**OBJECT DETECTION REFLECTION REPORT**

**WORK ACCOMPLISHED**

**GENERATING TF\_RECORD**

The following steps were followed respectively

1. Cloning the labelImg repository from its Github repository

To get LabeImg repository [click here](https://github.com/tzutalin/labelImg)

1. Downloading Anaconda, open as administrator creating a new virtual environment and activate it

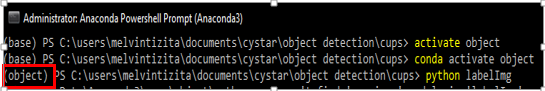
To create a new virtual environment, use the command:

*conda create –n ‘name of virtual environment’*

To activate the virtual environment

*conda activate ‘name of virtual environment’*

to confirm whether it has worked, the terminal path should start with the name of the virtual environment



1. Install the following requirements. Type the following commands in Anaconda terminal

*conda install pyqt=5*

Move to the directory of the cloned repository ‘labelImg’. Then type the following command one after the other.

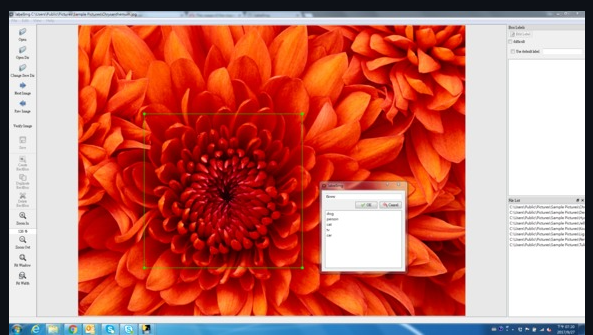
*pyrcc5 –o libs/resources.py resources.qrc*

*pip install lxml*

*pip install pyqt5*

*python labelImg.py*

the annotation tools will pop up on the screen



1. Use the annotation tools to annotate the objects in the photos
2. Generating csv files from xml files

To get the xml\_to\_csv.py and generate\_tfrecord.py file [click here.](https://github.com/General-Manu/object-detection.git)

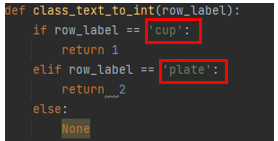
1. Move 10% of the pictures and their corresponding csv files to the test folder and the rest to the train folder

In the terminal, change directory to the location of the xml\_to\_csv.py file

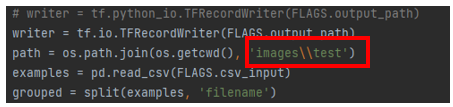
Type the following command

*Python xml\_to\_csv.py*

1. Open and change the label in the generate\_tfrecord.py file to suit your label



Replace the above value with the name of your object(s)



Configure the path in the code

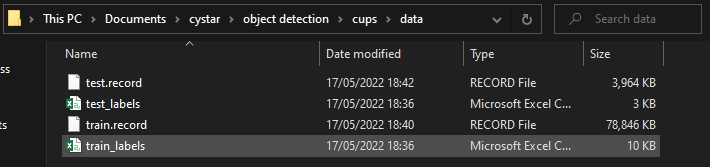
Run the following command in anaconda terminal

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

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

NB: change the generate\_tfrecord path in the code before executing commands

After this step the tf records will be generated in the data folder.



Xml files and the generated tf\_records

# **TRAINING THE MODEL**

1. Install requirements ready for training the custom model

Use pip command to install the following

*TensorFlow==1.15 lxml pillow matplotlib jupyter contextlib2 cython and tf\_slim*

1. Clone the models repository from the link below

To get the lick to the TensorFlow repository [click here](https://github.com/tensorflow/models.git)

1. Download protoc 3.4 (only this version works properly)

Extract the zip and open the bin folder. Move protoc.exe into the ‘research’ folder of the cloned repository

From the ‘research’ directory run this command to compile protoc:

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

If we want to run the object\_detection scripts on a local machine, we need to go to the ‘research’ directory and run these two commands:

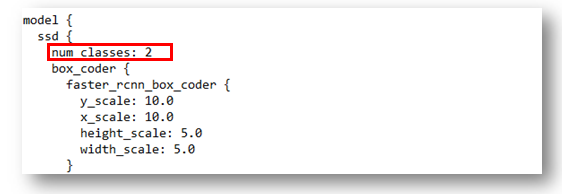
*Python setup.py build*

*Python setup.py install*

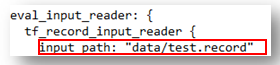
1. Download COCO trained model and config file from the link below

To get the COCO trianined model [click here](https://github.com/tensorflow/models/research/object_detection/g3doc/tf1_detection_zoo.md)

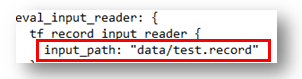
1. Edit the config file to configure the correct paths and names of the files



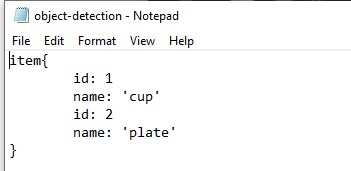
Change to the number of classes you have



change the input paths to your own



1. Create a pbtxt file and place it in the training folder



Type the above commands in the pbtxt file according to your number of classes

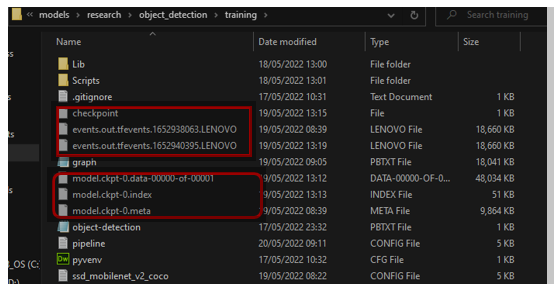
1. Change directory to the research folder and type the command:

*set PYTHONPATH=$PYTHONPATH:pwd:pwd/slim*

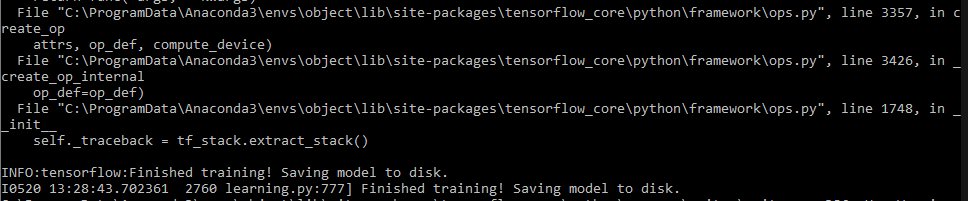
#used to set python path

1. Change directory into object\_detection folder and type the following command

*python train.py --train\_dir=training/ --pipeline\_config\_path=training/ssd\_mobilenet\_v2\_coco.config –logtostderr*

**

The model files and checkpoint files will be generated during training of the model.



This will be the output generated after the model has been trained

#training complete

1. For the last step run the command below to generate the new\_graph folder

*python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path training/ssdlite\_mobilenet\_v2\_coco\_2018\_05\_09.config --trained\_checkpoint\_prefix training/model.ckpt-0 --output\_directory new\_graph*

the number after the model index is the number of your last generated model.data file.

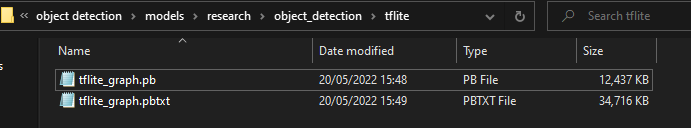
**GENERATING THE TFLITE FOLDER**

This process aims at converting the inference graph file to a Tensorflow lite (.tflite) file used on mobile.

1. Create a Tensorflow frozen graph with compatible ops that can be used with Tensorflow lite. Run the command below from the object\_detection folder

python export\_tflite\_ssd\_graph.py \   
--pipeline\_config\_path=training/ssd\_mobilenet\_v2\_coco.config \   
--trained\_checkpoint\_prefix=training/model.ckpt-50233 \   
--output\_directory=tflite \  
--add\_postprocessing\_op=true

A folder with two files will be generated as shown below.

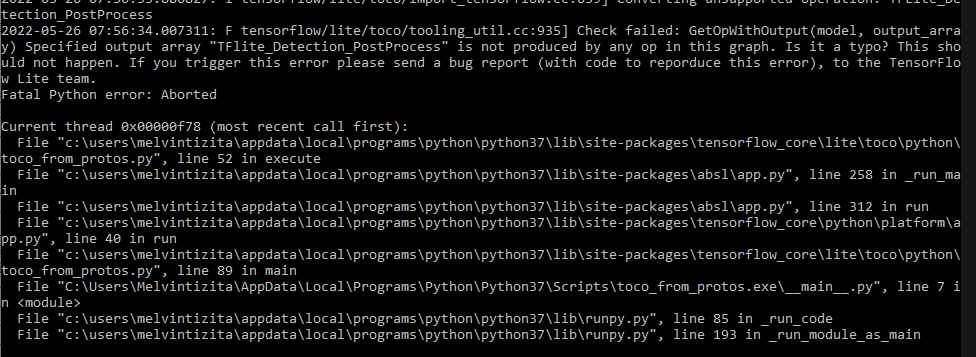


1. Next we run this command below

tflite\_convert \  
--graph\_def\_file=tflite/tflite\_graph.pb \   
--output\_file=tflite/detect.tflite \   
--output\_format=TFLITE \  
--input\_shapes=1,300,300,3 \  
--input\_arrays=normalized\_input\_image\_tensor \  
--output\_arrays='TFLite\_Detection\_PostProcess','TFLite\_Detection\_PostProcess:1','TFLite\_Detection\_PostProcess:2','TFLite\_Detection\_PostProcess:3' \  
--inference\_type=QUANTIZED\_UINT8 \  
--mean\_values=128 \  
--std\_dev\_values=127 \  
--change\_concat\_input\_ranges=false \  
--allow\_custom\_ops

it should generate a file called detect.tflite in the tflite folder. The size of this file is usually below 5MB, it can be lower depending on the size of your inference graph.

**NB: the above code may generate the following errors**



The error displayed above indicates that the specified output array is not produced by any op.

The alternative is to get the correct output using summarize graph tool, building it with Docker and bazel.

To install bazel in anaconda environment, use the command below

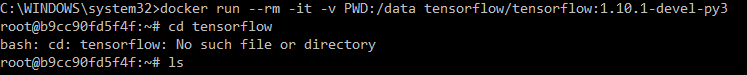
conda install –c conda-forge bazel

To download Docker for windows [click here](https://docs.docker.com/desktop/windows/install/)

To build with Docker, use the following command

Docker run --rm -it -v PWD:/data tensorflow/tensorflow:1.10.1-devel-py3

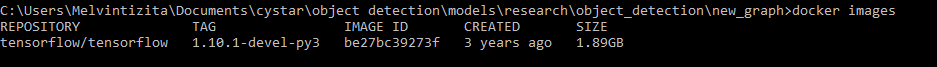
This is the expected output. It changes the system root to a Docker root.



Open a new command-line prompt window and run the following commands

Docker images: this shows you all the available images in the Docker platform.

Docker ps: shows the details of the image being used



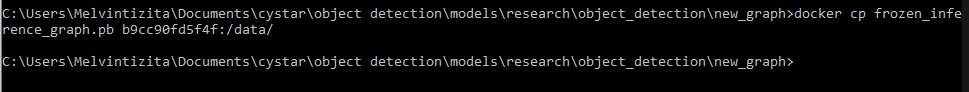
Use the commands above to identify your image and the image ID.

Next we need to move the frozen\_inference\_graph to the data folder in the Docker environment. Use the command below

Docker cp frozen\_inference\_graph.pb b9cc90fd5f4f:/data/

You must be in the file’s directory for it to work. That’s why we opened another command prompt window.

It move on to the root if command is correct as shown below



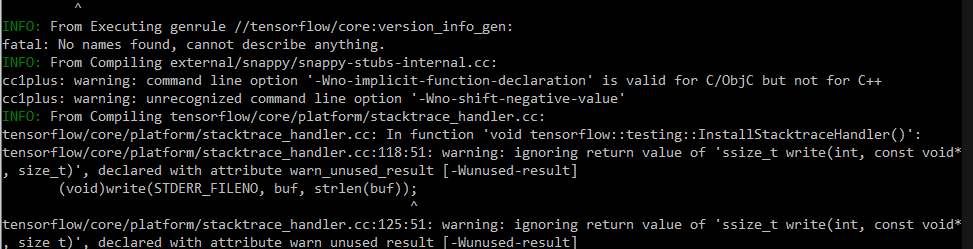
Now move back to the previous window that has Docker activated and type the commands below

cd /tensorflow

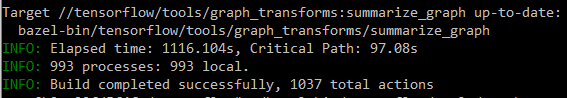
bazel build tensorflow/tools/graph\_transforms:summarize\_graph

./bazel-bin/tensorflow/tools/graph\_transforms/summarize\_graph --in\_graph=/data/mobilenet\_v1\_1.0\_224\_frozen.pb

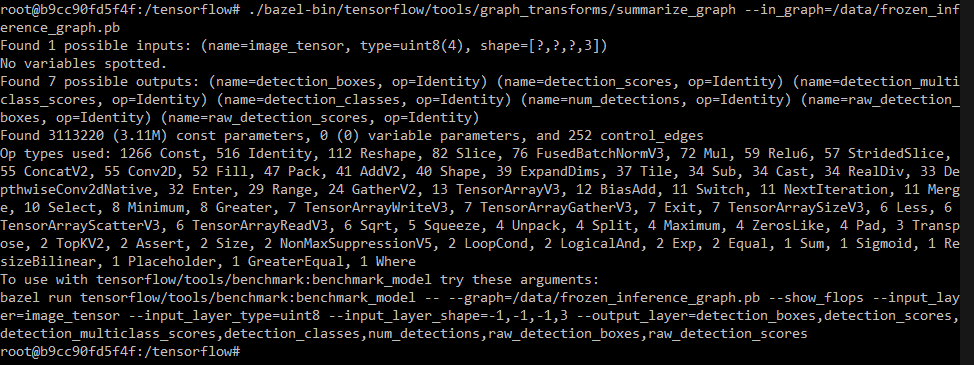
The following output is expected



When it is done, it will show the following output

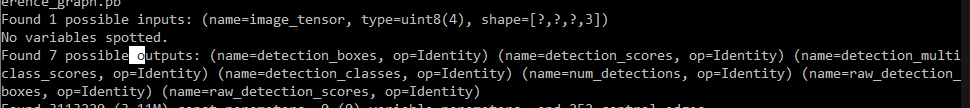


This shows that the build was successful. Run the last command to find the output arrays



The expected output will be as shown above.

Output arrays



**CHALLENGES EXPEREIENCED**

The errors that came up were mainly because of the following reasons

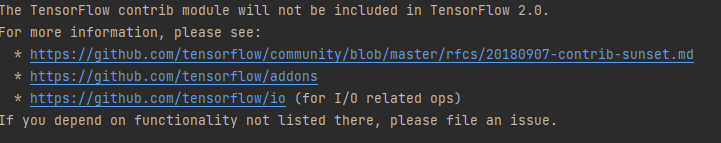
* Incompatible requirement versions e.g. TensorFlow 1.15 and python 3.8, 3.9
* Miss-guided directories
* Unconfirmed or misconfigured paths in codes
* Broken packages e.g. matplotlib, pylint,
* Outdated command syntax or non-compatible command syntax e.g. additional special character %, & and ‘’ = and == may give an error.

**FIXING THE CHALENGES**

The above challenges were fixed by:

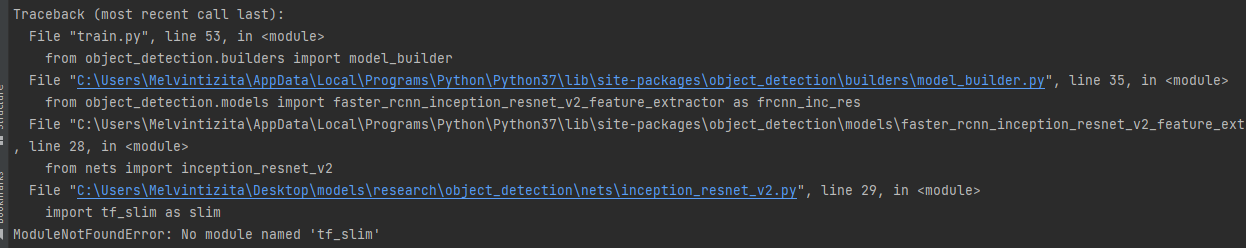
* Uninstalling and reinstalling the packages in question to recover avoid broken packages e.g. kiwisolver, pip, python, matplotlib, pylint
* Counter checking code and paths to be configured
* Using the specific versions stated e.g. TensorFlow 1.15, Python 3.7, numpy 1.19.5
* Forcing installation using the command ‘*conda install -c conda-forge 'name of app'* ‘
* Appending **‘.\’** to unrecognized user trusted commands e.g.  **.\ ‘command’**

**ERRORS EXPERIENCED**



**Fix:** Installing TensorFlow version 1.15

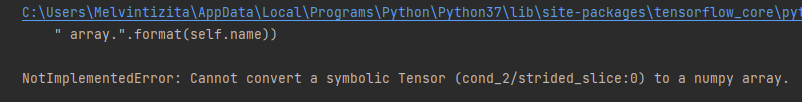
Command: *pip install Tensorflow = = 1.15*



**Fix:** install tf\_slim module, if you already had, uninstall and reinstall

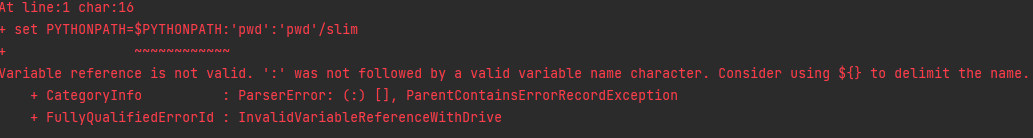
Command: *pip install tf\_slim*

*pip uninstall tf \_slim*

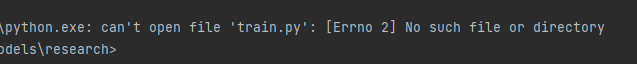


**Fix:** install numpy version 1.19.5

Command: *pip install numpy==1.19.5*



**Fix:** check syntax of command of the system python version, may be issues with parenthesis, brackets, curly braces.



**Fix:** go back to the research directory and make sure the ‘train.py’ file is in the correct directory

If it is not there, copy the file from the ‘slim’ folder to the research folder.



**Fix:** check whether the command entered is valid and correct. There may be an error in terms of syntax, keywords or compatibility of requirement versions

**REFERENCE**

Ben, G. (2020, July 16) *How to create a custom Object Detector with TensorFlow 2020*

<https://www.youtube.com/watch?v=C5-SEZ_IvaM&t=1068s>

Ben, G. (2020, July 22) *How to Train a Custom Model for Object Detection (Local and Google Colab!)* <https://www.youtube.com/watch?v=_gGI91BmIdk&t=610s>

Toure, N. (2019, June 8) Convert a TensorFlow frozen graph to a TensorFlow lite (tflite) file (Part 3) <https://teyou21.medium.com/convert-a-tensorflow-frozen-graph-to-a-tflite-file-part-3-1ccdb3874c4a>