Method 1 (Long Method Using Tensorflow Model Folder)

Prerequisites For OD

1. Tensorflow
2. TensorBoard
3. Python
4. Matplotlib
5. COCOAPI
6. PROTOBUF

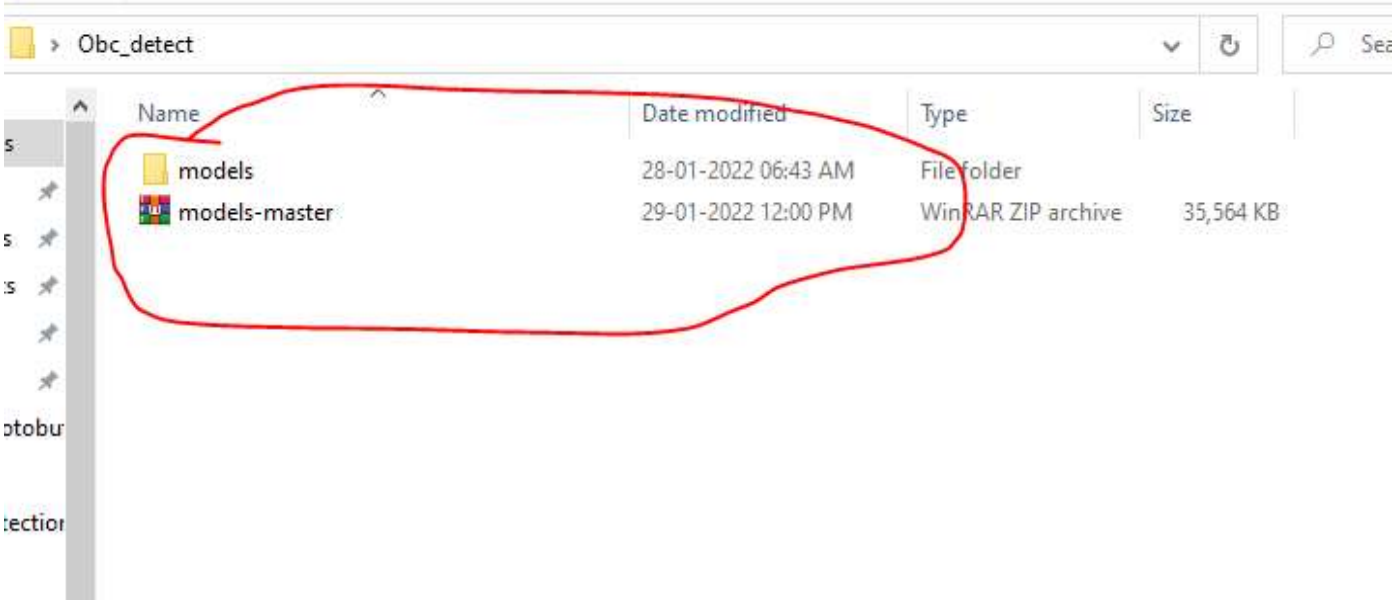
Step 1 (Download tensorflow models Folder from github)

Downloading **tensorflow models** from github . There are two ways of doing this, one is by using git and another by manually downloading it :

- Using git: This is the easiest way of downloading the Tensorflow Object detection API from the repository but you need to have git installed in the system. Open the command prompt and type this command git clone https://github.com/tensorflow/models (https://github.com/tensorflow/models)

- Downloading Manually: To manually download the API, go to this link and click on the code button(in green colour). You can see the download zip option, click on that you will have a compressed file. Now you need to extract the files.

After getting this Folder in your PC, rename the folder from models-master to models.



Step 2 (Installing all dependencies)

Installing all dependencies that you need for object detection.

- pip install tensorflow
- pip install pillow Cython lxml jupyter matplotlib contextlib2 tf_slim

Step 3 (Download and set in ENV Protocol Buffers (Protobuf))

Now we need to download Protocol Buffers (Protobuf) which are Google's language-neutral.

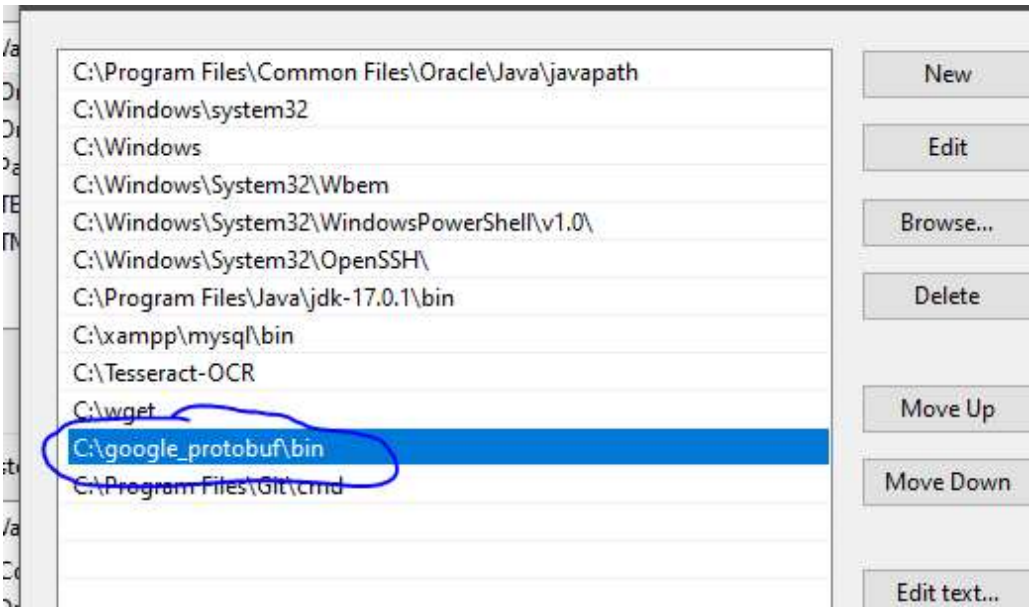
Download - https://github.com/protocolbuffers/protobuf/releases (https://github.com/protocolbuffers/protobuf/releases)

After download extract. After extracting i have 2 folder i.e bin, include and 1 redme file .

So here

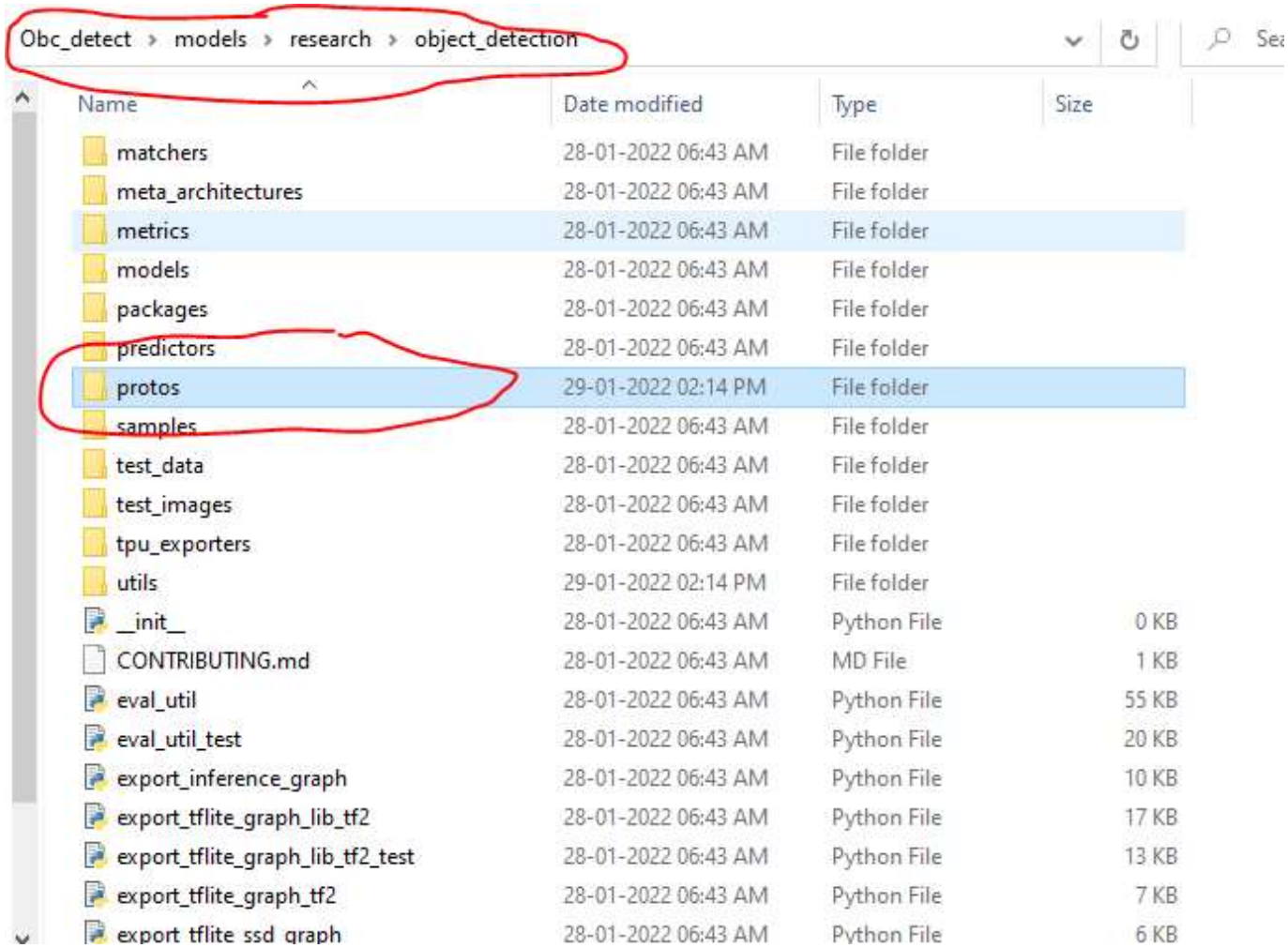
1st step is to copy all file and create new folder in your c drive and paste it

Then add bin file in **Enviorment Variable** by copying the path like C:\google_protobuf\bin.



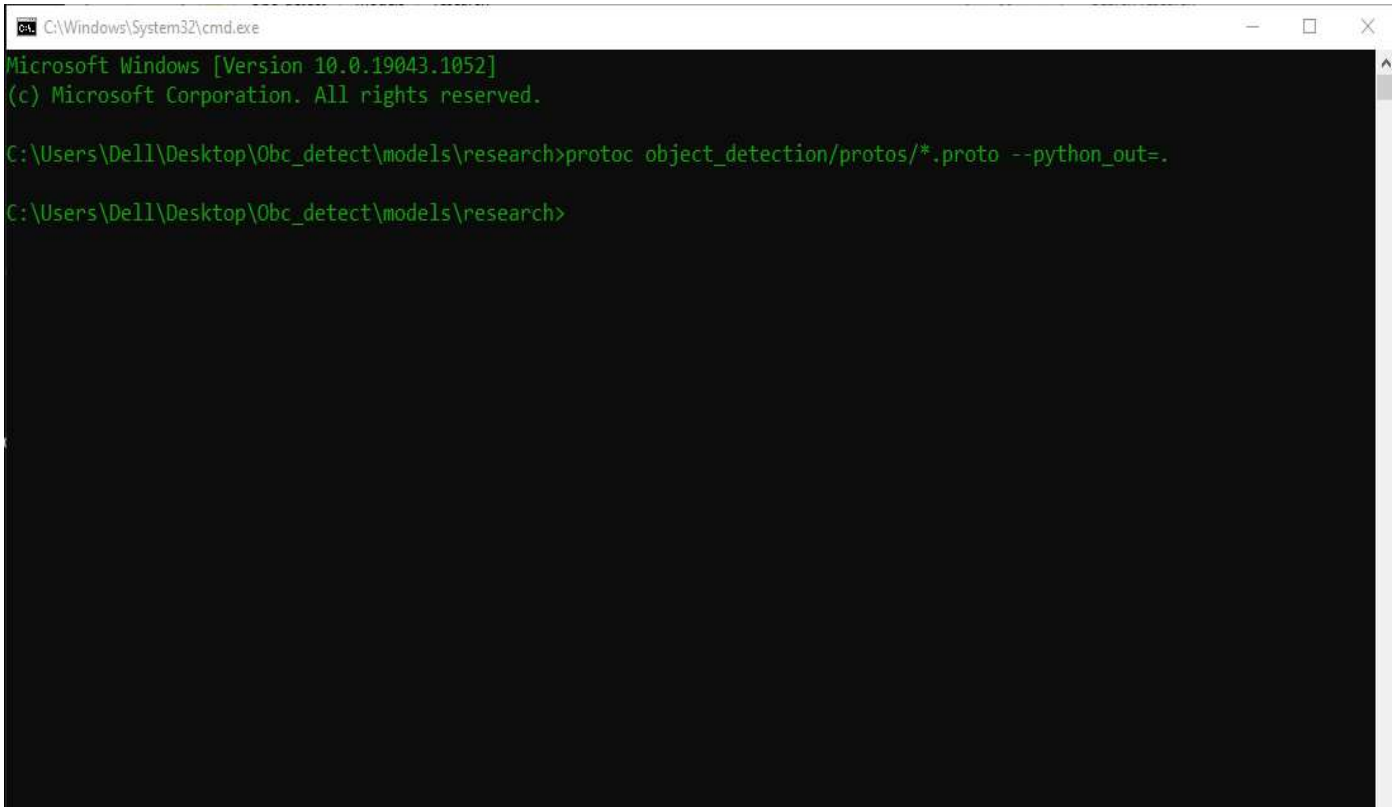
step 4 (convert protoc file to python)

In this step we converted protoc file that inside models/research/object_detection/protoc to python so



Open Command prompt

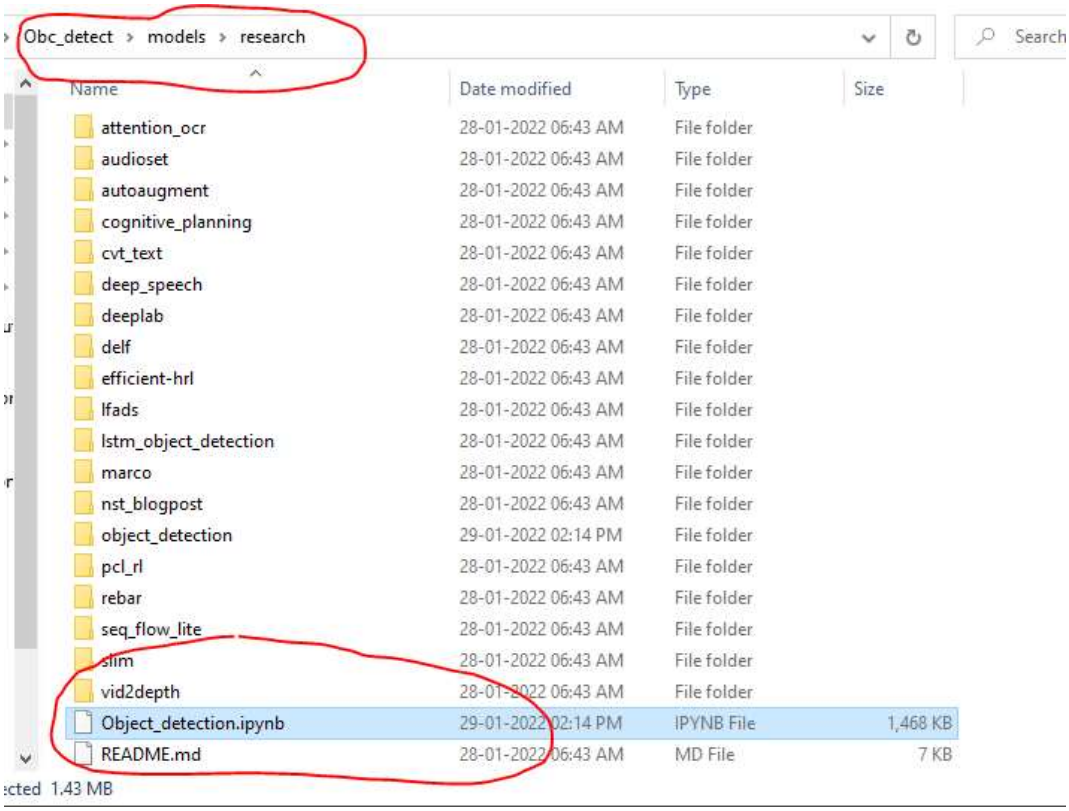
Go to models/research folder and type protoc object_detection/protos/*.proto —python_out=.



To check whether this worked or not, you can go to the protos folder inside models>object_detection>protos and there you can see that for every proto file there's one python file created.

Step 5 (All set Do Object Detection)

All are set Now we are going to doing Object detection . For Object detection you need to create python file inside research folder only otherwise it not working. This keep in mind. Like this



Method 2 (Short Method Using API)

Here we just run the following api on cmd

- pip install tensorflow-object-detection-api And all are set to detect object
- In this you can anywhere to create ipynb file and detect object.

if you getting any error in this you go to your C:\Python37\Lib\site-packages and go to object_detection folder and replace with my object_detection folder that i provide in below link :

https://github.com/pratyusa98/object_detection (https://github.com/pratyusa98/object_detection)

Code Implementaion

In [1]:

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
import pathlib
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
from IPython.display import display
from object_detection.utils import ops as utils_ops
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util
```

In [2]:

```
# while "models" in pathlib.Path.cwd().parts:
#     os.chdir('..')
```

In [4]:

```
def load_model(model_name):
    base_url = 'http://download.tensorflow.org/models/object_detection/tf2/20200711'
    model_file = model_name + '.tar.gz'
    model_dir = tf.keras.utils.get_file(
        fname=model_name,
        origin=base_url + model_file,
        untar=True)
    model_dir = pathlib.Path(model_dir)/"saved_model"
    model = tf.saved_model.load(str(model_dir))

    return model

model_name = 'ssd_mobilenet_v2_320x320_coco17_tpu-8' ## Change different model from zoo
detection_model = load_model(model_name)
```

All pretrained Models Are selected from

TensorFlow 2 Detection Model Zoo

Go and select name of model and paste in above model_name and run

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md (https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md)

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes

In [5]:

```
PATH_TO_LABELS = 'models/research/object_detection/data/mscoco_label_map.pbtxt'  # when use
# PATH_TO_LABELS = 'models/research/object_detection/data/mscoco_label_map.pbtxt' # when us


category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS, use_dis
```

In [6]:

```
def run_inference_for_single_image(model, image):
    image = np.asarray(image)
    # The input needs to be a tensor, convert it using `tf.convert_to_tensor`.
    input_tensor = tf.convert_to_tensor(image)
    # The model expects a batch of images, so add an axis with `tf.newaxis`.
    input_tensor = input_tensor[tf.newaxis,...]

    # Run inference
    model_fn = model.signatures['serving_default']
    output_dict = model_fn(input_tensor)


    # All outputs are batches tensors.
    # Convert to numpy arrays, and take index [0] to remove the batch dimension.
    # We're only interested in the first num_detections.
    num_detections = int(output_dict.pop('num_detections'))
    output_dict = {key:value[0, :num_detections].numpy()
                            for key,value in output_dict.items()}
    output_dict['num_detections'] = num_detections


    # detection_classes should be ints.
    output_dict['detection_classes'] = output_dict['detection_classes'].astype(np.int64)


    # Handle models with masks:
    if 'detection_masks' in output_dict:
        # Reframe the the bbox mask to the image size.
        detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(
            output_dict['detection_masks'], output_dict['detection_boxes'],
                            image.shape[0], image.shape[1])

        detection_masks_reframed = tf.cast(detection_masks_reframed > 0.5, tf.uint8)
        output_dict['detection_masks_reframed'] = detection_masks_reframed.numpy()

    return output_dict
```

In [7]:

```
def show_inference(model, image_path):
    # the array based representation of the image will be used later in order to prepare th
    # result image with boxes and labels on it.
    image_np = np.array(Image.open(image_path))


    # Actual detection.

    output_dict = run_inference_for_single_image(model, image_np)
    # Visualization of the results of a detection.
    vis_util.visualize_boxes_and_labels_on_image_array(
      image_np,
      output_dict['detection_boxes'],
      output_dict['detection_classes'],
      output_dict['detection_scores'],
      category_index,
      instance_masks=output_dict.get('detection_masks_reframed', None),
      use_normalized_coordinates=True,
      line_thickness=6)

    display(Image.fromarray(image_np))
```

In [11]:

```
PATH_TO_TEST_IMAGES_DIR = pathlib.Path('models/research/object_detection/test_images') # us
TEST_IMAGE_PATHS = sorted(list(PATH_TO_TEST_IMAGES_DIR.glob("*.jpg")))
```

In [12]:

```
for image_path in TEST_IMAGE_PATHS:
    print(image_path)
    show_inference(detection_model, image_path)
```

```
models\research\object_detection\test_images\image1.jpg
```