

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.

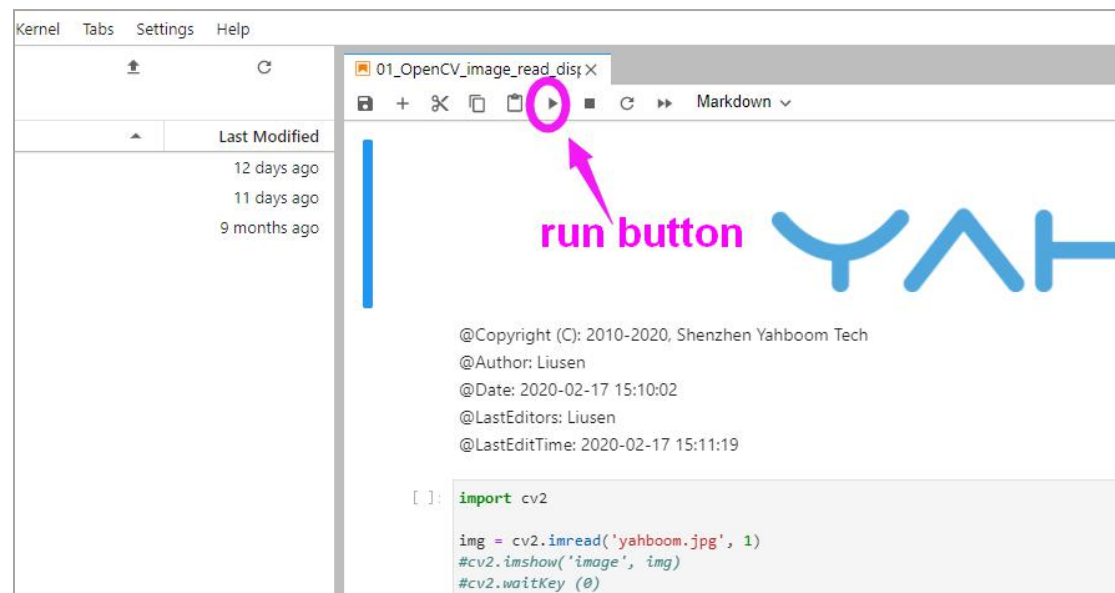
The camera imaging may be upside down during use this function, you can solve this problem by re-installing the camera.

1. About code

Please check [Gesture Recognition data collection](#) file.

2. Run program on JupyterLab

Open the [Data_collection.ipynb](#) on JupyterLab.





```

_v2_coco_2018_05_09      12 days ago
on.ipynb                 seconds ago
y                         10 months ago

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

    Loading...

```

3. Program analysis

3.1 Initialize the camera in the notebook and display.

The neural network uses an image of 224×224 pixels as input. So we set the camera to this size and minimize the data set by minimizing the file size. In some cases, we need to use a larger image size when collecting data, and then reduce the size when processing.

Real-time display camera

Initialize the camera in the notebook and display.

The neural network uses an image of 224×224 pixels as input. So we set the camera to this size and minimize the data set by minimizing the file size. In some cases, we need to use a larger image size when collecting data, and then reduce the size when processing.

```

import traitlets
import ipywidgets
import ipywidgets.widgets as widgets
from IPython.display import display
from camera import Camera
from image import bgr8_to_jpeg

camera = Camera.instance(width=224, height=224)

image = widgets.Image(format='jpeg', width=224, height=224) # this width and height doesn't need

camera_link = traitlets.dlink((camera, 'value'), (image, 'value'), transform=bgr8_to_jpeg)

display(image)

```



3.2 Create some directories to store data. We will create a dataset folder with 3 folders(one, two, three) inside.

```
import os
CLASS = ['one', 'two', 'three']
one = 'dataset/one'
two = 'dataset/two'
three = 'dataset/three'

# we have this "try/except" statement because these next functions can throw an error if the d
try:
    os.makedirs(one)
    os.makedirs(two)
    os.makedirs(three)

except FileExistsError:
    print('Directories not created because they already exist')
```

Tips:

If you run the cell code for the second time, the system will prompt'Directories not created because they already exist'.

As shown below.

```
[2]: import os
CLASS = ['one', 'two', 'three']
one = 'dataset/one'
two = 'dataset/two'
three = 'dataset/three'

# we have this "try/except" statement because these next functions can throw an error if the directories exist alr
try:
    os.makedirs(one)
    os.makedirs(two)
    os.makedirs(three)

except FileExistsError:
    print('Directories not created because they already exist')
```

Directories not created because they already exist

3.3 After running the code, the JupyterLab operation interface will display the “Save button” and the option bar to choose to save the picture to the

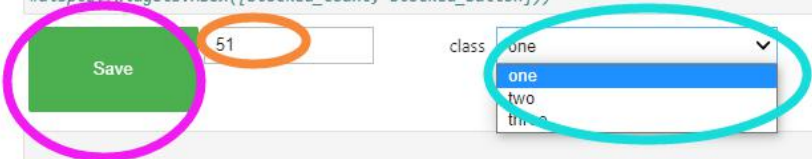
corresponding folder, as well as a text box showing the number of pictures in the folder.

As shown below.

```
[3]: dataset_widget = ipywidgets.Dropdown(options=CLASS, description='class')
button_layout = widgets.Layout(width='128px', height='64px')
save_button = widgets.Button(description='Save', button_style='success', layout=button_layout)
#blocked_button = widgets.Button(description='add blocked', button_style='danger', layout=button_layout)
save_count = widgets.IntText(layout=button_layout, value=len(os.listdir('dataset/'+str(dataset_widget.value))))
#blocked_count = widgets.IntText(layout=button_layout, value=len(os.listdir(blocked_dir)))

display(widgets.HBox([save_button, save_count, dataset_widget]))

#display(widgets.HBox([blocked_count, blocked_button]))
```



3.4 Add a function to the “Save button”.

```
from uuid import uuid1

def save_snapshot(directory):
    image_path = os.path.join(directory, str(uuid1()) + '.jpg')
    with open(image_path, 'wb') as f:
        f.write(image.value)

def save_():
    global save_count
    save_snapshot('dataset/'+str(dataset_widget.value))
    save_count.value = len(os.listdir('dataset/'+str(dataset_widget.value)))

# attach the callbacks, we use a 'lambda' function to ignore the
# parameter that the on_click event would provide to our function
# because we don't need it.
save_button.on_click(lambda x: save_())
#blocked_button.on_click(lambda x: save_blocked())
```

3.5 We can save the image to the corresponding directory through the button and option bar above.

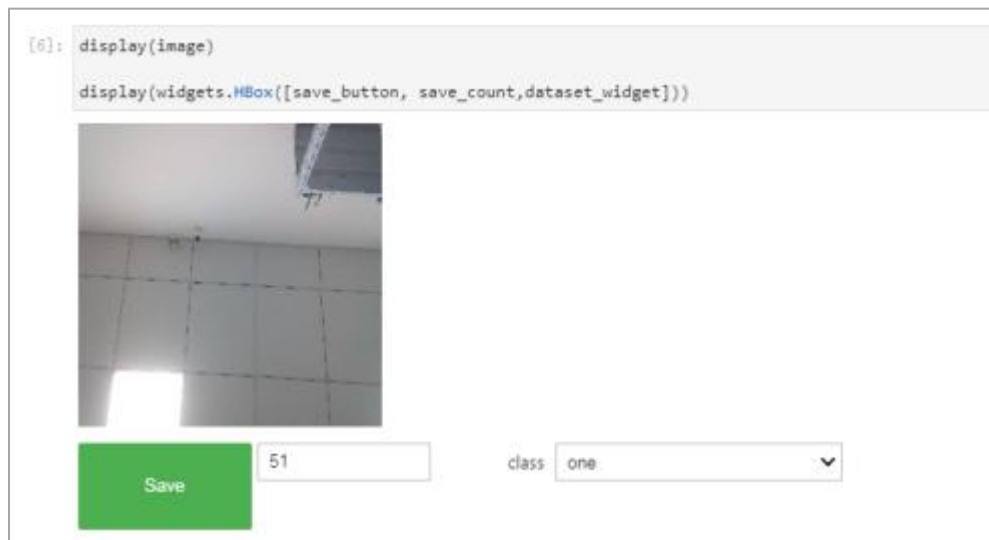
You can use the directory file browser on the left side of Jupyter to view these files.

We need to continue to collect some data. After selecting a different class, let the camera face the corresponding scene and save the picture with save.

1) Different directions

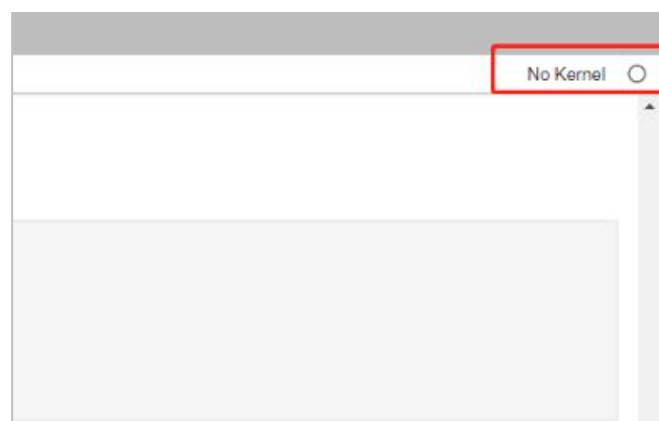
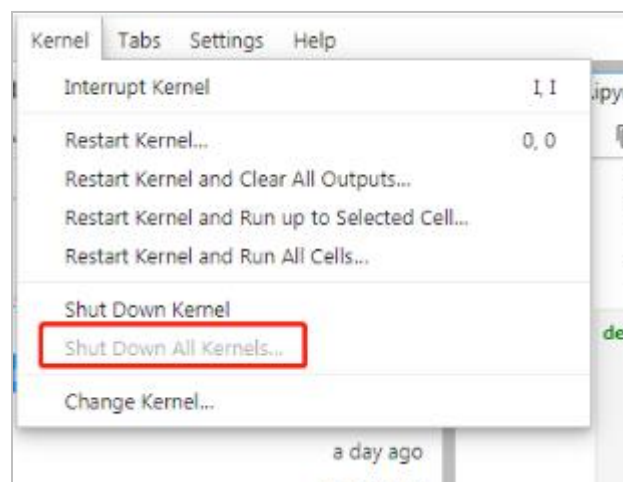
2) Different brightness

After running the cell code below, images and buttons will be displayed, and you can start collecting data.



3.6 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.

