4. Object Recognition

4.1. Introduction to TensorFlow

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while graph edges represent multidimensional data arrays (tensors) that flow between them. This flexible architecture allows you to deploy computations to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting your code. The TensorFlow User Guide provides a detailed overview and describes how to use and customize the TensorFlow deep learning framework. TensorFlow was originally developed by researchers and engineers in the Google Brain team in Google's Machine Intelligence Research organization for machine learning and deep neural network (DNN) research. The system is general enough to be applicable to a variety of other fields.

What is a Data Flow Graph?

A data flow graph describes mathematical computations as a directed graph of "nodes" and "edges". "Nodes" are generally used to represent mathematical operations applied, but can also represent the starting point of data input (feed in)/the end point of output (push out), or the end point of reading/writing persistent variables (persistent variables). "Wires" represent input/output relationships between "nodes". These data "wires" can transport multi-dimensional data arrays of "dynamically adjustable size", namely "tensors". The intuitive image of tensors flowing through the graph is the reason why this tool is named "Tensorflow". Once all tensors on the input end are ready, the nodes will be assigned to various computing devices to complete asynchronous and parallel execution of operations.

Key features of TensorFlow:

1. Data flow graph: The core concept of TensorFlow is the data flow graph, in which nodes represent mathematical operations and edges represent multi-dimensional data arrays (tensors) flowing between nodes. This graphical representation makes the calculation process clear and easy to optimize and parallelize.
2. Tensors and operations: In TensorFlow, data is represented as tensors, and operations (such as addition, multiplication, matrix operations, etc.) are defined as nodes in the graph. This structure enables the framework to handle a variety of tasks from simple mathematical operations to complex deep learning models.
3. Automatic differentiation: TensorFlow can automatically calculate the gradient of complex computational graphs, which is crucial for training neural networks. This means that users do not have to implement gradient calculations manually, saving a lot of time and effort.
4. Flexibility and modularity: TensorFlow provides a large number of modules and APIs that allow users to build complex models, while also enhancing their functionality through plug-ins and extensions. In addition, the TensorFlow 2.x version introduces the Eager Execution mode, which makes the code more intuitive and closer to immediate execution, making it easier to debug and prototype.

5. High-performance computing: TensorFlow is able to leverage hardware accelerators such as GPUs and TPUs (tensor processing units) to achieve efficient training and reasoning on large-scale data sets.
6. Distributed computing: TensorFlow supports distributed computing, allowing models to be trained in parallel on multiple devices or servers, which is especially important for processing large-scale data sets.
7. High-level API: In addition to the underlying API, TensorFlow also provides high-level APIs such as Keras, making model building and experimentation easier while maintaining access to the underlying details.
8. Model service and deployment: TensorFlow provides tools for model deployment and management, such as TensorFlow Serving, so that models can be easily used in production environments.
9. Community and Ecosystem: TensorFlow has a large developer community and a rich ecosystem, including pre-trained models, data sets, and various application examples, which greatly promotes learning and innovation.
10. Education and Documentation: TensorFlow provides detailed documentation and tutorials, as well as related educational materials, to help novices and professionals learn and master machine learning techniques.
11. Multi-language support: In addition to the Python interface, TensorFlow also supports languages such as C++, Java, Go, and R, allowing it to be integrated into different development environments.

4.2. Object Recognition

Source code path:

/home/pi/project_demo/07.AI_Visual_Recognition/04.Tensorflow_object_recognition/04_object_recognition.ipynb

```
#导入oled屏幕库 Import oled screen library
import sys
sys.path.append('/home/pi/software/oled_yahboom/')
from yahboom_oled import *
# 创建oled对象 Create an oled object
oled = Yahboom_OLED(debug=False)
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
```

```
# Init camera
cap = cv2.VideoCapture(0)  # 定义摄像头对象，参数0表示第一个摄像头 Define the camera object, parameter 0 represents the first camera
cap.set(3, 320) # set Width
cap.set(4, 240) # set Height
cap.set(5, 30)  #设置帧率 Setting the frame rate
# cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
# cap.set(cv2.CAP_PROP_BRIGHTNESS, 60) #设置亮度 -64 - 64  0.0 Set Brightness -64 - 64 0.0
```

```
# cap.set(cv2.CAP_PROP_CONTRAST, 50) #设置对比度 -64 - 64  2.0 Set Contrast -64 -
64 2.0
# cap.set(cv2.CAP_PROP_EXPOSURE, 156)  #设置曝光值 1.0 - 5000  156.0 Set the
exposure value 1.0 - 5000 156.0

# from picamera2 import Picamera2, Preview
# import libcamera
# picam2 = Picamera2()
# camera_config = picam2.create_preview_configuration(main=
{"format":'RGB888',"size":(320,240)})
# camera_config["transform"] = libcamera.Transform(hflip=1, vflip=1)
# picam2.configure(camera_config)
# picam2.start()
```

```
image_widget = widgets.Image(format='jpg', width=640, height=480)
```

```
# 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))
```

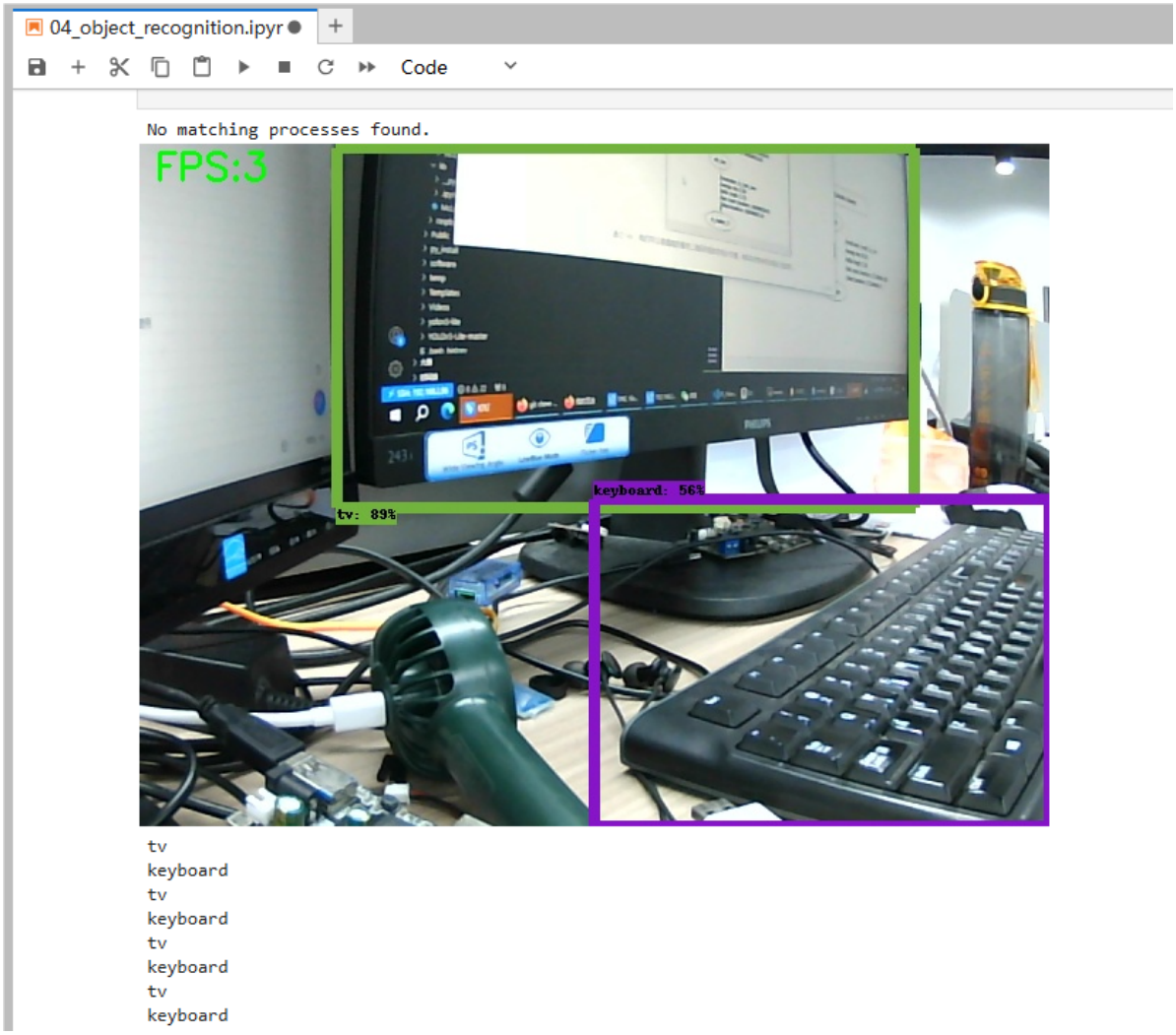
```
print("dict['Name']: ", category_index[1]['name'])
```

```
# Main
oled.init_oled_process() #初始化oled进程 Initialize oled process
```

```
oled.add_line("OBJTYPE:", 1)
oled.add_line("None", 3)
oled.refresh()
t_start = time.time()
fps = 0
display(image_widget)
with detection_graph.as_default():
    with tf.compat.v1.Session(graph=detection_graph) as sess:
        while True:
            #frame = picam2.capture_array()
            ret, frame = cap.read()
#             frame = cv2.flip(frame, -1) # Flip camera vertically
#              frame = cv2.resize(frame,(320,240))
            ###############
            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..')
            # 应用非极大值抑制
            selected_indices = tf.image.non_max_suppression(
                np.squeeze(boxes),
                np.squeeze(scores),
                max_output_size=100,  # 根据需要设置最大输出数量
                iou_threshold=0.7)  # 调整重叠阈值

            # 使用 tf.gather 来高效地应用索引
            filtered_boxes = tf.gather(tf.squeeze(boxes), selected_indices)
            filtered_scores = tf.gather(tf.squeeze(scores), selected_indices)
            filtered_classes = tf.gather(tf.cast(tf.squeeze(classes), tf.int32),
selected_indices)

            # 在会话中运行过滤操作
            (filtered_boxes, filtered_scores, filtered_classes) = sess.run([
                filtered_boxes,
                filtered_scores,
                filtered_classes
            ])

            # 可视化
            vis_utils.visualize_boxes_and_labels_on_image_array(
                frame,
```

```
                filtered_boxes,
                filtered_classes,
                filtered_scores,
                category_index,
                use_normalized_coordinates=True,
                line_thickness=8)

            for i in range(0, 10):
                if scores[0][i] >= 0.7:
                    print(category_index[int(classes[0][i])]['name'])
                    oled.clear()
                    objtype_str=category_index[int(classes[0][i])]['name']
                    oled.add_line("OBJTYPE:", 1)
                    oled.add_line(objtype_str, 3)
                    oled.refresh()
            ###############
            fps = fps + 1
            mfps = fps / (time.time() - t_start)
            cv2.putText(frame, "FPS:" + str(int(mfps)), (10,25),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,255,0), 2)
            image_widget.value = bgr8_to_jpeg(frame)

            k = cv2.waitKey(1) & 0xff
            if k == 27:# press 'ESC' to quit
                cap.release()
                # 恢复屏幕基础数据显示 Restore basic data display on screen
                os.system("python3
/home/pi/software/oled_yahboom/yahboom_oled.py &")
                break
```

```
cap.release()
# 恢复屏幕基础数据显示 Restore basic data display on screen
os.system("python3 /home/pi/software/oled_yahboom/yahboom_oled.py &")
# picam2.stop()
# picam2.close()
#最后需要释放掉摄像头的占用 Finally, you need to release the camera's occupancy
```



这个程序在JupyterLab运行帧率有点低的，有误识别的情况。运行时，我们可以看到oled会显示识别到的物体英文名字。在同个文件夹下的**04_object_recognition_nms.ipynb**文件，这个误识别率会相对低一点，帧率也会低一点。

在需要关闭的时候选中运行的这段代码点击停止按钮关闭程序，最后记得释放掉视频流才可以在其他地方使用摄像头。

04_object_recognition.ipyr ●

Code

```
            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'])
                    oled.clear()
                    objtype_str=category_index[int(classes[0][i])]['name']
                    oled.add_line("OBJTYPE:", 1)
                    oled.add_line(objtype_str, 3)
                    oled.refresh()
            ##############
            fps = fps + 1
            mfps = fps / (time.time() - t_start)
            cv2.putText(frame, "FPS:" + str(int(mfps)), (10,25), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,255,0), 2)
            image_widget.value = bgr8_to_jpeg(frame)

            k = cv2.waitKey(1) & 0xff
            if k == 27:# press 'ESC' to quit
                cap.release()
                # 恢复屏幕基础数据显示 Restore basic data display on screen
                os.system("python3 /home/pi/software/oled_yahboom/yahboom_oled.py &")
                break
```

```
No matching processes found.
```