**For Custom Dataset training**

* Make a folder ‘Drone’ and inside ‘Drone’ folder make three folders ‘data’ ‘images’ and ‘training’. Inside ‘images’ make ‘train’ and ‘test’. Copy the ‘object\_detection’ from ‘research’ folder n paste it inside ‘Drone’
* Put the images in the Train and Test Folder in 80-20 split
* First have to annotate the dataset images using LabelImage. This is used to draw labels around the object in each Image. Download it from here [https://github.com/tzutalin/labelImg](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa1VNeTdvR3ZMQkgxSndQQnpJeWJTUG0weHhFUXxBQ3Jtc0tuV090Tmk5VUpyV1RMWjh4OTBtRHdLVUdHSFRlOXhkek1MLTVQX1EwT1AwV3gyNnlMUjR2aUFFYW4xLUdQU2xCUy1mOFVBUHRIX2Z5b19jcDFSQi1EUGdEUk56MGtqS1c3UERfUW15QkVFY0FoTEY1WQ&q=https%3A%2F%2Fgithub.com%2Ftzutalin%2FlabelImg&v=C5-SEZ_IvaM) or available in Requirements folder
* Running Procedure:
* Be in the labelImg Directory
* Make sure lxml is installed
* pip install pyqt5==5.14.1
* pyrcc5 -o libs/resources.py resources.qrc
* python labelImg.py
* Select the ‘Open\_dir’ and go to the folder where the images are saved and ‘Change\_Save\_dir’. Save it in the same folder from which you are reading from.
* The previous step creates xml files which need to be converted into csv files so that we can then convert those into TFRecord Files.
* Now we convert the xml files generated into csv
* Download the python file to convert xml to csv from here <https://github.com/datitran/raccoon_dataset/blob/master/xml_to_csv.py> or available in Requirements folder
* Have to do an edit in the main function i.e change the directory to point to the folder images/train or images/test where the xml files of each images are

def main(): # this is the part we change to work with our setup

for directory in ['train','test']:

image\_path = os.path.join(os.getcwd(), 'images/{}'.format(directory))

xml\_df = xml\_to\_csv(image\_path)

xml\_df.to\_csv('data/{}\_labels.csv'.format(directory), index=None)

print('Successfully converted xml to csv.')

* Make sure the indentation is correct and also the output of the xml\_to\_csv file is stored in the ‘data’ folder that we have created earlier.
* Run: python xml\_to\_csv.py from the correct directory
* Now we need create tf\_record using “generate\_tfrecord.py”. This is what TensorFlow uses to help detect out objects.
* Dowload the python file from here: <https://github.com/datitran/raccoon_dataset/blob/master/generate_tfrecord.py> or available in Requirements folder
* Edit the function class text\_to\_int so that the label name matches your label from Labellmage. Here we changed the row\_label to ‘Drone’
* From the correct directory(inside drone), run these commands to generate tfrecords
* python generate\_tfrecord.py --csv\_input=data/test\_labels.csv --output\_path=data/test.record --image\_dir=images/test
* python generate\_tfrecord.py --csv\_input=data/train\_labels.csv --output\_path=data/train.record --image\_dir=images/train
* Possible error: the Labels ie the annotated name(Drone n drone are not same)should all be the same else there is error

return tf.train.Feature(int64\_list=tf.train.Int64List(value=value))

TypeError: None has type NoneType, but expected one of: int, long

* Go to TensorFlow/'s Detection Model Zoo and download one of the models. The models

have different speed and performance measurements so pick the one that best suits

* your model/application. Download and extract the model from here <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md> or available in Requirements folder
* From the 'samples' folder under object \_detection, grab the matching config model for the chosen model
* In the config file we need to edit the number of classes to the amount we are actually using (just 1 in this case).
* The line that looks like fine\_tune\_checkpoint:

"PATH\_TO\_BE\_CONFIGURED/model.ckpt", needs to be changed so that the

PATH part is replaced with the path of the folder that has the model. Just copy

the name of the folder and paste it here.

* In the train input reader function, the input \_path line should be data/train .record and label map path should be training/object-detection.pbtxt

(we will make the .pbxt soon)

* In the eval input reader function, tf record input reader should be data/test.record and label map path should be training/object-detection.pbtxt (just like the previous function)
* If when later training the model you receive memory errors, change the batch

size value in the config file to be smaller.

* Move the config file into “training”
* Go into the training folder and create (this follows for Windows) a new text document named object-detection. This will give the name for our labels in the output bounding box.
* Inside the document you need to add this:

item{

id: 1

name: ‘Drone’

}

* Change the name to whatever you labeled your images as
* Click File->Save as-> and change the file type to "All Files' and change the extension name to .pbtxt
* Copy only ‘nets’, ‘deployment’ and ‘utils’ folders from ‘object\_detection’ and drop it in ‘drone’
* Inside the object\_detection directory, go into legacy' and move train.py into the main directory
* Inside the document you need to add something like this: Training our model(pipeline)
* From object detection run

python train.py --train\_dir=training/ -- pipeline\_config\_path=training/ssd\_mobilenet\_v1\_coco.config –logtostderr

* Error: i) No module named nets/ No module named deployment:
* Go into research n copy slim/nets and slim/deployment and drop it in our main folder

ii)NotImplementedError: Cannot convert a symbolic Tensor (cond\_2/strided\_slice:0) to a numpy array

* pip install numpy==1.19.5

---------------------------------------------------------Training starts---------------------------------------------------------------

* Now take the latest checkpoint number here it is “model.ckpt-6874” and run the following
* To make the graph:

python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path training/ssd\_mobilenet\_v1\_coco.config --trained\_checkpoint\_prefix training/model.ckpt-6874 --output\_directory new\_graph