF")



What is TensorFlow (TF)?

TensorFlow is an open source offering from Google Brain Team.



It is a system that **processes a dataFLOW graph**, where the data that gets passed in and out of each node is a TENSOR (**multi-dim array**). In other words, it is a dataflow processor where ALL data is in the form of 'tensors'.



Why use TF?



Because CNNs involve pipelined neuron processing, where each neuron (a node in TF) processes arrays of inputs and weights (tensors in TF).



TF makes it possible to express neural networks as graphs (flexible), as opposed to a collection of chained function calls (rigid). The flexibility also allows the nodes to be processed in a variety of ways - in a CPU, GPU, cloud, mobile device..

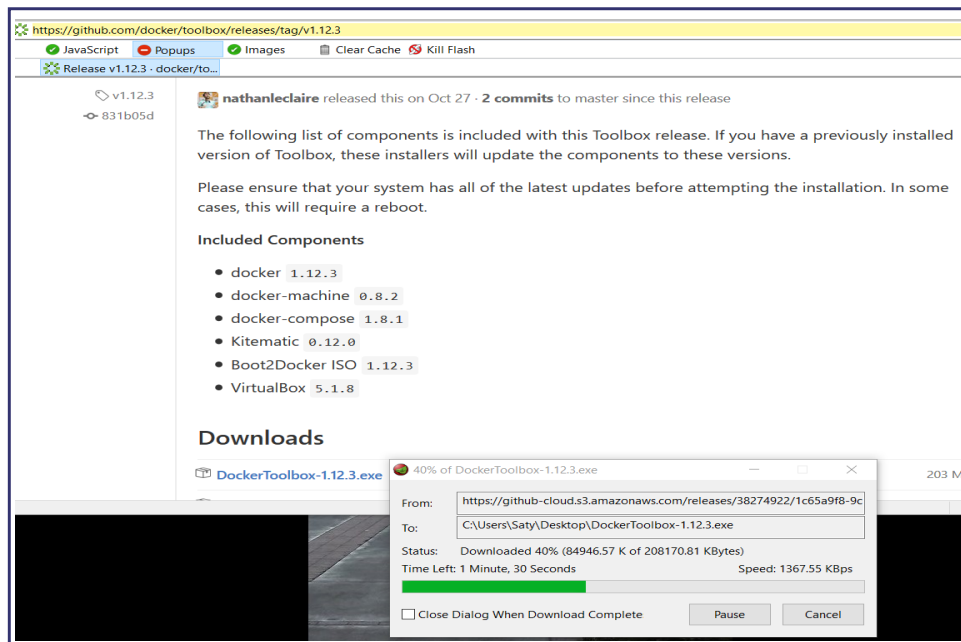


Setting up TF

You can download TF from [GitHub](#) via your native Python environment (eg. using pip install), and use it in that environment.

Or you can use [Docker](#)! Very simply, Docker is a lightweight alternative to running a VM such as VirtualBox or VMWare. Docker allows for easy downloading and installing of software inside it, using git-style pull requests. For our purposes, we'll need to install Python first, then pull in tensorflow.

Install Docker Toolbox first:



Next, [launch](#) a Docker shell (Docker Quickstart Terminal):

[illegible]

Next, verify that Docker is running properly [note - this screenshot is off a different PC compared to above!]:

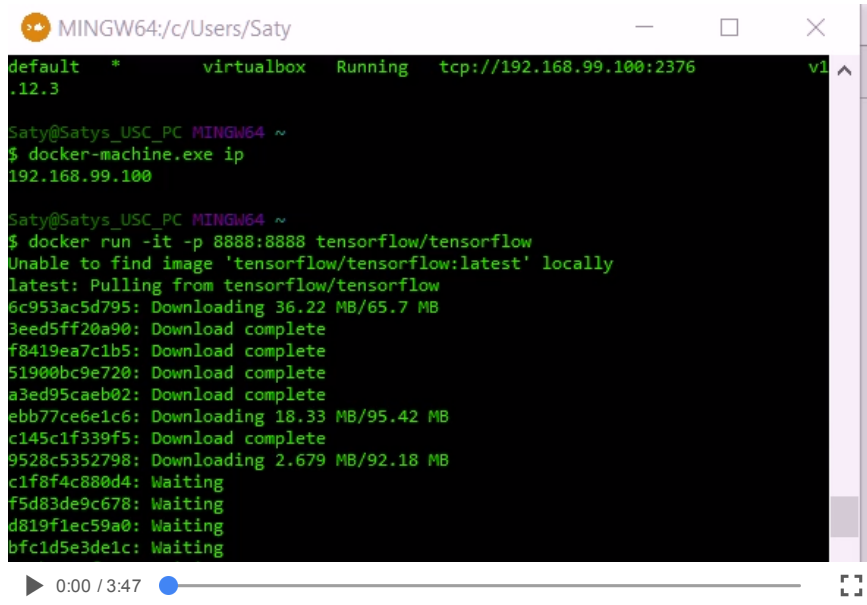
```
Saty@Satys_USC_PC MINGW64 ~
$ docker-machine.exe ls
NAME      ACTIVE   DRIVER        STATE     URL                  SWARM   DO
CKER      ERRORS
default   *        virtualbox     Running   tcp://192.168.99.100:2376   v1.12.3

Saty@Satys_USC_PC MINGW64 ~
$ docker-machine.exe ip
192.168.99.100

Saty@Satys_USC_PC MINGW64 ~
$
```

Now we can run Hello World :)



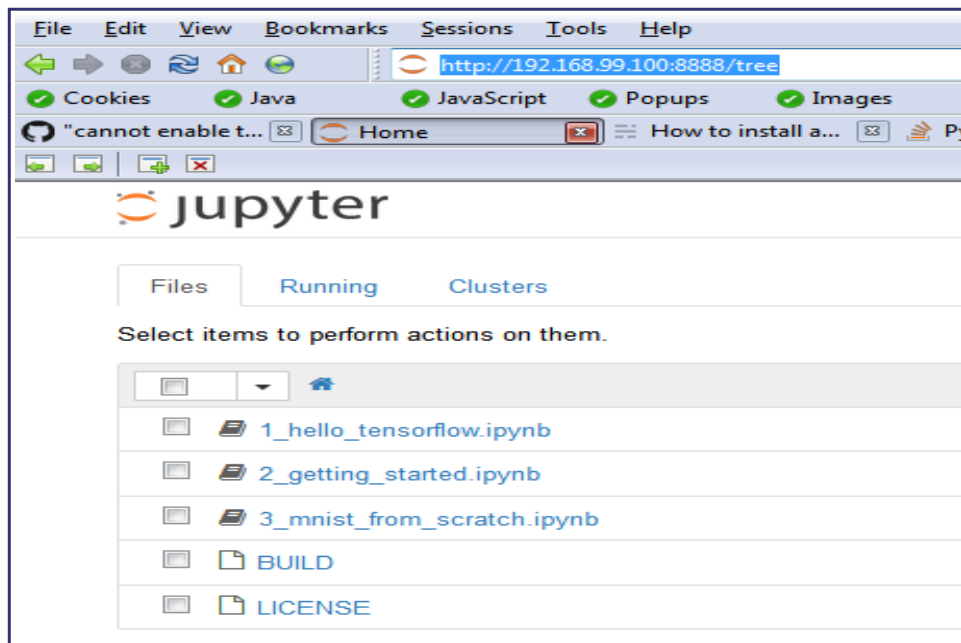


```
MINGW64:/c/Users/Saty
default * virtualbox Running tcp://192.168.99.100:2376 v1
.12.3

Saty@Satys_USC_PC MINGW64 ~
$ docker-machine.exe ip
192.168.99.100

Saty@Satys_USC_PC MINGW64 ~
$ docker run -it -p 8888:8888 tensorflow/tensorflow
Unable to find image 'tensorflow/tensorflow:latest' locally
latest: Pulling from tensorflow/tensorflow
6c953ac5d795: Downloading 36.22 MB/65.7 MB
3eed5ff20a90: Download complete
f8419ea7c1b5: Download complete
51900bc9e720: Download complete
a3ed95caeb02: Download complete
ebb77ce6e1c6: Downloading 18.33 MB/95.42 MB
c145c1f339f5: Download complete
9528c5352798: Downloading 2.679 MB/92.18 MB
c1f8f4c880d4: Waiting
f5d83de9c678: Waiting
d819f1ec59a0: Waiting
bfc1d5e3de1c: Waiting
```

Excellent! Now we can start playing with TensorFlow by visiting localhost:8888 (or <http://192.168.99.100:8888>) on our browser, and running "literate computing-style" Jupyter notebooks there.



You can even **run** Jupyter notebooks on the cloud (note - you can't edit using this interface). Eg. [here](#) is a pre-loaded example (GitHub can run this, too).

A quick example

Here is an example where we add two "tensors" (arrays of identical length/size) to obtain a resulting "tensor".

Another short example

Here we chain additions:

```
In [8]: import tensorflow as tf

with tf.Session():
    input1 = tf.constant(1.0, shape=[4])
    input2 = tf.constant(2.0, shape=[4])
    input3 = tf.constant(3.0, shape=[4])
    output = tf.add(tf.add(input1, input2), input3)
    result = output.eval()
    print(result)
```

▶ 0:00 / 0:13

⏮ ⏪ ⏩ ⏭



```
numpy("np")
```

numpy is a popular, capable module that we use with TF.

Mat mult

This is how to multiply two matrices [like we'd do in a CNN!].

A linear regression learner

This short and sweet example shows we can iteratively solve for m and c for a $y=mx+c$ line equation, given pairs of (x,y) data :)

jupyter Basic NN Last Checkpoint: 29 minutes ago (autosaved)



File Edit View Insert Cell Kernel Help

Python 2 C

+ ✂ 📄 📁 ⬆ ⬇ ⏮ ⏭ 🔄 Code CellToolbar

```
In [ ]: # https://raw.githubusercontent.com/nethsix/gentle_tensorflow/master/code/linear_regression_one_feature.py
import numpy as np
import tensorflow as tf

# Model linear regression y = Wx + b
x = tf.placeholder(tf.float32, [None, 1])
W = tf.Variable(tf.zeros([1,1]))
b = tf.Variable(tf.zeros([1]))
product = tf.matmul(x,W)
y = product + b
y_ = tf.placeholder(tf.float32, [None, 1])

# Cost function sum((y_-y)**2)
cost = tf.reduce_mean(tf.square(y_-y))

# Training using Gradient Descent to minimize cost
train_step = tf.train.GradientDescentOptimizer(0.0000001).minimize(cost)

sess = tf.Session()
```

▶ 0:00 / 0:38



You can use the above as a starting point for more regression experiments - multiple-linear, non-linear..

Resources

As you can imagine, we barely scratched the surface! Here are starting points for more exploring:

- [these](#) are Google's own TF tutorials
- [this](#) page from O'Reilly is a great introduction
- here are some [nice examples](#)
- [tflearn](#) - this add-on library makes it very easy to implement sophisticated NN algorithms.. [Here](#) is one way to install tflearn.
- an [Udacity](#) course

Also, Magenta (see [this](#) page, [this](#) one) is an experiment to get NNs to GENERATE art (including music)..