Picamera Object Detection Using Tensorflow Classifier

Import packages

Import os

Import cv2

Import numpy as np

#from picamera.array import PiRGBArray

#from picamera import PiCamera

#import tensorflow as tf

Import RPi.GPIO as GPIO

Import time

Import argparse

Import sys

Import tensorflow.compat.v1 as tf

Import pygame

From gtts import gTTS

From pydub import AudioSegment

From pydub.playback import play

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_UP)

Import RPi.GPIO as GPIO

Import time

Import io

Tf.disable_v2_behavior()

Set up camera constants

```
IM_WIDTH = 1280

IM_HEIGHT = 720

Language = “en”

Slow_speed = False

S=0

Z=1

#IM_WIDTH = 640    Use smaller resolution for

#IM_HEIGHT = 480   slightly faster framerate


# Select camera type (if user enters –usbcam when calling this script,

# a USB webcam will be used)

#camera_type = ‘picamera’

# parser = argparse.ArgumentParser()

# parser.add_argument(‘—usbcam’, help=’Use a USB webcam instead of picamera’,

#                 action=’store_true’)

# args = parser.parse_args()

# if args.usbcam:

#     camera_type = ‘usb’

#

# # This is needed since the working directory is the object_detection folder.

# sys.path.append(‘..’)


# Import utilites

From utils import label_map_util

From utils import visualization_utils as vis_util

Def reading():
```

```
    # Convert text to speech
    Tts = gTTS(text=text, lang=language, slow=slow_speed)

    # Save the audio to a bytes buffer
    Audio_fp = io.BytesIO()
    Tts.write_to_fp(audio_fp)
    Audio_fp.seek(0)

    # Load audio with pydub and play
    Audio = AudioSegment.from_file(audio_fp, format="mp3")
    Play(audio)

# Name of the directory containing the object detection module we're using
MODEL_NAME = 'ssdlite_mobilenet_v2_coco_2018_05_09'

# Grab path to current working directory
CWD_PATH = os.getcwd()

# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,'data','mscoco_label_map.pbtxt')

# Number of classes the object detector can identify
```

```
NUM_CLASSES = 90

## Load the label map.

# Label maps map indices to category names, so that when the convolution

# network predicts `5`, we know that this corresponds to `airplane`.

# Here we use internal utility functions, but anything that returns a

# dictionary mapping integers to appropriate string labels would be fine

Label_map = label_map_util.load_labelmap(PATH_TO_LABELS)

Categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)

Category_index = label_map_util.create_category_index(categories)


# Load the Tensorflow model into memory.

Detection_graph = tf.Graph()

With detection_graph.as_default():

  Od_graph_def = tf.compat.v1.GraphDef()

  With tf.compat.v2.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:

    Serialized_graph = fid.read()

    Od_graph_def.ParseFromString(serialized_graph)

    Tf.import_graph_def(od_graph_def, name='')

  With tf.Session(graph=detection_graph) as sess:

  # Your code here


    Sess = tf.Session(graph=detection_graph)



# Define input and output tensors (i.e. data) for the object detection classifier
```

```
# Input tensor is the image
Image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

# Output tensors are the detection boxes, scores, and classes
# Each box represents a part of the image where a particular object was detected
Detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

# Each score represents level of confidence for each of the objects.
# The score is shown on the result image, together with the class label.
Detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
Detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

# Number of objects detected
Num_detections = detection_graph.get_tensor_by_name('num_detections:0')

# Initialize frame rate calculation
Frame_rate_calc = 1
Freq = cv2.getTickFrequency()
Font = cv2.FONT_HERSHEY_SIMPLEX

Camera = cv2.VideoCapture(0)
Ret = camera.set(3,IM_WIDTH)
Ret = camera.set(4,IM_HEIGHT)
Print("FRAME LOOP")
```

```
While z:
   Button_state = GPIO.input(18)
   If button_state==0:
      S=s+1
      Time.sleep(1)
      Print(s)
      If s==1:
         Y=1
         Print("FRAME OPEN")
         While y:
            Ret, frame = camera.read()
            Cv2.imshow('Video Feed', frame)



            Frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            Cv2.imshow('Camera', frame_rgb)
            Frame_expanded = np.expand_dims(frame, axis=0)
#
#       # Perform the actual detection by running the model with the image as input
            (boxes, scores, classes, num) = sess.run(
               [detection_boxes, detection_scores, detection_classes, num_detections],
            Feed_dict={image_tensor: frame_expanded})
#
    # Draw the results of the detection (aka 'visulaize the results')
            Vis_util.visualize_boxes_and_labels_on_image_array(
                  Frame,
```

```
            Np.squeeze(boxes),
            Np.squeeze(classes).astype(np.int32),
            Np.squeeze(scores),
            Category_index,
            Use_normalized_coordinates=True,
            Line_thickness=8,
            Min_score_thresh=0.85)
#
        Cv2.putText(frame,"FPS: {0:.2f}".format(frame_rate_calc),(30,50),font,1,(255,255,0),2,cv2.LINE_AA)
#
#       # All the results have been drawn on the frame, so it's time to display it.
        Cv2.imshow('Object detector', frame)
        Detected_objects = [category_index[i]['name'] for i in np.squeeze(classes).astype(np.int32) if np.squeeze(scores)[np.squeeze(classes).astype(np.int32).tolist().index(i)] > 0.85]
  # Convert detected objects to a single string
        If detected_objects:
          Text = "Detected objects are: " + ', '.join(detected_objects)
        Else:
          Text = "No objects detected with sufficient confidence."
  # Convert text to speech and play directly
        Tts = gTTS(text=text, lang=language, slow=slow_speed)
        Audio_fp = io.BytesIO()
        Tts.write_to_fp(audio_fp)
        Audio_fp.seek(0)
        Audio = AudioSegment.from_file(audio_fp, format="mp3")
```

```
        Play(audio)
    # Press ‘q’ to exit the loop
        If cv2.waitKey(30) & 0xff == ord(‘q’):
                Break
                Button_state = GPIO.input(18)
                If button_state==0:
                    Y=0
                Elif s==2:
                    S=0
                    Z=0
                    Print(“FRAME CLOSE”)

# Release the capture and writer objects
Camera.release()

Cv2.destroyAllWindows()
```