4.1.2 Object recognition+Voice broadcast

Combining the two functions of TensorFlow object recognition and voice broadcast. The process is as follows:

1) Import Tensorflow library and camera voice library
2) Camera initialization and display components
3) Initialize the tf model
4) Get camera data and transfer it to TensorFlow
5) Get the recognition and assign the result to speech synthesis
6) Broadcast the identified objects

Code path:
/home/pi/Yahboom_Project/4.AI Voice course/
03.Object_Recognition_Voice/Object_Recognition_Voice.ipynb

```
# Import the library for speech synthesis and broadcast
import time
import pygame
from aip import AipSpeech

# The following key should be replaced with your own key
""" Voice technology APPID AK SK """
SpeechAPP_ID = '17852430'
SpeechAPI_KEY ='eGeO4iQGAjHCrzBTYd1uvTtf'
SpeechSECRET_KEY = 'Cn1EVsUngZDbRLv4OxAFrDHSo8PsvFVP'

# Connect client
Speechclient = AipSpeech(SpeechAPP_ID, SpeechAPI_KEY,
SpeechSECRET_KEY)

# Voice broadcast initialization
pygame.mixer.init()

def AudioPlay(text):
    result = Speechclient.synthesis(text, 'zh', 1, {'spd': 2, 'vol': 2, 'per': 1})

    if not isinstance(result, dict):
        with open('./02.mp3', 'wb') as f:
            f.write(result)
        pygame.mixer.init()
        pygame.mixer.music.load('./02.mp3')
        pygame.mixer.music.play()
        time.sleep(2)
```

```
# Import TensorFlow library and camera library
import numpy as np
import cv2
import os,time
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_utils
import ipywidgets.widgets as widgets
from image_fun import bgr8_to_jpeg
```

```
# Camera initialization
# Init camera
cap = cv2.VideoCapture(0)
cap.set(3, 320) # set Width
cap.set(4, 240) # set Height
cap.set(5, 30)    # Set frame rate
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
cap.set(cv2.CAP_PROP_BRIGHTNESS, 40) #Set brightness -64 - 64    0.0
cap.set(cv2.CAP_PROP_CONTRAST, 50) #Set contrast -64 - 64    2.0
cap.set(cv2.CAP_PROP_EXPOSURE, 156)    #Set exposure 1.0 - 5000 156.0
```

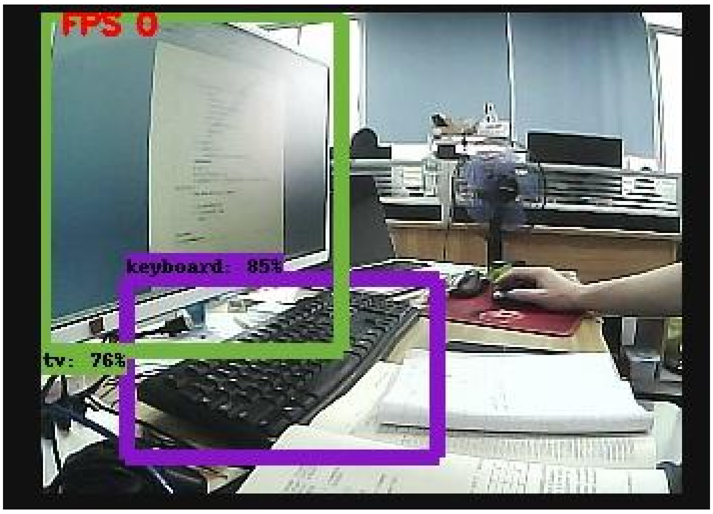
```
# Display camera assembly
image_widget = widgets.Image(format='jpg', width=320, height=240)
display(image_widget)
```

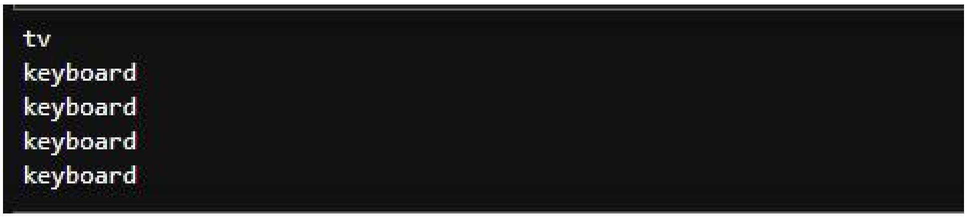
```
# Initialize the tf model
# Init tf model

MODEL_NAME = 'ssdlite_mobilenet_v2_coco_2018_05_09' #fast
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')
NUM_CLASSES = 90
IMAGE_SIZE = (12, 8)
fileAlreadyExists = os.path.isfile(PATH_TO_CKPT)

if not fileAlreadyExists:
	print('Model does not exsist !')
	exit
# LOAD GRAPH
print('Loading...')
detection_graph = tf.Graph()
with detection_graph.as_default():
	od_graph_def = tf.compat.v1.GraphDef()
	with tf.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='')
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)
print('Finish Load Graph..')
print(type(category_index))
print("dict['Name']: ", category_index[1]['name'])
```

```
# Main process
t_start = time.time()
fps = 0

with detection_graph.as_default():
    with tf.compat.v1.Session(graph=detection_graph) as sess:
        while True:
            ret, frame = cap.read()
            ###############
            image_np_expanded = np.expand_dims(frame, axis=0)
            image_tensor                                          =
detection_graph.get_tensor_by_name('image_tensor:0')
            detection_boxes                                       =
detection_graph.get_tensor_by_name('detection_boxes:0')
            detection_scores                                      =
detection_graph.get_tensor_by_name('detection_scores:0')
            detection_classes                                     =
detection_graph.get_tensor_by_name('detection_classes:0')
            num_detections                                        =
detection_graph.get_tensor_by_name('num_detections:0')

#             print('Running detection..')
            (boxes, scores, classes, num) = sess.run(
                [detection_boxes, detection_scores, detection_classes,
num_detections],
                feed_dict={image_tensor: image_np_expanded})

#             print('Done.   Visualizing..')
            vis_utils.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)

            for i in range(0, 10):
                if scores[0][i] >= 0.5:
                    print(category_index[int(classes[0][i])]['name'])
                    AudioPlay(category_index[int(classes[0][i])]['name'])
            ###############
            fps = fps + 1
            mfps = fps / (time.time() - t_start)
            cv2.putText(frame, "FPS " + str(int(mfps)), (10,10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 2)
            image_widget.value = bgr8_to_jpeg(frame)

            k = cv2.waitKey(3000) & 0xff
            if k == 27: # press 'ESC' to quit
                break

cap.release()
```

After we run above program, we can see following interface.
When some objects are recognized, the voice will be broadcast.





```
tv
keyboard
keyboard
keyboard
keyboard
```