Introduction to machine learning. A crash course using Tensorflow and main concepts explained.

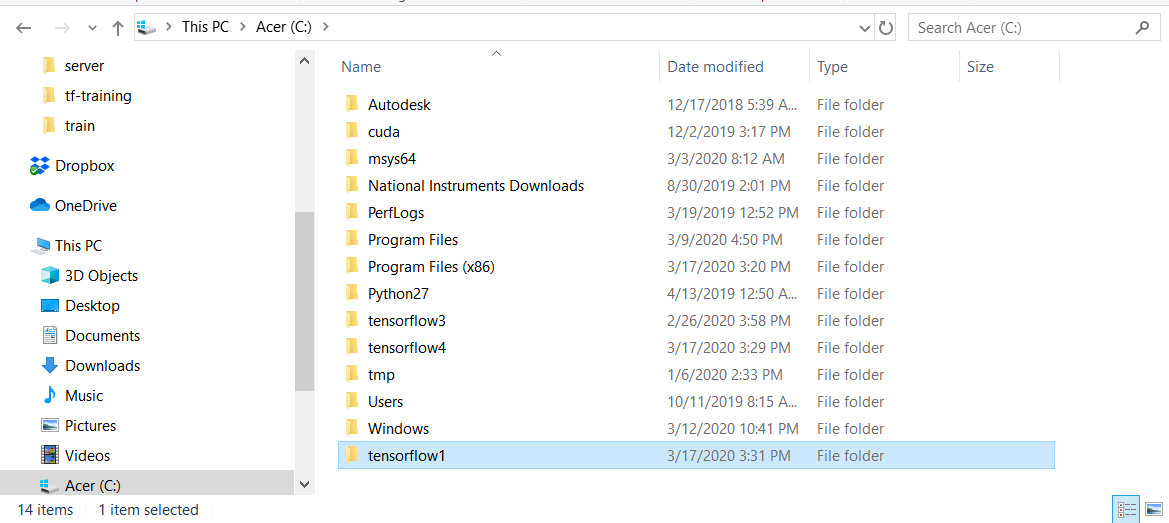
<https://developers.google.com/machine-learning/crash-course>

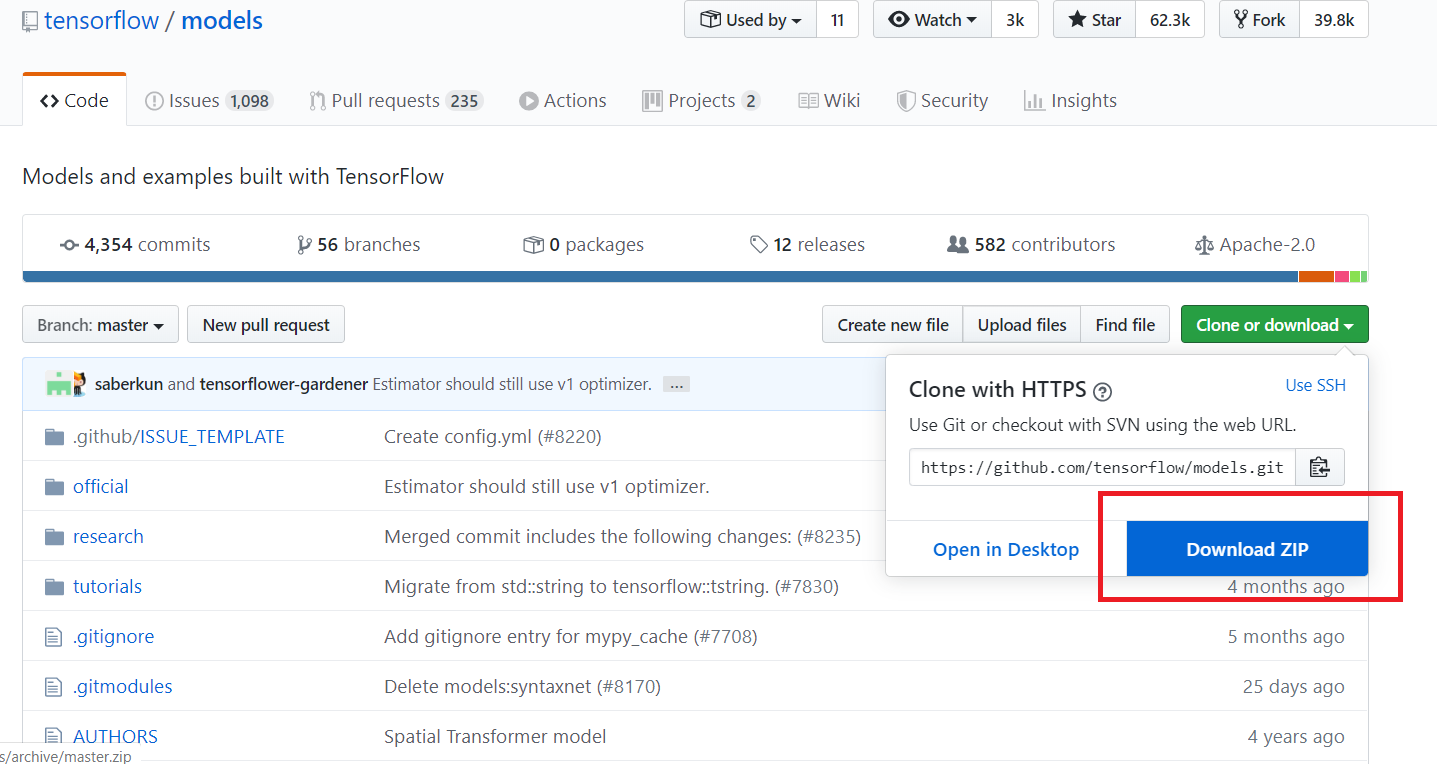
If you don’t want to use tensorflow to learn, just purely python, this book is good

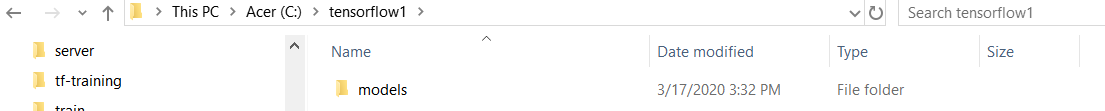
<https://www.kickstarter.com/projects/sentdex/neural-networks-from-scratch-in-python>

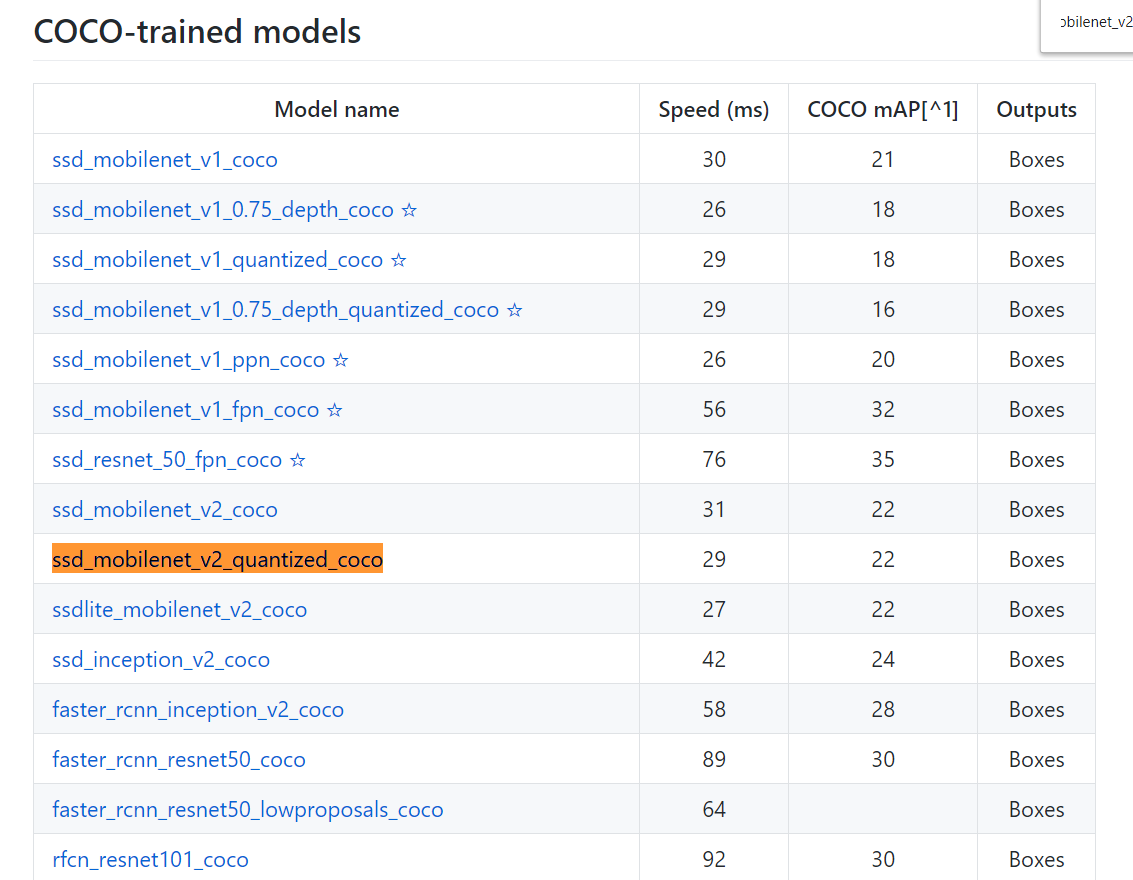
This tutorial is based off of:

<https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10>

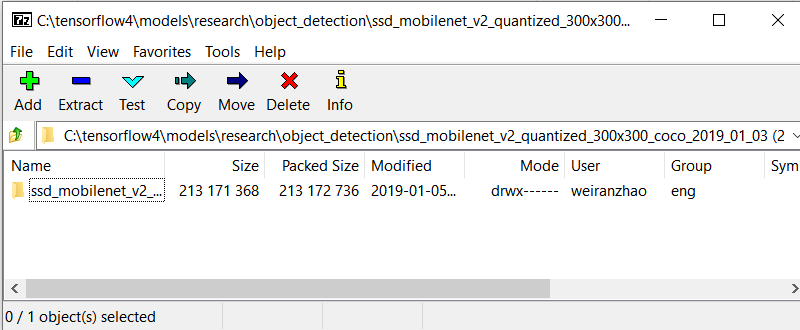
1. Install
   1. SET UP WINDOWS MACHINE TRAINING ENVIRONMENT
      1. Create a new folder in **C:/tensorflow1**
      2. Download <https://github.com/tensorflow/models>

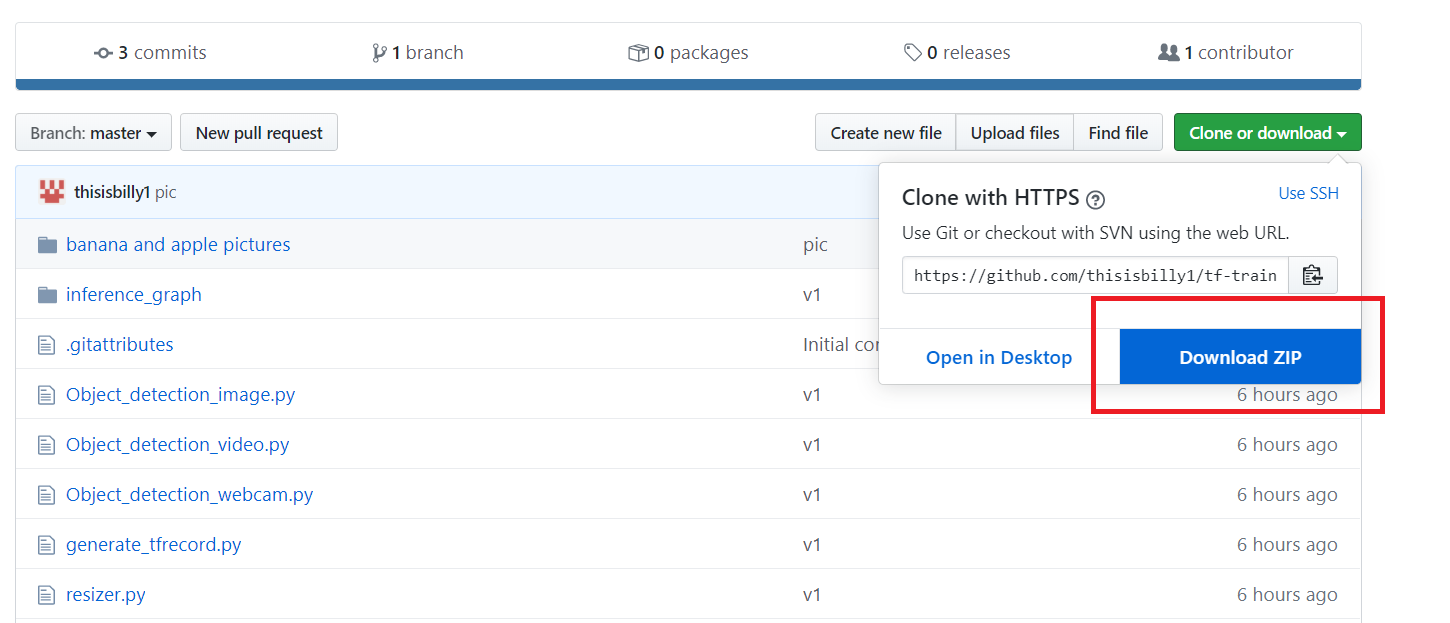


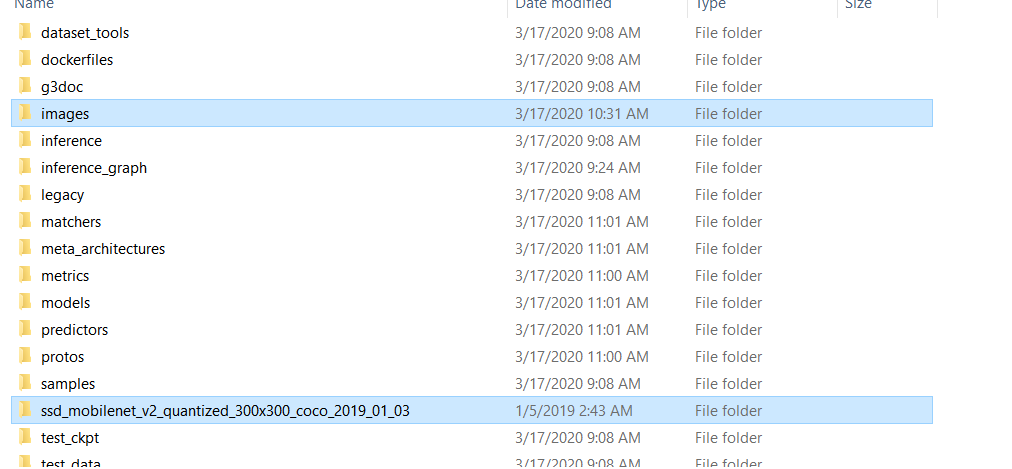
* + 1. Extract it to it in **C:/tensorflow1**
    2. Rename from “**models-master**” to “**models**”
    3. Go to model zoo <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>
       1. Download “[ssd\_mobilenet\_v2\_quantized\_coco](http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_quantized_300x300_coco_2019_01_03.tar.gz)”
          1. We need this one in order to convert it to a TFlite model later. A TFlite model can run on a raspberry pi.

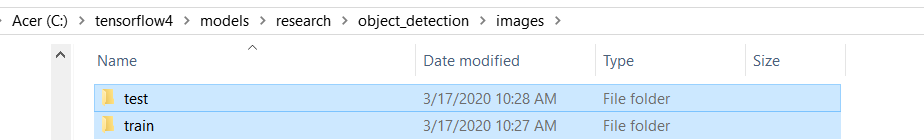


* + 1. Move that file into the folder into “**C:\tensorflow1\models\research\object\_detection**”
       1. Extract the contents using 7zip



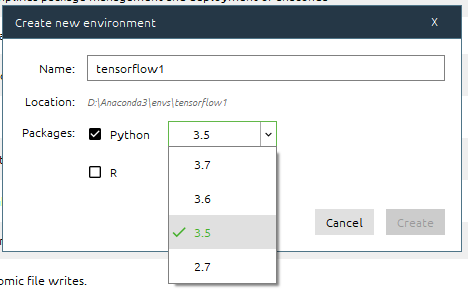
* + 1. Go to <https://github.com/thisisbilly1/tf-training>
       1. Download and paste into the \object\_detection folder
    2. Create a folder named “**images**” in **\object\_detection** folder. Your folder should look like this:



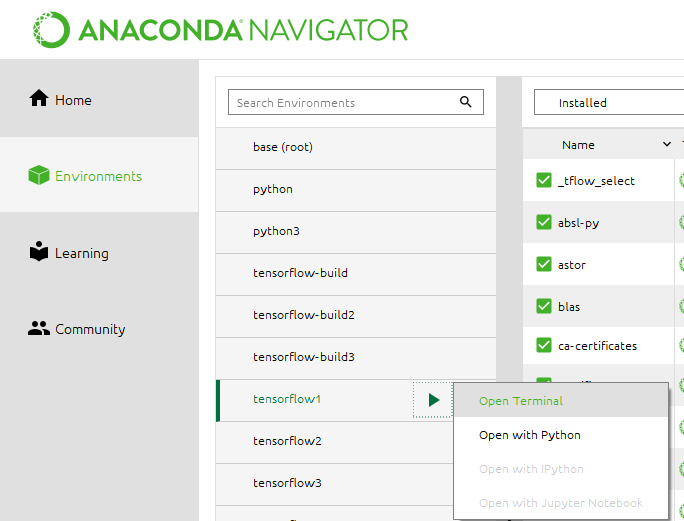
* + - 1. Create a “**test**” folder in the **\images** folder
      2. Create a “**train**” folder in the **\images** folder
  1. SET UP ANACONDA VIRTUAL ENVIRONMENT
     1. Install anaconda

https://repo.anaconda.com/archive/Anaconda3-2020.02-Windows-x86\_64.exe

* + 1. Create new env named “tensorflow1” – use python 3.5



* + 1. Click open terminal – this is where we will install the libraries that we’ll need



**python -m pip install --upgrade pip**

**pip install --ignore-installed --upgrade tensorflow-gpu**

* + - * 1. **test that TF is installed by importing it**

import tensorflow as tf

tf.\_\_version\_\_

* + - * 1. if it does not work, try using:

pip install tensorflow==1.14.0

* + - 1. install other required libraries

**conda install -c anaconda protobuf**

**pip install pillow**

**pip install lxml**

**pip install Cython**

**pip install contextlib2**

**pip install jupyter**

**pip install matplotlib**

**pip install pandas**

**pip install opencv-python**

* + 1. Set a path that points to \models, \models\research and \models\research\slim

**set PYTHONPATH=C:\tensorflow1\models;C:\tensorflow1\models\research;C:\tensorflow1\models\research\slim**

This command has to be done every time the tensorflow1 virtual environment is closed.

* 1. Compile protobufs
     1. Protobuf files are used by TF to configure model and training parameters. We need to compile these

**cd C:\tensorflow1\models\research**

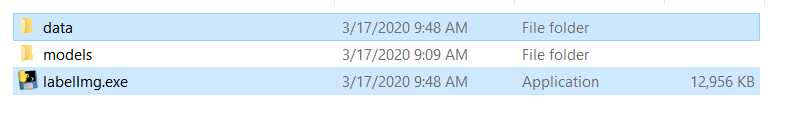
**protoc --python\_out=. .\object\_detection\protos\anchor\_generator.proto .\object\_detection\protos\argmax\_matcher.proto .\object\_detection\protos\bipartite\_matcher.proto .\object\_detection\protos\box\_coder.proto .\object\_detection\protos\box\_predictor.proto .\object\_detection\protos\eval.proto .\object\_detection\protos\faster\_rcnn.proto .\object\_detection\protos\faster\_rcnn\_box\_coder.proto .\object\_detection\protos\grid\_anchor\_generator.proto .\object\_detection\protos\hyperparams.proto .\object\_detection\protos\image\_resizer.proto .\object\_detection\protos\input\_reader.proto .\object\_detection\protos\losses.proto .\object\_detection\protos\matcher.proto .\object\_detection\protos\mean\_stddev\_box\_coder.proto .\object\_detection\protos\model.proto .\object\_detection\protos\optimizer.proto .\object\_detection\protos\pipeline.proto .\object\_detection\protos\post\_processing.proto .\object\_detection\protos\preprocessor.proto .\object\_detection\protos\region\_similarity\_calculator.proto .\object\_detection\protos\square\_box\_coder.proto .\object\_detection\protos\ssd.proto .\object\_detection\protos\ssd\_anchor\_generator.proto .\object\_detection\protos\string\_int\_label\_map.proto .\object\_detection\protos\train.proto .\object\_detection\protos\keypoint\_box\_coder.proto** **.\object\_detection\protos\multiscale\_anchor\_generator.proto .\object\_detection\protos\graph\_rewriter.proto .\object\_detection\protos\calibration.proto .\object\_detection\protos\flexible\_grid\_anchor\_generator.proto**

* + 1. If you get no errors, it ran successfully. Check in \object\_detection\protos for the “\*\_pb2.py” files
  1. Run setup.py

**cd C:\tensorflow1\models\research directory**

**python setup.py build**

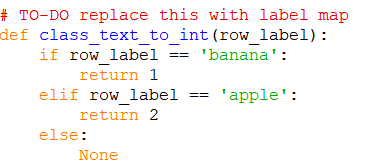
**python setup.py install**

1. Training
   1. Gather pictures
      1. Label them using labelImg.
         1. <https://www.dropbox.com/s/tq7zfrcwl44vxan/windows_v1.6.0.zip?dl=1>
         2. Put it in **C:\tensorflow1** folder
      2. Split the images 80% in the train folder, 20% in the test folder
      3. Open the image labeler and start labelling the images
   2. Generate training data

**cd C:\tensorflow1\models\research\object\_detection**

**python xml\_to\_csv.py**

* + - 1. check in your images folder. There should now be 2 .csv files
  1. edit “**generate\_tfrecord.py**” in **\object\_detection** folder
     1. change the function “**class\_text\_to\_int**” to fit your label map
     2. example:



* 1. Generate the TF record files

**python generate\_tfrecord.py --csv\_input=images\train\_labels.csv --image\_dir=images\train --output\_path=train.record**

**python generate\_tfrecord.py --csv\_input=images\test\_labels.csv --image\_dir=images\test --output\_path=test.record**

* 1. Create label map create a file called “**labelmap.pbtxt**” in **object\_detection/training**
     1. This tells the trainer what each object is this is related to how we set up our “class\_text\_to\_int” function.
     2. For our example:

item {

id: 1

name: 'banana'

}

item {

id: 2

name: 'apple'

}

* 1. Configure training
     1. Copy the “**ssd\_mobilenet\_v2\_quantized\_300x300\_coco.config**” from the **\object\_detection\samples\configs** folder into **\object\_detection\training**
     2. Line 9: change the **num\_classses** to however many objects you are detecting
     3. Line 156: change **fine\_tune\_checkpoint** to "C:/tensorflow1/models/research/object\_detection/ssd\_mobilenet\_v2\_quantized\_300x300\_coco\_2019\_01\_03/model.ckpt"
     4. Line 175: change **input\_path** to "C:/tensorflow1/models/research/object\_detection/train.record"
     5. Line 177: change **label\_map\_path** to "C:/tensorflow1/models/research/object\_detection/training/labelmap.pbtxt"
     6. Line 181: change the **num\_examples** to the number of images that you have in your \images\test folder
     7. Line 189: change **input\_path** to "C:/tensorflow1/models/research/object\_detection/test.record"
     8. Line 191: change **label\_map\_path** to "C:/tensorflow1/models/research/object\_detection/training/labelmap.pbtxt"

1. TRAINING run this command to start. Make sure you are in **\object\_detection**

**python model\_main.py --model\_dir=training --pipeline\_config\_path=training/ssd\_mobilenet\_v2\_quantized\_300x300\_coco.config**

1. Viewing training progress
   1. Use tensorboard
   2. Open another terminal

**cd C:\tensorflow1\models\research\object\_detection**

**tensorboard --logdir=training**

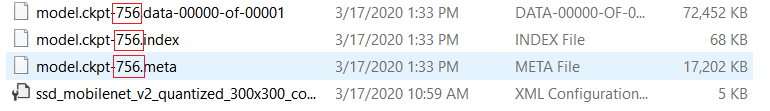
* 1. copy the link and paste it into FireFox

1. export inference graph

**python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path training/ssd\_mobilenet\_v2\_quantized\_300x300\_coco.config --trained\_checkpoint\_prefix training/model.ckpt-XXXX --output\_directory inference\_graph**

replace XXXX with the biggest number in your \training folder

example:



1. convert the model to a TF Lite model
   1. <https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi#step-2-build-tensorflow-from-source>
2. convert the tflite model to be compatible to run with a TPU
   1. install a Linux virtual machine on your windows PC, or have a Linux PC ready (Debian 6.0+)
   2. <https://coral.ai/docs/edgetpu/compiler/>

SET UP PI <https://www.raspberrypi.org/documentation/installation/installing-images/README.md>

1. Download the pi imager for windows
   1. <https://www.raspberrypi.org/downloads/>
   2. Click on “Operating System” and click the recommended setting



* 1. Plug in your SD card and then follow this tutorial:

https://youtu.be/aimSGOAUI8Y?t=82