Installation

General Remarks

- There are two different variations of TensorFlow that you might wish to install, depending on whether you would like TensorFlow to run on your CPU or GPU, namely TensorFlow CPU and TensorFlow GPU. I will proceed to document both and you can choose which one you wish to install.
- If you wish to install both TensorFlow variants on your machine, ideally you should install each variant under a different (virtual) environment. If you attempt to install both TensorFlow CPU and TensorFlow GPU, without making use of virtual environments, you will either end up failing, or when we later start running code there will always be an uncertainty as to which variant is being used to execute your code.
- To ensure that we have no package conflicts and/or that we can install several different versions/variants of TensorFlow (e.g. CPU and GPU), it is generally recommended to use a virtual environment of some sort. For the purposes of this tutorial we will be creating and managing our virtual environments using Anaconda, but you are welcome to use the virtual environment manager of your choice (e.g. virtualenv).

Install Anaconda Python 3.7 (Optional)

Although having Anaconda is not a requirement in order to install and use TensorFlow, I suggest doing so, due to it's intuitive way of managing packages and setting up new virtual environments. Anaconda is a pretty useful tool, not only for working with TensorFlow, but in general for anyone working in Python, so if you haven't had a chance to work with it, now is a good chance.

Windows | Linux

- Go to https://www.anaconda.com/download/
- Download Anaconda Python 3.7 version for Windows
- Run the downloaded executable (`.exe`) file to begin the installation. See here for more details.
- (Optional) In the next step, check the box "Add Anaconda to my PATH environment variable". This will make Anaconda your default Python distribution, which should ensure that you have the same default Python distribution across all editors.

TensorFlow Installation

As mentioned in the Remarks section, there exist two generic variants of TensorFlow, which utilise different hardware on your computer to run their computationally heavy Machine Learning algorithms.

1. The simplest to install, but also in most cases the slowest in terms of performance, is TensorFlow CPU, which runs directly on the CPU of your machine.
2. Alternatively, if you own a (compatible) Nvidia graphics card, you can take advantage of the available CUDA cores to speed up the computations performed by TesnsorFlow, in which case you should follow the guidelines for installing TensorFlow GPU.

TensorFlow CPU

Getting setup with an installation of TensorFlow CPU can be done in 3 simple steps.

Create a new Conda virtual environment (Optional)

- Open a new *Anaconda/Command Prompt* window
- Type the following command:

```
conda create -n tensorflow_cpu pip python=3.6
```

- The above will create a new virtual environment with name `tensorflow_cpu`
- Now lets activate the newly created virtual environment by running the following in the *Anaconda Promt* window:

```
activate tensorflow_cpu
```

Once you have activated your virtual environment, the name of the environment should be displayed within brackets at the beggining of your cmd path specifier, e.g.:

```
(tensorflow_cpu) C:\Users\sglvladi>
```

Install TensorFlow CPU for Python

- Open a new *Anaconda/Command Prompt* window and activate the *tensorflow_cpu* environment (if you have not done so already)
- Once open, type the following on the command line:

```
pip install --ignore-installed --upgrade tensorflow==1.9
```

- Wait for the installation to finish

Test your Installation

- Open a new *Anaconda/Command Prompt* window and activate the *tensorflow_cpu* environment (if you have not done so already)
- Start a new Python interpreter session by running:

```
python
```

- Once the interpreter opens up, type:

```
>>> import tensorflow as tf
```

- If the above code shows an error, then check to make sure you have activated the *tensorflow_cpu* environment and that tensorflow_cpu was successfully installed within it in the previous step.
- Then run the following:

```
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
```

- Once the above is run, if you see a print-out similar (or identical) to the one below, it means that you could benefit from installing TensorFlow by building the sources that correspond to you specific CPU. Everything should still run as normal, just slower than if you had built TensorFlow from source.

```
2019-02-28 11:59:25.810663: I
T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:141] Your CPU
supports instructions that this TensorFlow binary was not compiled to use: AVX2
```

- Finally, for the sake of completing the test as described by TensorFlow themselves (see here), let's run the following:

```
>>> print(sess.run(hello))
b'Hello, TensorFlow!'
```

TensorFlow GPU

The installation of *TesnorFlow GPU* is slightly more involved than that of *TensorFlow CPU*, mainly due to the need of installing the relevant Graphics and CUDE drivers. There's a nice Youtube tutorial (see here), explaining how to install TensorFlow GPU. Although it describes different versions of the relevant components (including TensorFlow itself), the installation steps are generally the same with this tutorial.

Before proceeding to install TesnsorFlow GPU, you need to make sure that your system can satisfy the following requirements:

Prerequisites
Nvidia GPU (GTX 650 or newer)
CUDA Toolkit v9.0
CuDNN v7.0.5
Anaconda with Python 3.7 (Optional)

Install CUDA Toolkit

Windows | Linux

Follow this link to download and install CUDA Toolkit v9.0.

Install CUDNN

Windows | Linux

- Go to https://developer.nvidia.com/rdp/cudnn-download
- Create a user profile if needed and log in
- Select cuDNN v7.0.5 (Feb 28, 2018), for CUDA 9.0
- Download cuDNN v7.0.5 Library for Windows 10
- Extract the contents of the zip file (i.e. the folder named `cuda`) inside `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v9.0\`, where `<INSTALL_PATH>` points to the installation directory specified during the installation of the CUDA Toolkit. By default `<INSTALL_PATH>` = `C:\Program Files`.

Environment Setup

- Go to *Start* and Search “environment variables”
- Click the Environment Variables button
- Click on the `Path` system variable and select edit
- Add the following paths:
    - `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin`
    - `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v9.0\libnvvp`
    - `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v9.0\extras\CUPTI\libx64`
    - `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v9.0\cuda\bin`

Update your GPU drivers (Optional)

If during the installation of the CUDA Toolkit (see Install CUDA Toolkit) you selected the *Express Installation* option, then your GPU drivers will have been overwritten by those that come bundled with the CUDA toolkit. These drivers are typically NOT the latest drivers and, thus, you may wish to updte your drivers.

- Go to http://www.nvidia.com/Download/index.aspx
- Select your GPU version to download
- Install the driver for your chosen OS

Create a new Conda virtual environment

- Open a new *Anaconda/Command Prompt* window
- Type the following command:

```
conda create -n tensorflow_gpu pip python=3.6
```

- The above will create a new virtual environment with name `tensorflow_gpu`
- Now lets activate the newly created virtual environment by running the following in the *Anaconda Promt* window:

```
activate tensorflow_gpu
```

Once you have activated your virtual environment, the name of the environment should be displayed within brackets at the beggining of your cmd path specifier, e.g.:

```
(tensorflow_gpu) C:\Users\sglvladi>
```

Install TensorFlow GPU for Python

- Open a new *Anaconda/Command Prompt* window and activate the *tensorflow_gpu* environment (if you have not done so already)
- Once open, type the following on the command line:

```
pip install --ignore-installed --upgrade tensorflow-gpu==1.9
```

- Wait for the installation to finish

Test your Installation

- Open a new *Anaconda/Command Prompt* window and activate the *tensorflow_gpu* environment (if you have not done so already)
- Start a new Python interpreter session by running:

```
python
```

- Once the interpreter opens up, type:

```
>>> import tensorflow as tf
```

- If the above code shows an error, then check to make sure you have activated the *tensorflow_gpu* environment and that tensorflow_gpu was successfully installed within it in the previous step.
- Then run the following:

```
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
```

- Once the above is run, you should see a print-out similar (but not identical) to the one bellow:

```
2019-02-28 06:56:43.617192: I
T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU
supports instructions that this TensorFlow binary was not compiled to use: AVX2
2019-02-28 06:56:43.792865: I
T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:1356] Found
device 0 with properties:
name: GeForce GTX 1080 major: 6 minor: 1 memoryClockRate(GHz): 1.7335
pciBusID: 0000:01:00.0
totalMemory: 8.00GiB freeMemory: 6.61GiB
2019-02-28 06:56:43.799610: I
T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:1435] Adding
visible gpu devices: 0
2019-02-28 06:56:44.338771: I
T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:923] Device
interconnect StreamExecutor with strength 1 edge matrix:
2019-02-28 06:56:44.348418: I
T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:929]      0
2019-02-28 06:56:44.351039: I
T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:942] 0:   N
2019-02-28 06:56:44.352873: I
T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:1053] Created
TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 6387 MB memory) ->
physical GPU (device: 0, name: GeForce GTX 1080, pci bus id: 0000:01:00.0, compute
capability: 6.1)
```

- Finally, for the sake of completing the test as described by TensorFlow themselves (see here), let's run the following:

```
>>> print(sess.run(hello))
b'Hello, TensorFlow!'
```

TensorFlow Models Installation

Now that you have installed TensorFlow, it is time to install the models used by TensorFlow to do its magic.

Install Prerequisites

Building on the assumption that you have just created your new virtual environment (whether that's *tensorflow_cpu*,`tensorflow_gpu` or whatever other name you might have used), there are some packages which need to be installed before installing the models.

Prerequisite packages	
Name	**Tutorial version-build**
pillow	5.4.1-py36hdc69c19_0
lxml	4.3.1-py36h1350720_0
jupyter	1.0.0-py36_7

Prerequisite packages	
Name	**Tutorial version-build**
matplotlib	3.0.2-py36hc8f65d3_0
opencv	3.4.2-py36h40b0b35_0

The packages can be installed using `conda` by running:

```
conda install <package_name>(=<version>), <package_name>(=<version>), ..., <package_name>(=
<version>)
```

where `<package_name>` can be replaced with the name of the package, and optionally the package version can be specified by adding the optional specifier `=<version>` after `<package_name>`. For example, to simply install all packages at their latest versions you can run:

```
conda install pillow, lxml, jupyter, matplotlib, opencv
```

Alternatively, if you don't want to use Anaconda you can install the packages using `pip`:

```
pip install <package_name>(==<version>) <package_name>(==<version>) ... <package_name>(==
<version>)
```

but you will need to install `opencv-python` instead of `opencv`.

Downloading the TensorFlow Models

- Create a new folder under a path of your choice and name it `TensorFlow`. (e.g. `C:\Users\sglvladi\Documents\TensorFlow`).
- From your *Anaconda/Command Prompt* `cd` into the `TensorFlow` directory.
- To download the models you can either use Git to clone the TensorFlow Models repo inside the `TensorFlow` folder, or you can simply download it as a ZIP and extract it's contents inside the `TensorFlow` folder. To keep things consistent, in the latter case you will have to rename the extracted folder `models-master` to `models`. [1]
- You should now have a single folder named `models` under your `TensorFlow` folder, which contains another 4 folders as such:

```
TensorFlow
└─ models
    ├── official
    ├── research
    ├── samples
    └── tutorials
```

[1] The latest repo commit when writing this tutorial is 4b566d4.

Protobuf Installation/Compilation

The Tensorflow Object Detection API uses Protobufs to configure model and training parameters. Before the framework can be used, the Protobuf libraries must be downloaded and compiled.

This should be done as follows:

- Head to the protoc releases page
- Download the latest `*-win32.zip` release (e.g. `protoc-3.5.1-win32.zip`)
- Create a folder in `C:\Program Files` and name it `Google Protobuf` .
- Extract the contents of the downloaded `*-win32.zip` , inside `C:\Program Files\Google Protobuf`
- Add `C:\Program Files\Google Protobuf\bin` to your `Path` environment variable (see Environment Setup)
- In a new *Anaconda/Command Prompt* [2], `cd` into `TensorFlow/models/research/` directory and run the following command:

```
# From within TensorFlow/models/research/
protoc object_detection/protos/*.proto --python_out=.
```

> **Important**
>
> If you are on Windows and using Protobuf 3.5 or later, the multi-file selection wildcard (i.e `*.proto`) will not work but you can do one of the following:

Windows Powershell | **Command Prompt**

```
# From within TensorFlow/models/research/
for /f %i in ('dir /b object_detection\protos\*.proto') do protoc
object_detection\protos\%i --python_out=.
```

[2] NOTE: You MUST open a new *Anaconda/Command Prompt* for the changes in the environment variables to take effect.

Adding necessary Environment Variables

1. As `Tensorflow\models\research\object_detection` is the core package for object detection, it's convenient to add the specific folder to our environmental variables.

Linux **Windows**

The following folder must be added to your `PYTHONPATH` environment variable (See Environment Setup):

- `<PATH_TO_TF>\TensorFlow\models\research\object_detection`

Note

The above can also be achieved, in both Linux and Windows environments, by running the following from `Tensorflow\models\research`:

```
# From within TensorFlow/models/research/
python setup.py build
python setup.py install
```

The above commands essentially build and install the `object_detection` Python package.

DRAWBACK: The above commands need to be run everytime there is a change/update of the `object_detection` package.

2. For whatever reason, some of the TensorFlow packages that are required to perform object detection, do not come pre-installed with our tensorflow installation.

Linux **Windows**

The only way that I found works best, is to simply add the following folders to your `PYTHONPATH` environment variable (See also Environment Setup):

- `<PATH_TO_TF>\TensorFlow\models\research`
- `<PATH_TO_TF>\TensorFlow\models\research\slim`

where, in both cases, `<PATH_TO_TF>` replaces the absolute path to your `TesnorFlow` folder. (e.g. `<PATH_TO_TF>` = `C:\Users\sglvladi\Documents` if `TensorFlow` resides within your `Documents` folder)

COCO API installation (Optional)

The `pycocotools` package should be installed if you are interested in using COCO evaluation metrics.

Windows | Linux

Run the following command to install `pycocotools` with Windows support:

```
pip install git+https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI
```

Note that, according to the package's instructions, Visual C++ 2015 build tools must be installed and on your path. If they are not, make sure to install them from here.

The default metrics are based on those used in Pascal VOC evaluation. To use the COCO object detection metrics add *metrics_set: "coco_detection_metrics"* to the *eval_config* message in the config file. To use the COCO instance segmentation metrics add *metrics_set: "coco_mask_metrics"* to the *eval_config* message in the config file.

Test your Installation

- Open a new *Anaconda/Command Prompt* window and activate the *tensorflow_gpu* environment (if you have not done so already)
- `cd` into `TensorFlow\models\research\object_detection` and run the following command:

```
# From within TensorFlow/models/research/object_detection
jupyter notebook
```

- This should start a new `jupyter notebook` server on your machine and you should be redirected to a new tab of your default browser.
- Once there, simply follow sentdex's Youtube video to ensure that everything is running smoothly.