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 colorama
pip3 install colorama

Or Input following command
git clone https://github.com/tartley/colorama.git
cd colorama
sudo python3 setup.py install

2. About code
Please check **Guessing_game** file.

3. Program analysis
3.1 Load model

```
import torch
import torchvision

model = torchvision.models.alexnet(pretrained=False)
model.classifier[6] = torch.nn.Linear(model.classifier[6].in_features, 3)

# Load the ``gesture_model.pth'' model that has been trained

model.load_state_dict(torch.load('gesture_model.pth'))
```

3.2 Preprocessing function

```
import cv2
import numpy as np

mean = 255.0 * np.array([0.485, 0.456, 0.406])
stdev = 255.0 * np.array([0.229, 0.224, 0.225])

normalize = torchvision.transforms.Normalize(mean, stdev)

def preprocess(camera_value):
    global device, normalize
    x = camera_value
    x = cv2.cvtColor(x, cv2.COLOR_BGR2RGB)
    x = x.transpose((2, 0, 1))
    x = torch.from_numpy(x).float()
    x = normalize(x)
    x = x.to(device)
    x = x[None, ...]
    return x
```

Due to the low 2G memory, it is prone to screen freezes, so we use the jetcham library to call the camera to achieve better experimental results.

```
#from jetcam.usb_camera import USBCamera
from jetcam.csi_camera import CSICamera
from jetcam.utils import bgr8_to_jpeg
import traitlets
from IPython.display import display
import ipywidgets
import ipywidgets.widgets as widgets


#camera = USBCamera(width=WIDTH, height=HEIGHT, capture_fps=30)
camera = CSICamera(width=224, height=224, capture_fps=30)

camera.running = True

image = widgets.Image(format='jpeg', width=224, height=224)
display(widgets.HBox([image]))
```

3.3 Create a function that will call this function whenever the value of the camera changes. This function will perform the following steps

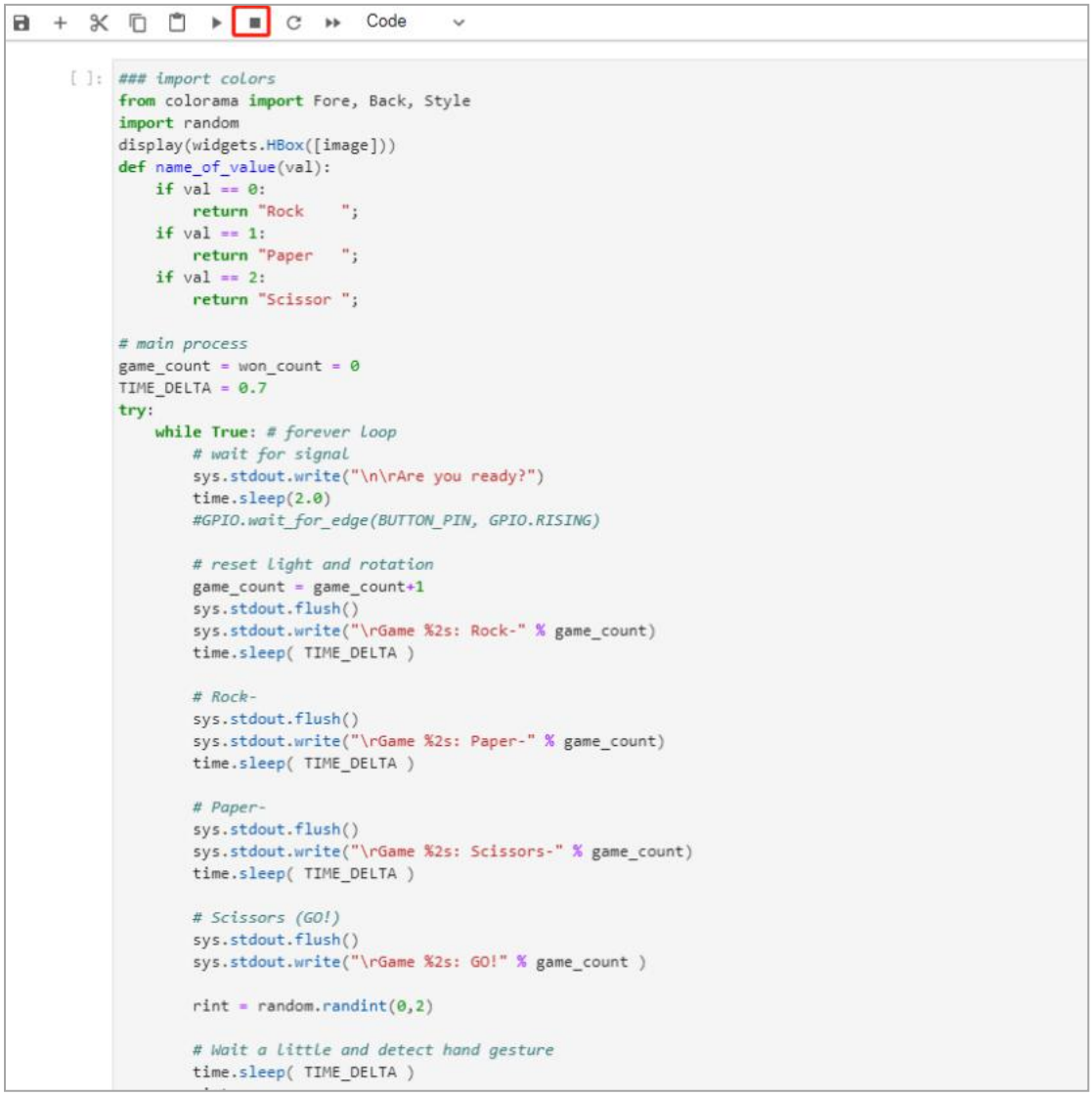
1) Preprocess the camera image
2) Execute neural network
3) Compare the values of the 3 categories and assign the number to a

```
[2]: import torch.nn.functional as F
import time
import sys

a=0
one_blocked=0.0
two_blocked=0.0
three_blocked=0.0

def update(change):
    global one_blocked, two_blocked, three_blocked,a
    x = change['new']
    x = preprocess(x)
    y = model(x)

    # we apply the `softmax` function to normalize the output vector so it sums to 1 (which
#    y = F.softmax(y, dim=1).detach().cpu().numpy().flatten()
    y = F.softmax(y, dim=1)
    one_blocked = float(y.flatten()[0])
    two_blocked = float(y.flatten()[1])
    three_blocked = float(y.flatten()[2])
    if(one_blocked > two_blocked and one_blocked > three_blocked):
        a = 0
    elif(two_blocked > one_blocked and two_blocked > three_blocked):
        a = 1
    elif(three_blocked > one_blocked and three_blocked > one_blocked):
        a = 2



    '''index = y.argmax()
    if y[index]>0.70:
        prediction_widget.value = hand[index]
    else:
        prediction_widget.value = hand[0]'''


    time.sleep(0.001)

update({'new': camera.value})  # we call the function once to intialize
```

3.4 We have created the neural network to perform the function, but now we need to attach it to the camera for processing. We use the “observe” function to complete this process.

```
[ ]: camera.observe(update, names='value')  # this attaches the 'update' function to the 'value' traitlet of our camera
```

If you need to stop this program, please press stop button on JuyterLab.



```
[ ]: ### import colors
from colorama import Fore, Back, Style
import random
display(widgets.HBox([image]))
def name_of_value(val):
    if val == 0:
        return "Rock    ";
    if val == 1:
        return "Paper   ";
    if val == 2:
        return "Scissor ";

# main process
game_count = won_count = 0
TIME_DELTA = 0.7
try:
    while True: # forever loop
        # wait for signal
        sys.stdout.write("\n\rAre you ready?")
        time.sleep(2.0)
        #GPIO.wait_for_edge(BUTTON_PIN, GPIO.RISING)

        # reset light and rotation
        game_count = game_count+1
        sys.stdout.flush()
        sys.stdout.write("\rGame %2s: Rock-" % game_count)
        time.sleep( TIME_DELTA )

        # Rock-
        sys.stdout.flush()
        sys.stdout.write("\rGame %2s: Paper-" % game_count)
        time.sleep( TIME_DELTA )

        # Paper-
        sys.stdout.flush()
        sys.stdout.write("\rGame %2s: Scissors-" % game_count)
        time.sleep( TIME_DELTA )

        # Scissors (GO!)
        sys.stdout.flush()
        sys.stdout.write("\rGame %2s: GO!" % game_count )

        rint = random.randint(0,2)

        # Wait a little and detect hand gesture
        time.sleep( TIME_DELTA )
```

3.5 After successfully running the program, the code will randomly display a simple stone cloth.

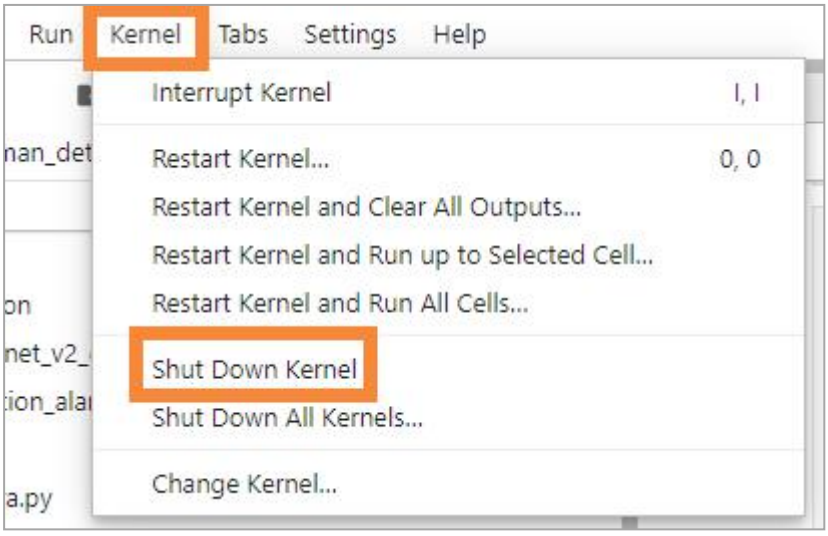
Then compare according to the currently recognized gestures, and print out the computer's gesture name and the gesture name that recognizes you to judge whether you win or lose.

At the same time, the proportion of the 3 gestures will be printed out, ranging from 1 to 10.

```
Game  1  Opponent: Scissor  You: Scissor       R:0.411234, P:0.155385, S:0.433380      Draw
Game  2  Opponent: Rock     You: Paper        R:0.070242, P:0.786884, S:0.142873      You win!!
Game  3  Opponent: Paper    You: Paper        R:0.036942, P:0.895408, S:0.067650      Draw
Game  4  Opponent: Rock     You: Paper        R:0.353006, P:0.499509, S:0.147485      You win!!
Game  5  Opponent: Rock     You: Paper        R:0.202601, P:0.492071, S:0.305328      You win!!
Game  6  Opponent: Paper    You: Paper        R:0.332642, P:0.409246, S:0.258112      Draw
Game  7  Opponent: Scissor  You: Rock         R:0.693621, P:0.091128, S:0.215251      You win!!
Game  8  Opponent: Rock     You: Rock         R:0.632806, P:0.213825, S:0.153368      Draw
Game  9  Opponent: Paper    You: Paper        R:0.190023, P:0.433364, S:0.376612      Draw
Game 10  Opponent: Paper    You: Rock         R:0.487510, P:0.206356, S:0.306134      You lose
Game 11  Opponent: Scissor  You: Paper        R:0.326193, P:0.373969, S:0.299837      You lose
Game 12  Opponent: Scissor  You: Rock         R:0.452247, P:0.230600, S:0.317153      You win!!
Game 13  Opponent: Rock     You: Rock         R:0.491080, P:0.195816, S:0.313103      Draw
Game 14  Opponent: Paper    You: Rock         R:0.586139, P:0.203905, S:0.209956      You lose
Game 15  Opponent: Paper    You: Rock         R:0.814554, P:0.026020, S:0.159426      You lose
Game 16  Opponent: Rock     You: Rock         R:0.696741, P:0.013689, S:0.289570      Draw
Game 17  Opponent: Rock     You: Paper        R:0.395465, P:0.460628, S:0.143907      You win!!
Game 18  Opponent: Paper    You: Paper        R:0.159063, P:0.508364, S:0.332573      Draw
Game 19  Opponent: Rock     You: Paper        R:0.105390, P:0.766344, S:0.128266      You win!!
Game 20  Opponent: Rock     You: Scissor      R:0.146979, P:0.167586, S:0.685436      You lose
Game 21  Opponent: Scissor  You: Scissor      R:0.273961, P:0.101969, S:0.624070      Draw
Game 22  Opponent: Rock     You: Rock         R:0.524342, P:0.056651, S:0.419007      Draw
Game 23  Opponent: Rock     You: Paper        R:0.308945, P:0.419330, S:0.271725      You win!!
Game 24  Opponent: Paper    You: Scissor      R:0.078199, P:0.336727, S:0.585074      You win!!
Game 25  Opponent: Rock     You: Rock         R:0.382448, P:0.303195, S:0.314357      Draw
Game 26  Opponent: Paper    You: Paper        R:0.080430, P:0.480732, S:0.438839      Draw
Game 27  Opponent: Rock     You: Rock         R:0.537649, P:0.190170, S:0.272182      Draw
Game 28  Opponent: Rock     You: Scissor      R:0.416978, P:0.121089, S:0.461933      You lose
Are you ready?
```

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.