10 Self-learning classification

10.1 experimental goals

This lesson mainly learns the K 210 self-learning classification function, takes out three different objects, takes pictures from different angles, and K 210 classifies them.

The reference code path for this experiment is： CanMV\05-AI\self_learning.py

10.2 Preparation before the experiment

Please first import the model file into the memory card, and then insert the memory card into the memory card slot of the K210 module.

10.3 experimental procedure

The factory firmware of the module has integrated the AI vision algorithm module. If you have downloaded other firmware, please burn it back to the factory firmware before doing the experiment.

1. Import the relevant libraries, initialize the camera and LCD display, and load the model file:/sd/KPU/self_learn_classifier/mb-0.25.kmodel。

```
kpu = KPU()
print("ready load model")
kpu.load_kmodel("/sd/KPU/self_learn_classifier/mb-0.25.kmodel")
```

2. The whole routine is divided into three states, the first state (INIT) is the initialization state, the second state (CLASSIFY) is the classification state, and the third state (TRAIN) is the training state. Each state corresponds to a loop function running the program content.

```
if state_machine.current_state == STATE.INIT:
        loop_init()
    elif state_machine.current_state == STATE.CLASSIFY:
        loop_classify()
    elif state_machine.current_state == STATE.TRAIN_CLASS_1 \
            or state_machine.current_state == STATE.TRAIN_CLASS_2 \
            or state_machine.current_state == STATE.TRAIN_CLASS_3:
        loop_capture()
```

3. The main function of the loop function content of the initialization state is to display prompt information, press BOOT briefly to proceed to the next step, and press the BOOT key long to

restart.

```
def loop_init():
    if state_machine.current_state != STATE.INIT:
        return

    img_init.draw_rectangle(0, 0, lcd.width(), lcd.height(), color=(0, 0, 255),
fill=True, thickness=2)
    img_init.draw_string(65, 90, "Self Learning Demo", color=(255, 255, 255),
scale=2)
    img_init.draw_string(5, 210, "Short press:  next", color=(255, 255, 255),
scale=1)
    img_init.draw_string(5, 225, "Long press:    restart", color=(255, 255, 255),
scale=1)
    lcd.display(img_init)
```

4. The content of the loop function of the training state, the main function is to display the training category and the progress of the operation. In the training state, you need to press the BOOT button several times to proceed to the next step. By default, a total of three classes are trained, and each class needs to take 5 pictures.

```
def loop_capture():
    global central_msg, bottom_msg
    img = sensor.snapshot()
    if central_msg:
        img.draw_rectangle(0, 90, lcd.width(), 22, color=(0, 0, 255), fill=True,
thickness=2)
        img.draw_string(55, 90, central_msg, color=(255, 255, 255), scale=2)
    if bottom_msg:
        img.draw_string(5, 208, bottom_msg, color=(0, 0, 255), scale=1)
    lcd.display(img)
```

5. The main function of the loop function content of the classification state is to analyze and compare, classify which category the current camera picture belongs to, and display the corresponding category and recognition score.

```
def loop_classify():
    global central_msg, bottom_msg
    img = sensor.snapshot()

    scores = []
    feature = kpu.run_with_output(img, get_feature=True)
    high = 0
    index = 0
    for j in range(len(features)):
        for f in features[j]:
            score = kpu.feature_compare(f, feature)
            if score > high:
                high = score
                index = j
    if high > THRESHOLD:
        bottom_msg = "class:{},score:{:2.1f}".format(index + 1, high)
    else:
```

```
        bottom_msg = None

    # display info
    if central_msg:
        print("central_msg:{}".format(central_msg))
        img.draw_rectangle(0, 90, lcd.width(), 22, color=(0, 255, 0), fill=True,
thickness=2)
        img.draw_string(55, 90, central_msg, color=(255, 255, 255), scale=2)
    if bottom_msg:
        print("bottom_msg:{}".format(bottom_msg))
        img.draw_string(5, 208, bottom_msg, color=(0, 255, 0), scale=1)
    lcd.display(img)
```

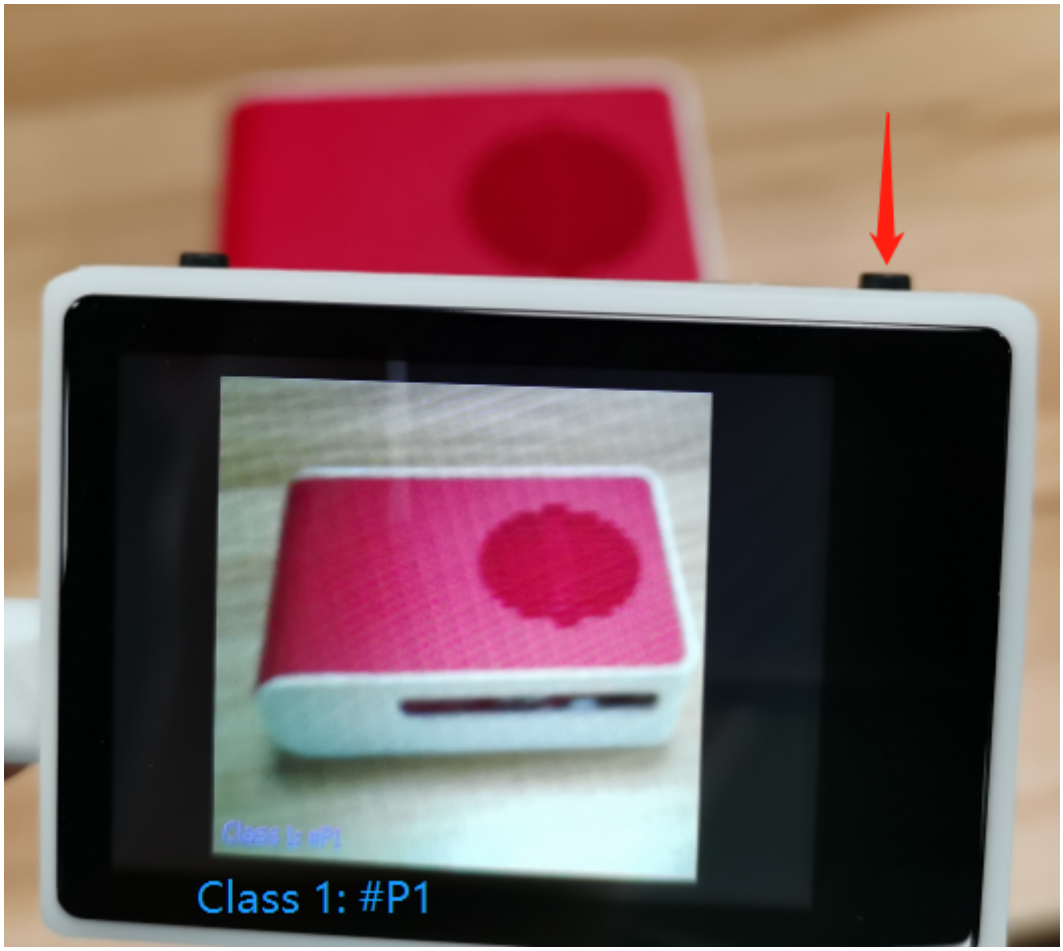
10.4 experimental results

Since the BOOT button is required, do not run it directly in the CanMV IDE, the CanMV IDE cannot detect the BOOT button at present, please download the code as a main.py to the K210 module to run.

Wait for the boot to complete, the LCD screen displays 'Self Learing Demo', prompts you to press to enter the next state, long press to restart. At this point, press the BOOT button once to enter the next state.

Start training the first object. Press the BOOT button once to go to the next step and start collecting object image materials.

Put the first object to be identified in the camera's field of view, you can see 'Class 1: P1' displayed in the lower left corner of the screen, press the BOOT button, and record the first stock image of category 1. where Class represents the category serial number, and P represents the picture serial number.



Change the shooting angle slightly, and then press the BOOT button again to capture the second footage. When five images have been collected, the category 1 collection ends.

Next, enter the collection process of category 2, press the BOOT button to start collecting the material images of category 2, the steps are the same as those of category 1.

After collecting five stock images in category 2, go to category 3 to collect five stock images.

After the collection is completed, it automatically enters the classification state. A 'Classificatiion' display in the middle of the screen indicates that it has entered the classification state, and pressing the BOOT button again will automatically disappear the prompt.

At this point, the camera will shoot one of the three categories just trained, and the corresponding category ordinal number and score will be displayed in the lower left corner of the screen.


class:1,score:99.0


class:2,score:98.6

If the recognition is not good, or the image acquisition is wrong, you can press reset once, or press and hold the BOOT button to restart.

10.5 experiment summary

Face recognition needs to use a memory card to load the model file, so you need to import the model file into the memory card in advance, and then insert the memory card into the memory card slot of the K 210 module, if the model file in the memory card cannot be read, an error will be reported.

Since face recognition requires the BOOT button, do not run the face recognition code in the CanMV IDE, download the code as a main.py to the K210 chip, and then press the reset button to start running.

When training object classification, it is best to train objects that are different from the background color, so that the accuracy will be better. Due to the limited number of pictures collected, classified products generally take the front side as an example, and the reverse side may not be recognized.