**Installation Instructions for the Python3/Numpy/Tensorflow tutorials:**

1. This assumes you have installed Anaconda

<https://www.anaconda.com/download/>

1. Once you have that downloaded Anaconda please follow the instructions here:

<https://github.com/sachin80/nps_tutorials/blob/master/README.md>

**Installation Instructions for the Tensorflow Object Detection Library:**

1. This assumes you have successfully followed the steps above.
2. Go to your local directory. In my case this is called development
3. Active your virtual environment with:

conda activate NAMEOFYOURVIRTUALENVIRONMENT

I’m typing these commands into the command line interface called “Anaconda Prompt”.

1. Clone the Tensorflow model repository with the following command:

git clone <https://github.com/tensorflow/models.git>

1. Follow the object detection installation instructions up to COCO API installation:

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md>

Make sure that all the **Dependencies** are installed.

**COCO API:**

1. Clone the COCO API with following command:

git clone <https://github.com/cocodataset/cocoapi.git>

1. Change your directory to the COCO PythonAPI directory:

cd cocoapi\PythonAPI

1. From Chrome or another browser download mingw32 to your local C: drive. [http://www.mingw.org/](http://www.mingw.org/%20)

Download the mingw-get-setup.exe**.** Run the executable**.** Make sure to check that (“apply changes”) all the available packages are being downloaded when prompted by the mingw32 installation manager.

1. Once you have downloaded mingw32 make sure to update your Windows PATH variable. Go to System -> Advanced system settings -> Environment Variables and add:

C:\MinGW\msys\1.0\bin

1. From the Start Menu launch the Anaconda3 Navigator. Once that is open, launch the Spyder code editor. From the my Spyder Code editor change local directory to:

C:\Users\srdc\Desktop\development\cocoapi\PythonAPI

Edit the setup.py file such that you comment (using an #) on the following line:

#extra\_compile\_args=['-Wno-cpp', '-Wno-unused-function', '-std=c99'],

Add this line below the commented line above:

extra\_compile\_args={'gcc': ['/Qstd=c99']},

Save the file and close Spyder.

1. Back to the Anaconda Prompt. Check that you’re still in the Coco Python API directory using the pwd command. Should get something like this:

/c/Users/srdc/Desktop/development/cocoapi/PythonAPI

1. Run the following command:

make

Your output after running the make command should be something like:

python setup.py build\_ext --inplace

running build\_ext

skipping 'pycocotools\\_mask.c' Cython extension (up-to-date)

copying build\lib.win-amd64-3.6\pycocotools\\_mask.cp36-win\_amd64.pyd -> pycocotools

rm -rf build

If this doesn’t work then please follow the directions here:

<https://github.com/philferriere/cocoapi>

1. Copy the folder pycocotools located in the cocoapi\PythonAPI to your Tensorflow model research folder C:\Users\srdc\Desktop\development\models\research

**Google ProtoBuf:**

1. Download Google Protocol Buffers for Windows*.* Note: Download [version 3.4.0 for Windows](https://github.com/google/protobuf/releases/tag/v3.4.0), as later versions may give you a “Permission Error” when trying to compile.

Use this link <https://github.com/protocolbuffers/protobuf/releases/tag/v3.4.0>, then download “protoc-3.4.0-win32.zip”.

1. Extract the Protobuf download. I did it to C:\Users\srdc
2. Execute the protobuf compile within the command prompt.

Open a new Anaconda prompt. Go to your models\research directory. For me, the full path is C:\Users\srdc\Desktop\development\models\research

Execute the protobuf compiler using the location of the “protoc.exe” file. You must provide the full path to the protobuf executable. For me this whole command is:

“C:\Users\srdc\protoc-3.4.0-win32\bin\protoc.exe” object\_detection/protos/\*.proto --python\_out=.

Note: you can test if this has worked by going to the object\_detection/protos folder, and if there are .py files, you’ve successfully completed the compilation of your .proto files!

Also, instead of the full path to the executable you may change the environment variable

first to:

C:\Users\srdc\protoc-3.4.0-win32\bin

Then you can execute the protoc with:

protoc object\_detection/protos/\*.proto --python\_out=.

Add Paths:

1. Add the libraries to your Python path.

Go to System -> Advanced system settings -> Environment Variables -> New, and add a variable with the name PYTHON\_PATH and these values

* YOURPATHTO\models\research
* YOURPATHTO\models\research\slim
* YOURPATHTO\models\research\object\_detection

**Run it!**

1. From models\research\object\_detection type the following command:

jupyter notebook

Once your Jupyter Notebook is opened in your browser click on the file: **object\_detection\_tutorial.ipynb**

**In this file add the following lines to the cell 1 right after “Imports” :**

**import matplotlib**

**import matplotlib.pyplot as plt**

Make sure you run all the cells. You should get the following image at the end of running all the cells:



**Installation Instructions for Tensorflow-GPU:**

To install Tensorflow-GPU please activate your conda virtual environment and run the following command from the Anaconda Prompt:

**conda install tensorflow-gpu**

**This command will download the needed software for the GPU including Nvidia CUDA and cuDNN.**

To check that your installation is successful please run the following commands from the Anaconda prompt:

python

>>> import tensorflow as tf

>>> sess = tf.Session(config=tf.ConfigProto(log\_device\_placement=True))

After running the above command you should something similar to the following output:

2019-01-30 15:16:41.437206: I tensorflow/core/platform/cpu\_feature\_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2

2019-01-30 15:16:41.969287: I tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1432] Found device 0 with properties:

name: GeForce GTX 1080 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.6705

pciBusID: 0000:2d:00.0

totalMemory: 11.00GiB freeMemory: 9.10GiB

2019-01-30 15:16:41.979583: I tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1511] Adding visible gpu devices: 0

2019-01-30 15:16:43.732913: I tensorflow/core/common\_runtime/gpu/gpu\_device.cc:982] Device interconnect StreamExecutor with strength 1 edge matrix:

2019-01-30 15:16:43.738334: I tensorflow/core/common\_runtime/gpu/gpu\_device.cc:988] 0

2019-01-30 15:16:43.741472: I tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1001] 0: N

2019-01-30 15:16:43.747016: I tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 8786 MB memory) -> physical GPU (device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:2d:00.0, compute capability: 6.1)

Device mapping:

/job:localhost/replica:0/task:0/device:GPU:0 -> device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:2d:00.0, compute capability: 6.1

2019-01-30 15:16:43.770033: I tensorflow/core/common\_runtime/direct\_session.cc:307] Device mapping:

/job:localhost/replica:0/task:0/device:GPU:0 -> device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:2d:00.0, compute capability: 6.1

**References:**

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md>

<https://github.com/cocodataset/cocoapi/issues/51>

<https://medium.com/@marklabinski/installing-tensorflow-object-detection-api-on-windows-10-7a4eb83e1e7b>