

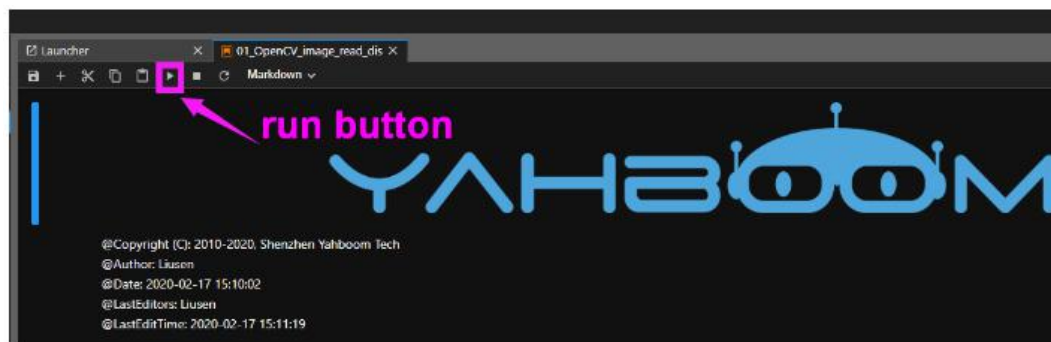


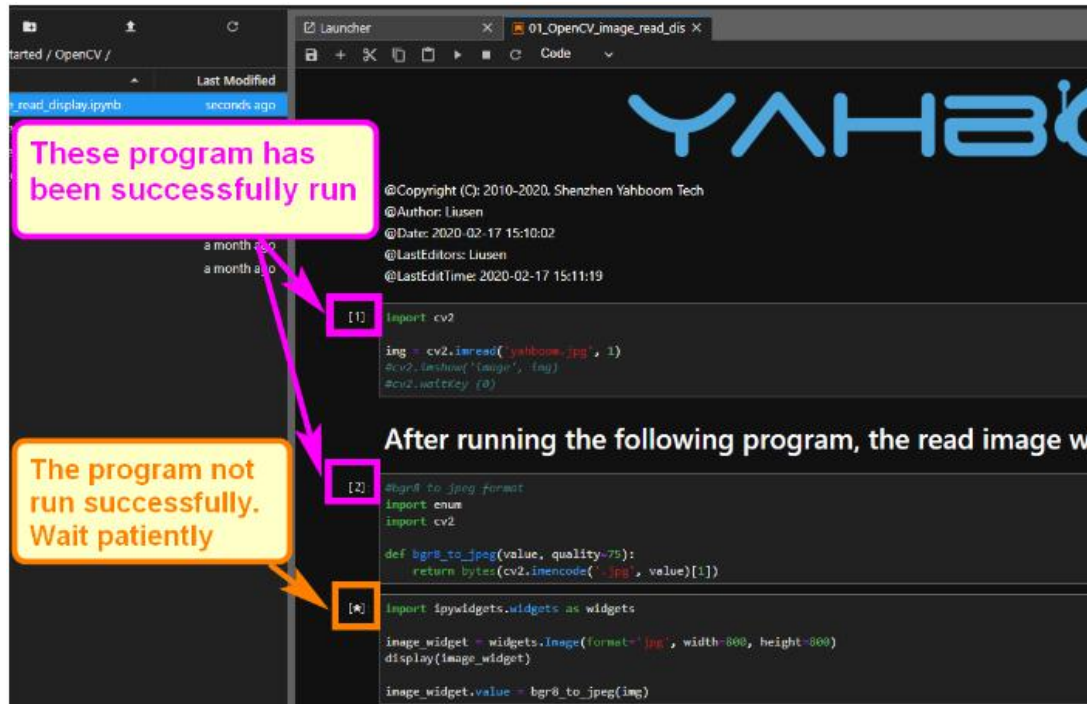
TensorFlow is an open source software library that uses data flow graphs for numerical calculations.

Features

1. High flexibility
2. True Portability
3. Connect scientific research and products
4. Differentiate automatically
5. Support multi-language
6. Performance optimization

1. Run program





2. Program analysis

2.1 Import opencv, tensorflow, control display related libraries.

```
[1]: 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
```

2.2 Import jetcham library for camera use.

Note:

The camera number used when calling the jetcham library needs to be video0.

For example, the code we are using now is a CSI camera, so the CSI camera number in the system also needs to be video0 to be able to call normally.

If you need to use a USB camera, you need to remove the CSI camera on the Jeston NANO.

If you connect a USB camera and a CSI camera at the same time, it is generally assigned to the CSI camera as video0 and the USB camera as video1, so that the USB camera cannot be used normally.

```
[2]: #from jetcam.usb_camera import USBCamera
      from jetcam.csi_camera import CSICamera
      from jetcam.utils import bgr8_to_jpeg

      #camera = USBCamera(width=320, height=240, capture_fps=30)
      camera = CSICamera(width=320, height=240, capture_fps=30)

      camera.running = True
```

2.3 Import libraries related to tensorflow object recognition and create camera display controls.

After running, a frame of the camera will be displayed, and the real-time image will only be displayed if the following continuous cycle update is required.

```
[3]: # 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 exist !')
    exit

[4]: # LOAD GRAPH
print('Loading...')
detection_graph = tf.Graph()
with detection_graph.as_default(): #语句下定义属于计算图detection_graph的张量和操作
    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..')

Loading...
Finish Load Graph..

[5]: print(type(category_index))

<class 'dict'>

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

[ ]: image_widget = widgets.Image(format='jpg', width=320, height=240)
      display(image_widget)
      image_widget.value = bgr8_to_jpeg(camera.value)
```

2.4 Method-1, call the camera to identify through the while loop. We can stop modifying the program through the stop button above.

```
[*]:
# Main
t_start = time.time()
fps = 0

with detection_graph.as_default():
    with tf.compat.v1.Session(graph=detection_graph) as sess:
        while True:
            frame = camera.value
            # 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..')
            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'])
            #####
            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)
```

2.5 Method-2:

```
[ ]: detection_graph.as_default()
sess = tf.compat.v1.Session(graph=detection_graph)

t_start = time.time()
fps = 0

def update_image(change):
    global fps
    global sess
    frame = change['new']
    image_np_expanded = np.expand_dims(frame, axis=0)
    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'])
    #####
    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)
```


2.6 Only one of the above two methods can be used at the same time.

```

#camera = USBCamera(width=320, height=240, capture_fps=30)
camera = CSICamera(width=320, height=240, capture_fps=30)

camera.running = True

[3]: image_widget = widgets.Image(format='jpg', width=320, height=240)
display(image_widget)
image_widget.value = bgr8_to_jpeg(camera.value)



[4]: # 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)

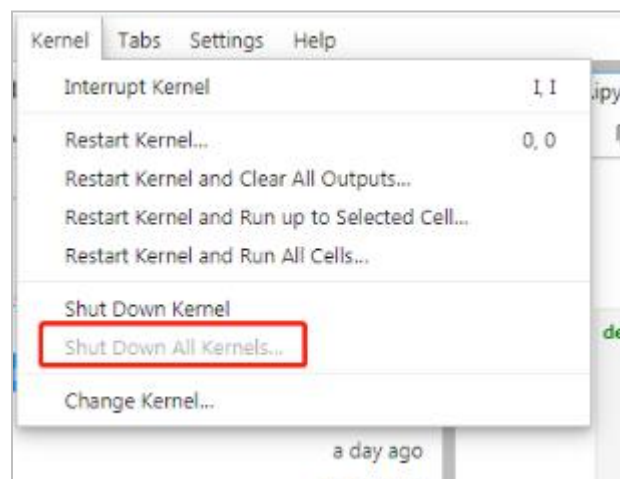
if not fileAlreadyExists:
    print('Model does not exist !')
    exit

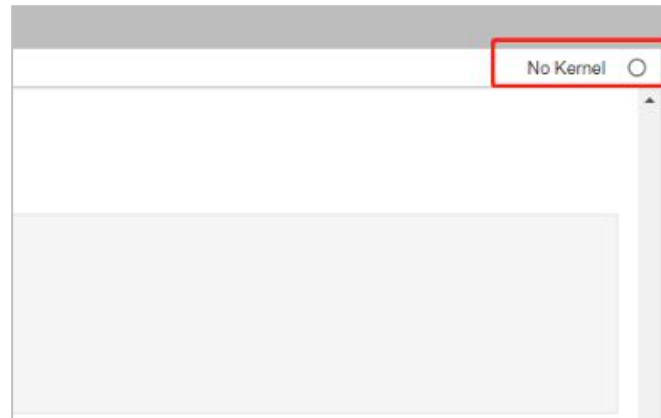
[5]: # LOAD GRAPH
print('Loading...')
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

```

2.7 If you need to shut down this process completely, please do the following operation .

1) Click **[shut down all kernels]** and wait for **[no kernels]** on the upper right corner. After restarting the kernel and clear output, wait for the right side to become python3. If the camera is still occupied, it is recommended to restart





2) Click [restart kernel and clear output], and wait for [Python3] on the upper right corner.

