Tips:
Because the Jetson NANO 2G version with smaller memory, if you have run other programs before executing this program, the video will freeze because the memory is not released.
If this happens, please restart Jetson NANO 2G to release the memory before running this program.
The camera imaging may be upside down during use this function, you can solve this problem by re-installing the camera.

1. Install dependencies

1.1 Install PyTorch and Torchvision
Please check **[3.AI Getting started tutorial]**--**[11.Install Pytorch]**

1.2 Install torch2trt

```
git clone https://github.com/NVIDIA-AI-IOT/torch2trt
cd torch2trt
sudo python3 setup.py install --plugins
```

1.3 Install others package

```
sudo pip3 install tqdm cython pycocotools
sudo apt-get install python3-matplotlib
```

1.4 Install trt_pose

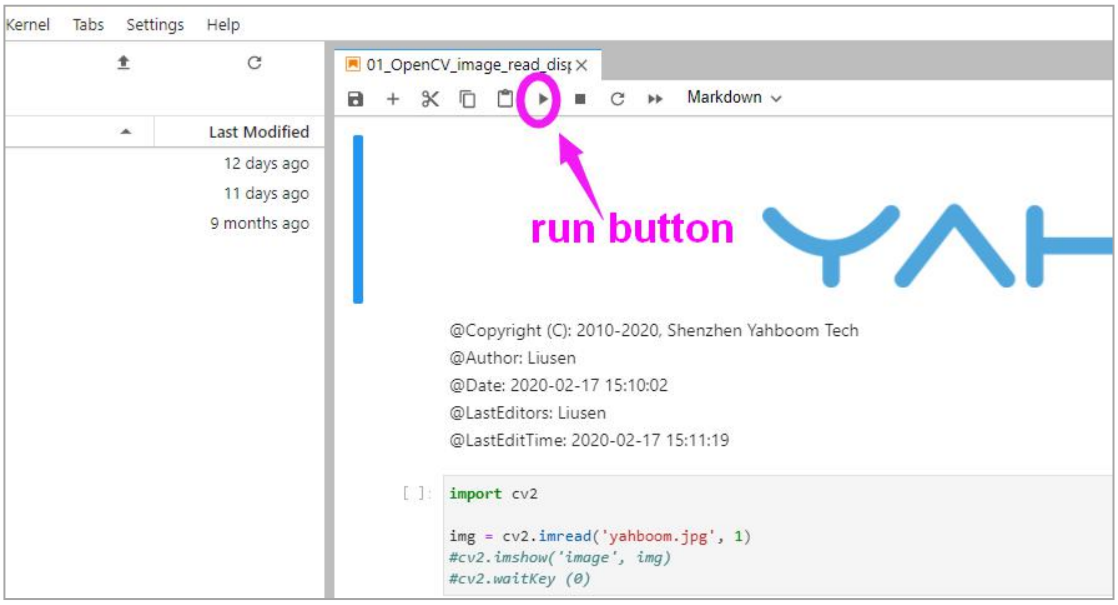
```
git clone https://github.com/NVIDIA-AI-IOT/trt_pose
cd trt_pose
sudo python3 setup.py install
```

2. About code

Please check **Human_posture_recognition** file.

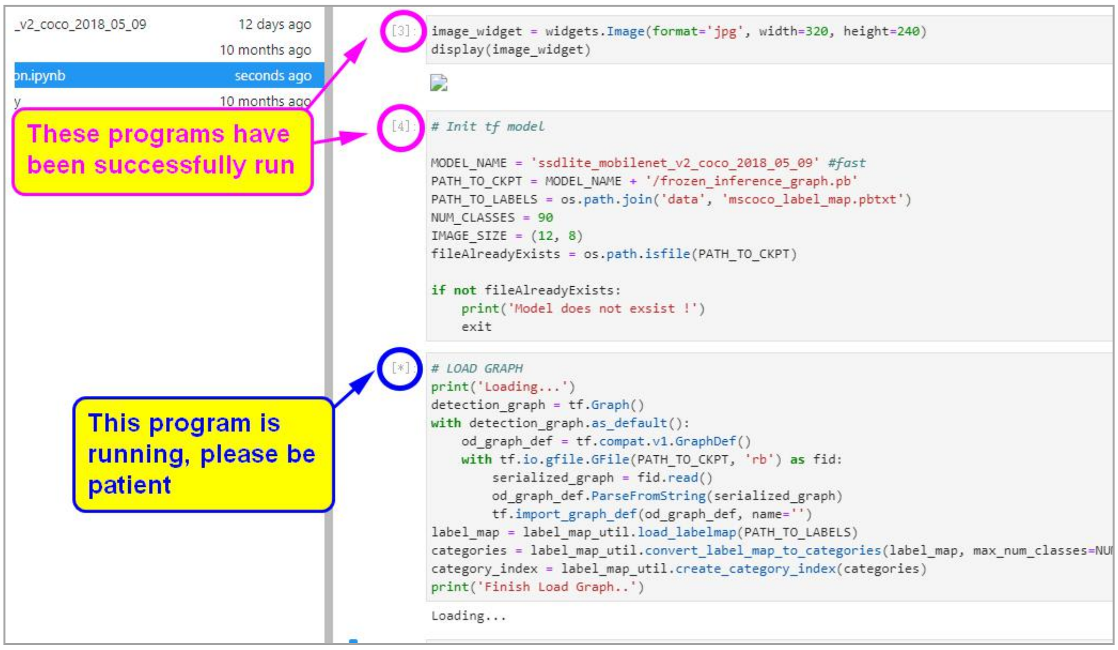
3. Run program on JupyterLab

Open the **live_demo.ipynb** on JupyterLab.



@Copyright (C): 2010-2020, Shenzhen Yahboom Tech
@Author: Liusen
@Date: 2020-02-17 15:10:02
@LastEditors: Liusen
@LastEditTime: 2020-02-17 15:11:19

```
[ ]: import cv2

     img = cv2.imread('yahboom.jpg', 1)
     #cv2.imshow('image', img)
     #cv2.waitKey (0)
```


_v2_coco_2018_05_09
12 days ago
10 months ago
on.ipynb
seconds ago
10 months ago
These programs have been successfully run
This program is running, please be patient

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

[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 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=NU
     category_index = label_map_util.create_category_index(categories)
     print('Finish Load Graph..')

     Loading...
```

3.2 Import jetcham library for camera.

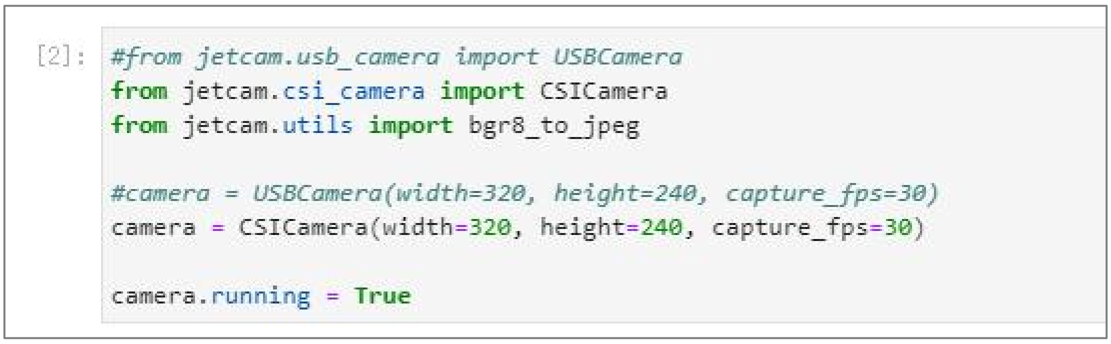
Note:

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.

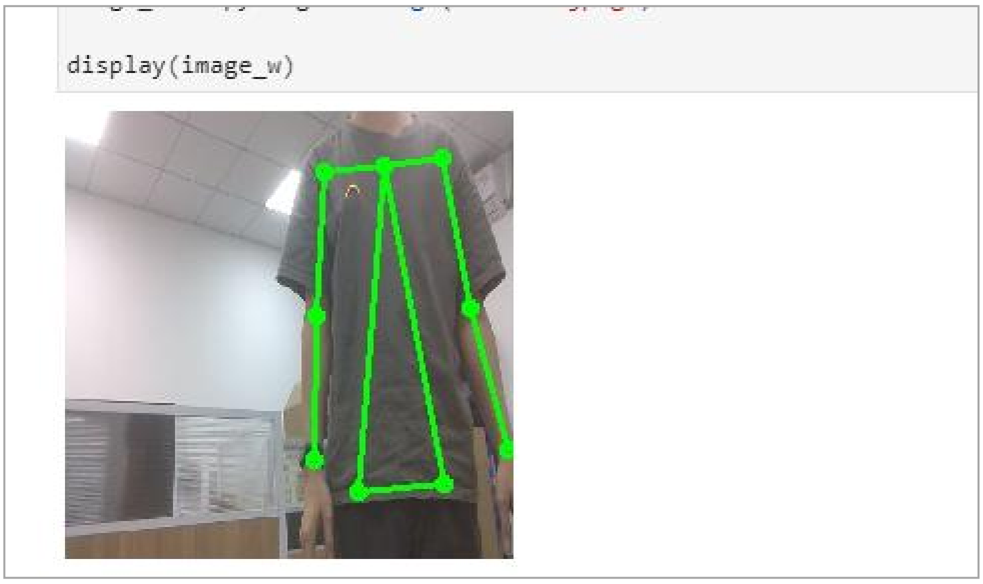
If you need to use a USB camera, you need to remove the CSI camera on the Jeston NANO.

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.

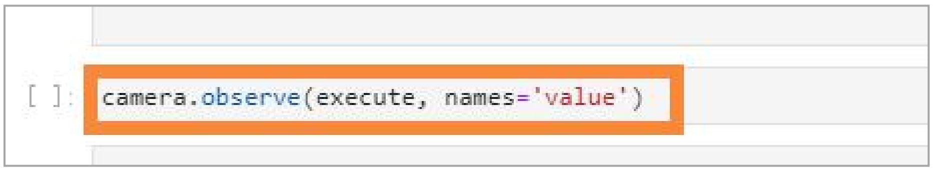


```
[2]: #from jetcam.usb_camera import USBCamera
     from jetcam.csi_camera import CSICamera
     from jetcam.utils import bgr8_to_jpeg

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

     camera.running = True
```

3.3 We need to run the program to **camera.observe(execute, names='value')**. Whenever a new camera frame is received, the execute function is called.

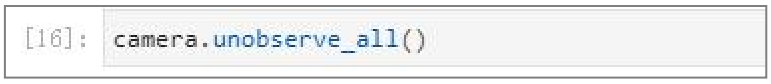


```
display(image_w)
```



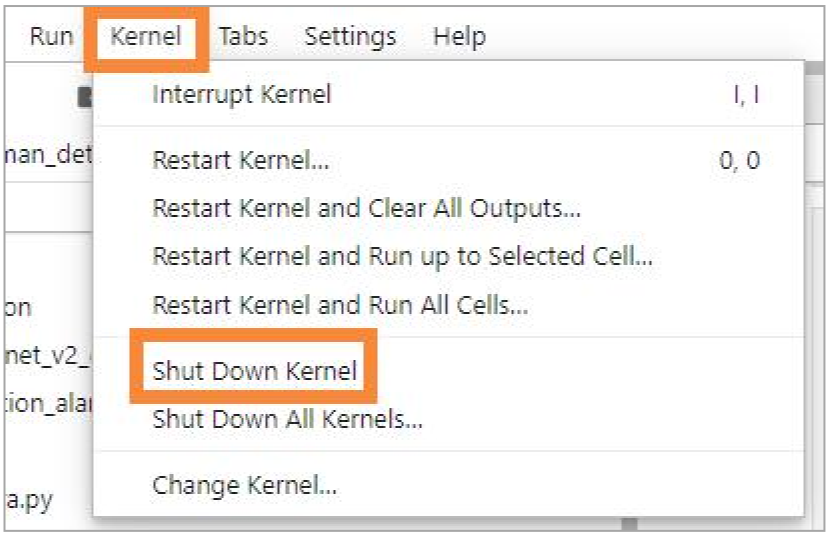
```
[ ]: camera.observe(execute, names='value')
```

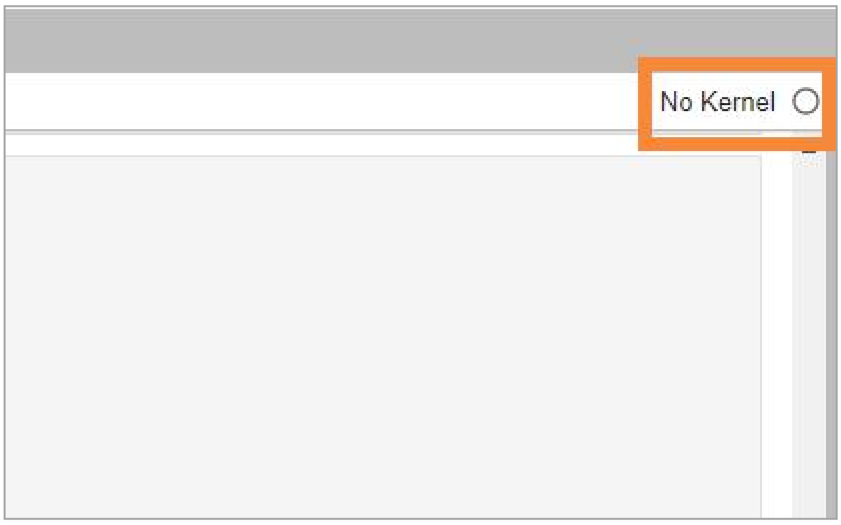
This function is used to end the camera frame callback, but the camera cannot be released.



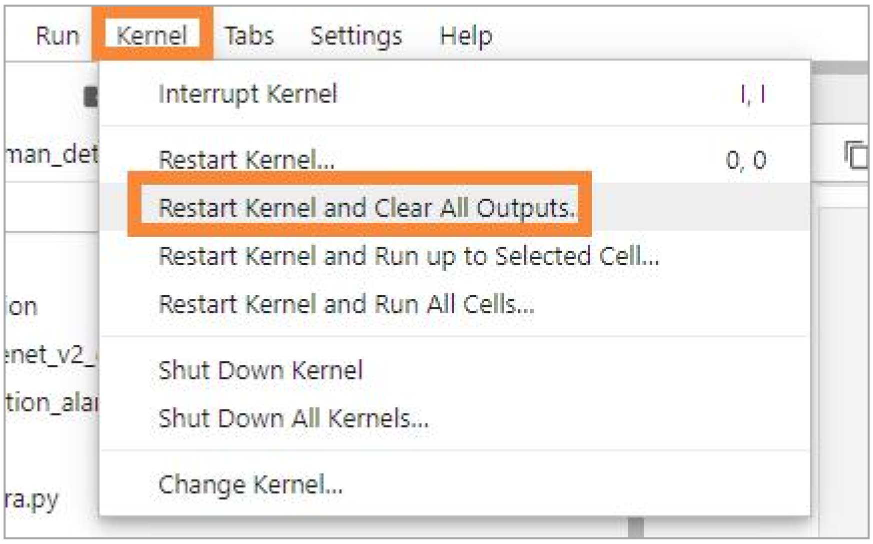
```
[16]: camera.unobserve_all()
```

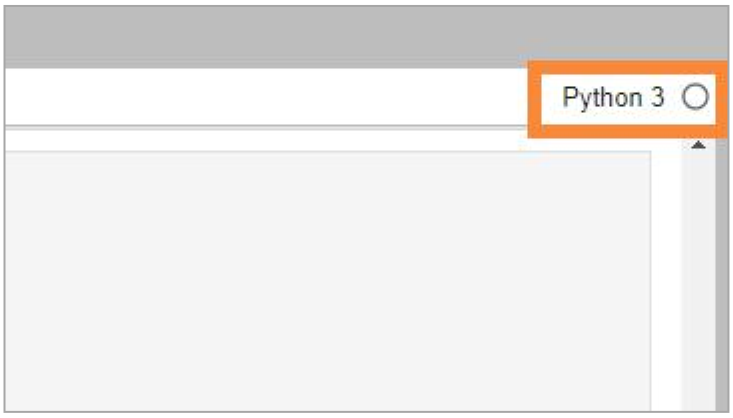
If you need to shut down this process completely, please do the following operation .
1) Click **[shut down all kernels]** and wait for **[no kernels]** on the upper right corner. After restarting the kernel and clear output, wait for the right side to become python3. If the camera is still occupied,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





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




Python 3