Implementation

start with one new colab notebook and follow the steps one by one.

step 1

Install tensorflow version 2 or higher

```
!pip install -U --pre tensorflow=="2.*"
```

step 2

make sure to install *pycocotools* for coco detection API.

```
!pip install pycocotools
```

step 3

get *tensorflow/models* by cloning the repository.

```
import os
import pathlib

if "models" in pathlib.Path.cwd().parts:
  while "models" in pathlib.Path.cwd().parts:
    os.chdir('..')
elif not pathlib.Path('models').exists():
  !git clone --depth 1 https://github.com/tensorflow/models
```

move (cd) to *research* directory of the repo

```
cd models/research
```

step 4

compile *protobufs*

```
!protoc object_detection/protos/*.proto --python_out=.
```

install *object_detection* python package

```
!pip install object_detection
```

step 5

import required libraries

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
from IPython.display import display
```

install *tf_slim* python package:

```
!pip install tf_slim
```

import object detection modules:

```
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
```

step 6

function to load your model

```
def load_model(model_name):
  base_url = 'http://download.tensorflow.org/models/object_detection/'
  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))
  model = model.signatures['serving_default']

  return model
```

this is the code to load your label map. Label maps map indices to category names/Class names. For example when our neural network predicts 1, it will correspond to “person” class or if it will predict, suppose 18, it will correspond to “dog” category.

```
PATH_TO_LABELS = 'object_detection/data/mscoco_label_map.pbtxt'
category_index =
label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS,
use_display_name=True)
```

this is the path to your test images. This will help you to check your model detections over the given class. You can change your test images by going to models/research/object_detection/test_images to check the accuracy of SSD mobilenet over the given class.

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

step 7

load your object detection SSD mobilenet v1 model for object detection

```
model_name = 'ssd_mobilenet_v1_coco_2017_11_17'
detection_model = load_model(model_name)
```

Now, check the model's input signature.

```
print(detection_model.inputs)
detection_model.output_dtypes
```

Now add this wrapper function which is calling the model and returns the output.

```
def run_inference_for_single_image(model, image):
  image = np.asarray(image)
  input_tensor = tf.convert_to_tensor(image)
  input_tensor = input_tensor[tf.newaxis,...]
  output_dict = model(input_tensor)
  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_detectionsoutput_dict['detection_classes'] =
output_dict['detection_classes'].astype(np.int64)

  if 'detection_masks' in output_dict: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
```

step 8

Now, this is the main step where you can just pass on the class id corresponding to the category you want to detect in the class_id parameter of the function given below.(provided it should be present in the coco dataset). You can check the class id and their respective classes here.

```
def show_inference(model, image_path,class_id):
  image_np = np.array(Image.open(image_path))
  output_dict = run_inference_for_single_image(model, image_np)
  boxes = []
  classes = []
  scores = []
  for i,x in enumerate(output_dict['detection_classes']):
    if x==class_id and output_dict['detection_scores'][i] > 0.5:
      classes.append(x)
      boxes.append(output_dict['detection_boxes'][i])
      scores.append(output_dict['detection_scores'][i])
  boxes = np.array(boxes)
  classes = np.array(classes)
  scores = np.array(scores)
  vis_util.visualize_boxes_and_labels_on_image_array(
      image_np,
      boxes,
      classes,
      scores,
      category_index,
      instance_masks=output_dict.get('detection_masks_reframed', None),
      use_normalized_coordinates=True,
      line_thickness=2)

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

step 9

this is the final step to see your output on the test images.

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

Results

Let’s suppose we have two images in the test_images directory and I’ve passed class_id to be 1 which corresponds to “person” so in both the images only persons are detected.

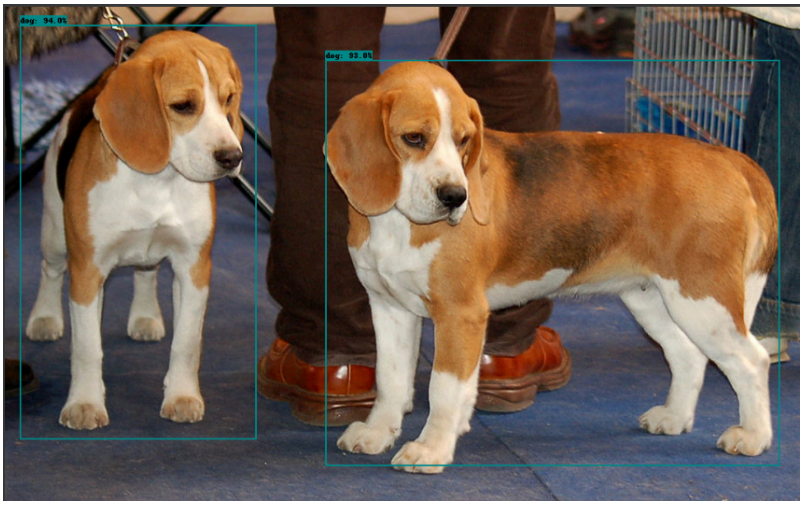
Classes of object -
https://drive.google.com/file/d/1ds-UMrPqAG3NQUXQ9itDyb1N06t4SYWb/view?usp=sharing

```
for image_path in TEST_IMAGE_PATHS:
  show_inference(detection_model, image_path, 1)
```

In this case only persons are detected in the second image and no other object is detected.

Now, let’s change the class_id to be 18 which corresponds to “dog” category.

```
for image_path in TEST_IMAGE_PATHS:
  show_inference(detection_model, image_path, 18)
```



Here, only dogs are detected and no other object.