8.10、自主学习分类

8.10、自主学习分类
- 8.10.1、实验目标
- 8.10.2、实验前准备
- 8.10.3、实验过程
- 8.10.4、实验效果
- 8.10.5、实验总结

8.10.1、实验目标

本节课主要学习K210自主学习分类功能，拿出三个不同的物体，拍摄不同角度的图片，K210进行分类。

本次实验的参考代码路径为：K210_Broad\05-AI\self_learning.py

8.10.2、实验前准备

请先将模型文件导入内存卡上，再将内存卡插入到K210开发板的内存卡插槽上。具体操作步骤请参考：

附录：导入模型文件到内存卡

8.10.3、实验过程

模块的出厂固件已经集成AI视觉算法模块，如果下载过其他固件，请烧录回出厂固件再进行实验。

1. 导入相关库，并初始化摄像头和LCD显示屏，加载模型文件：/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. 整个例程分为三种状态，第一种状态（INIT）是初始化状态，第二种状态（CLASSIFY）是分类状态，第三种状态（TRAIN）是训练状态，每种状态对应一个loop函数运行程序内容。

```
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. 初始化状态的loop函数内容，主要功能是显示提示信息，短按BOOT进行下一步，长按BOOT键重新开始。

```
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. 训练状态的loop函数内容，主要功能是显示训练的类别和操作的进度。在训练状态中，需要多次按下BOOT按键进行下一步。默认情况下，总共训练三个类别，每个类别需要拍摄5张图片。

```
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. 分类状态的loop函数内容，主要功能是进行分析对比，分类出当前摄像头画面是属于哪一种类别，显示对应的类别和识别分数。

```
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)
```

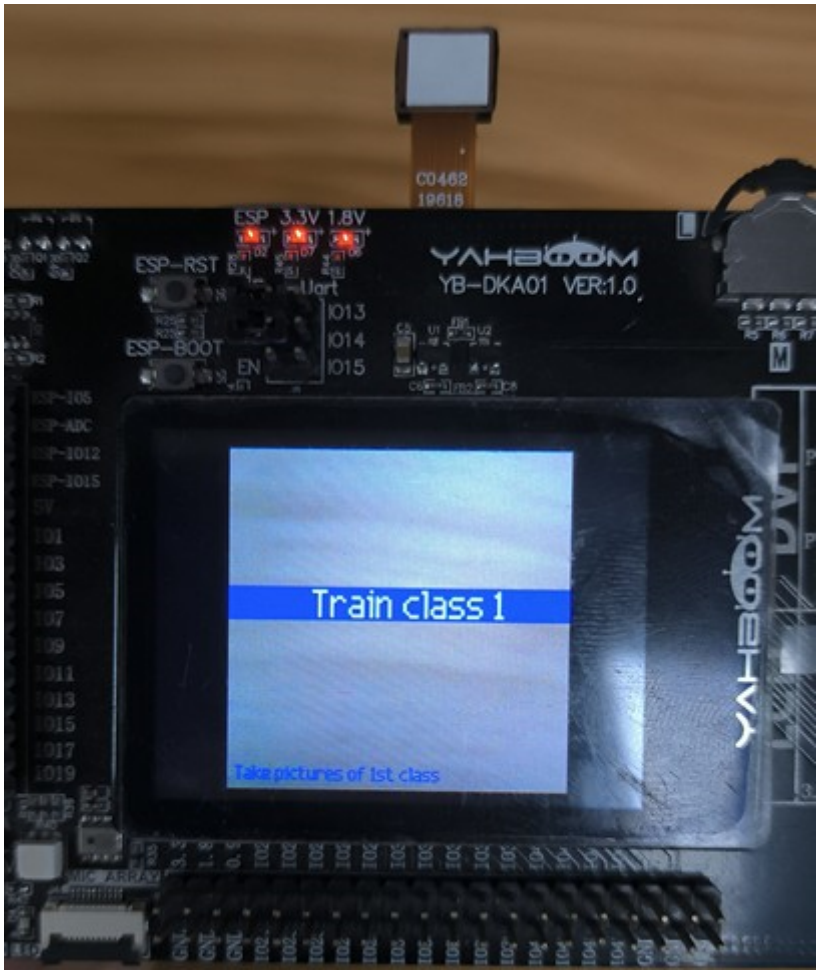
```
    if bottom_msg:
        print("bottom_msg:{}".format(bottom_msg))
        img.draw_string(5, 208, bottom_msg, color=(0, 255, 0), scale=1)
    lcd.display(img)
```

8.10.4、实验效果

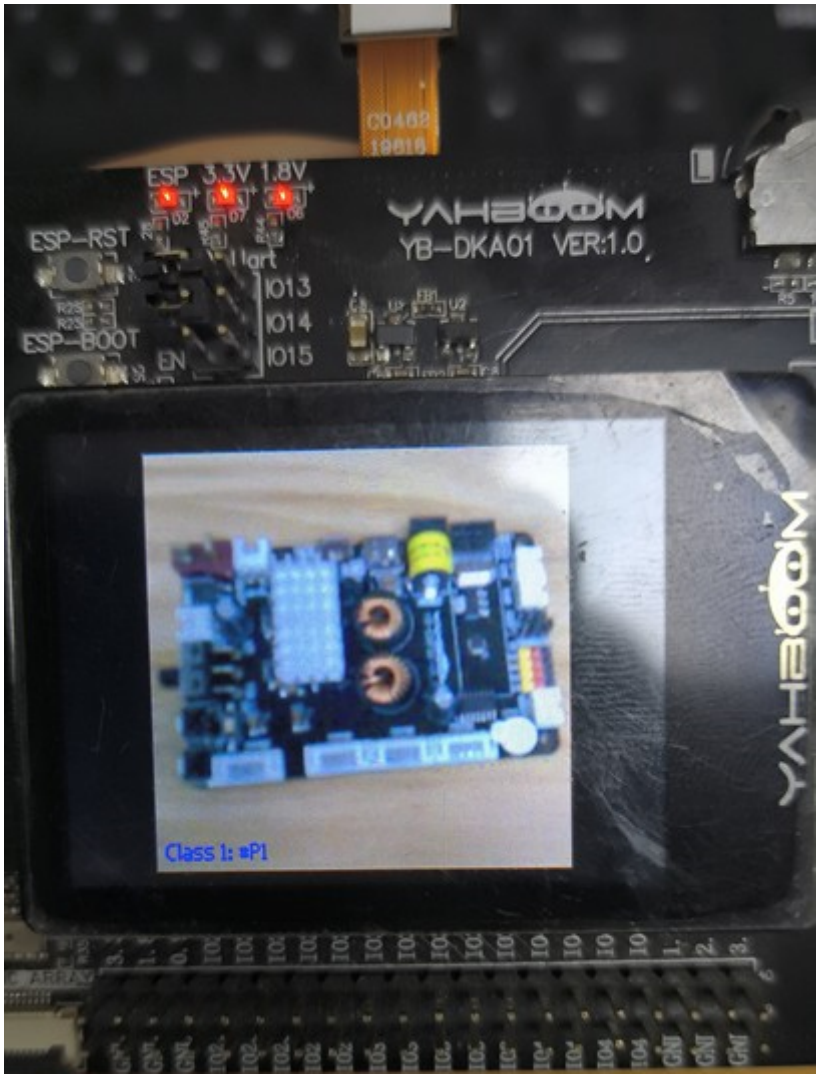
由于需要用到BOOT按键，不要在CanMV IDE里直接运行，CanMV IDE目前无法检测到BOOT按键，请将代码作为main.py下载到K210开发板上运行。

等待开机完成，LCD屏幕显示`Self Learing Demo`，提示按一下进入下一个状态，长按重新开始。此时按一下BOOT按键进入下一个状态。

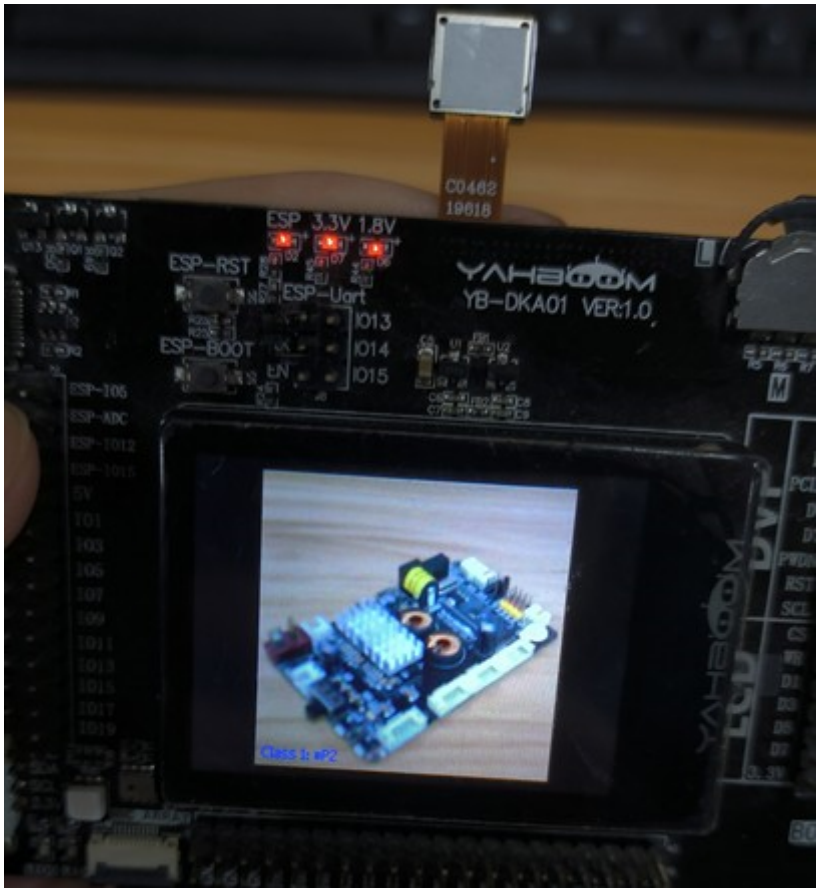
开始训练第一个物体。按一下BOOT按键进入下一步，开始采集物体图片素材。



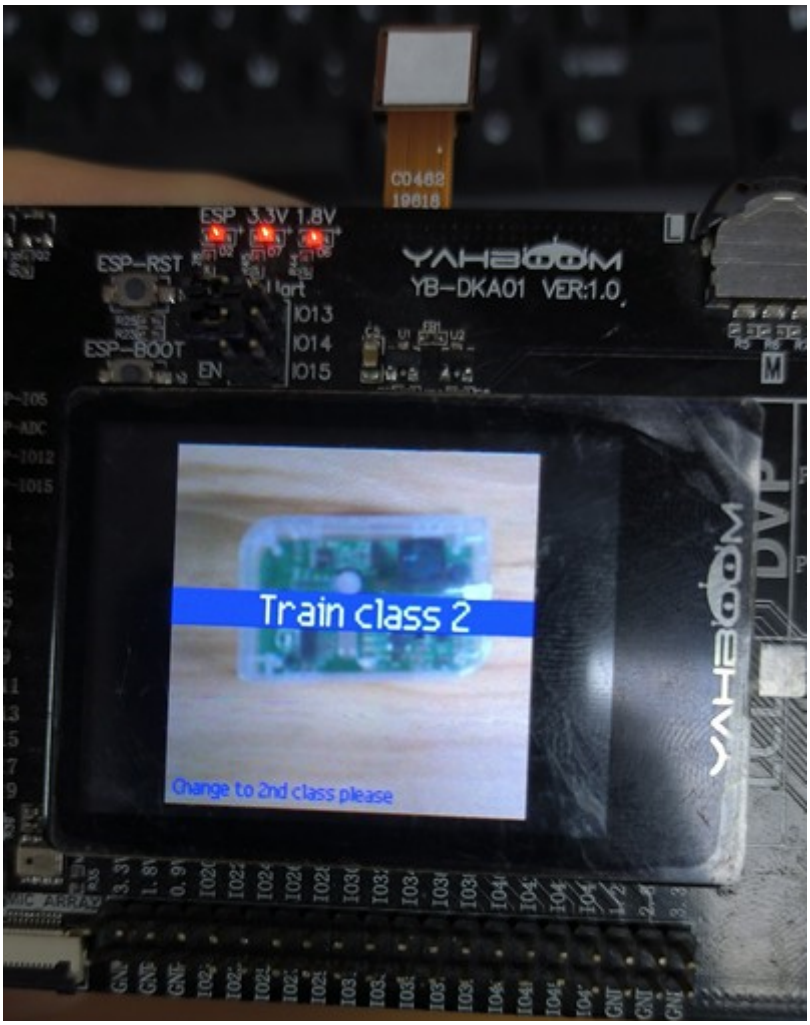
将第一个要识别的物体放到摄像头可视范围内，可以看到屏幕左下角显示`Class 1: #P1`，按一下BOOT按键，则记录类别1的第1个素材图片。其中Class表示类别序号，P表示图片序号。



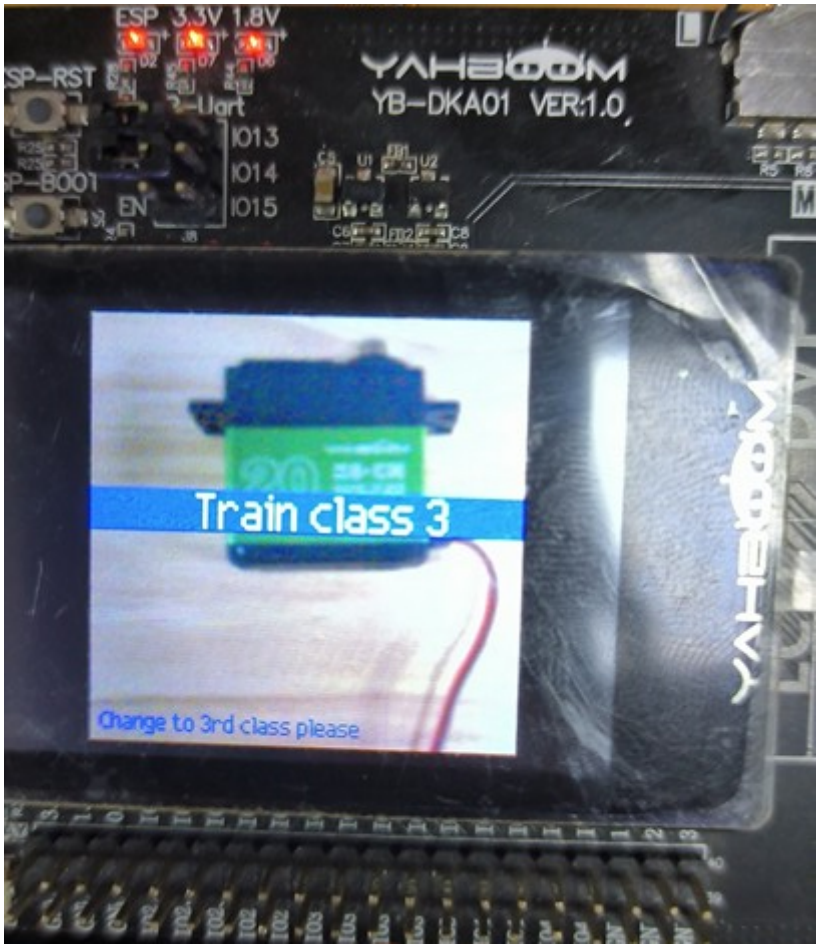
稍微转变一下拍摄角度，再按一下BOOT键，采集第二张素材图片。当采集了五张图片后，类别1采集结束。



接下来进入类别2的采集过程，按一下BOOT键开始采集类别2的素材图片，步骤与采集类别1的相同。

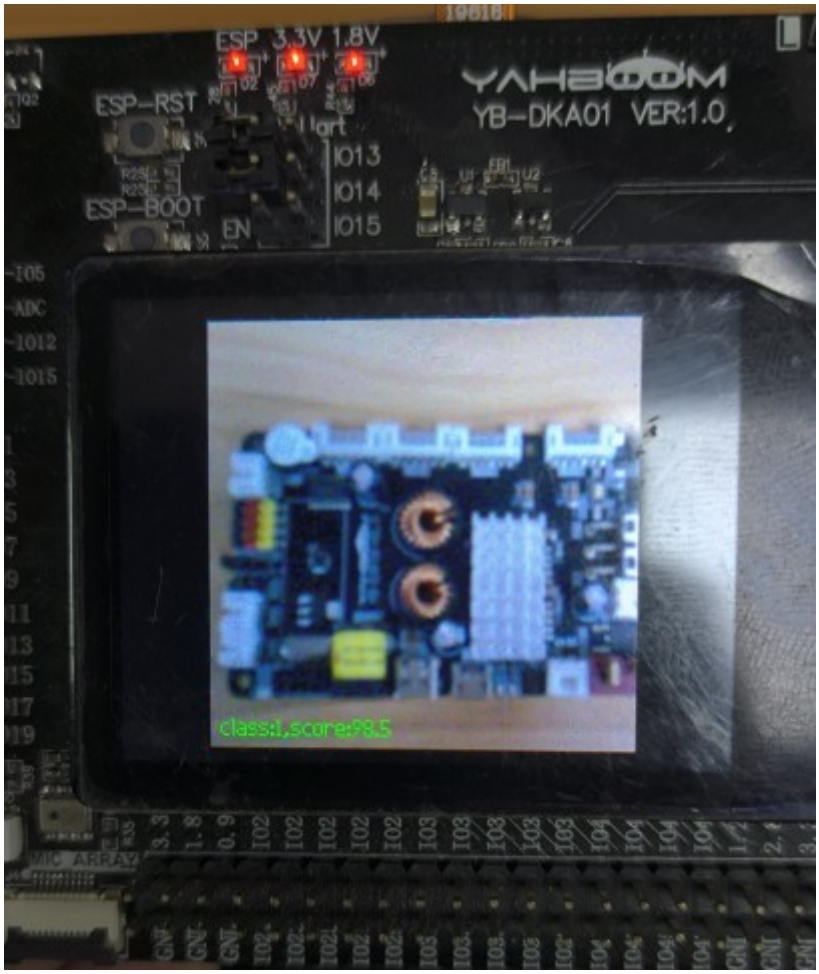


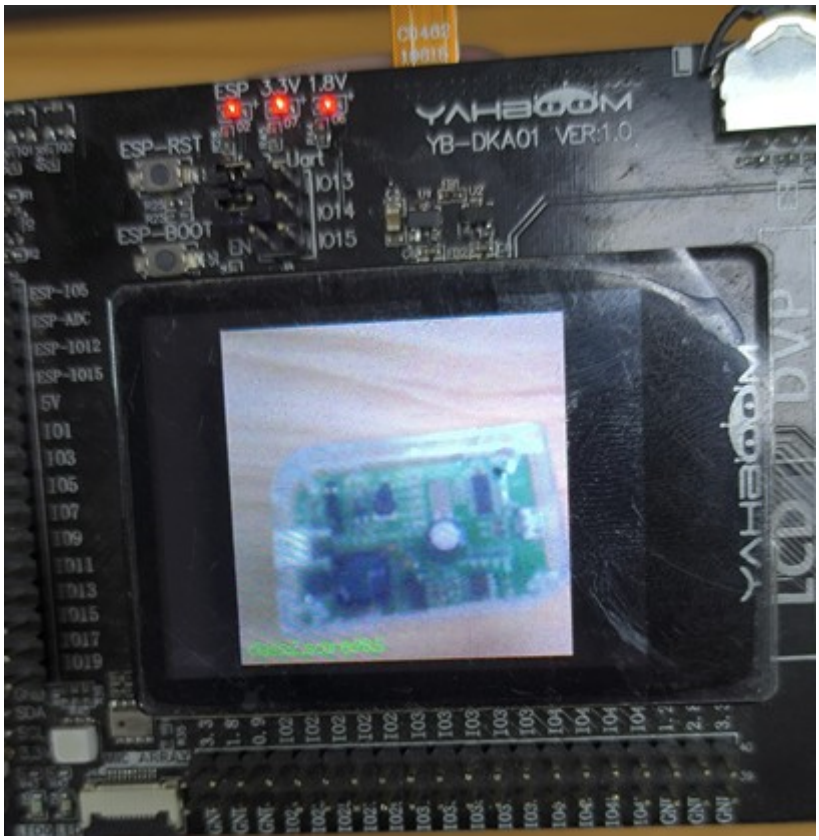
类别2采集完五张素材图片后，进入类别3采集五张素材图片。

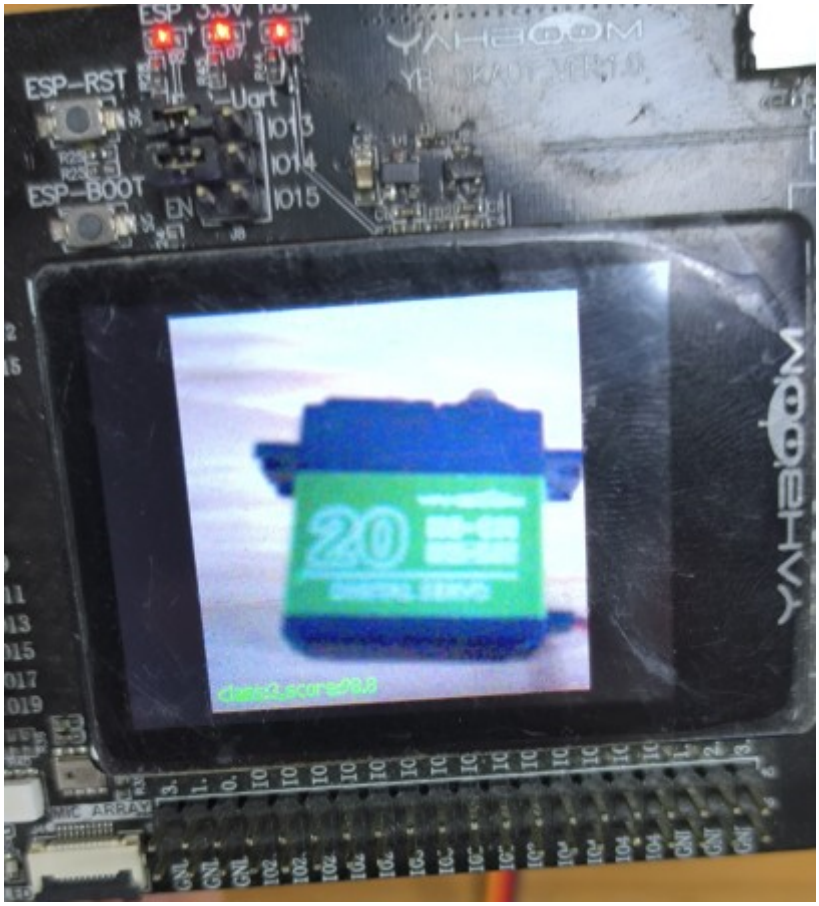


结束采集后自动进入分类状态。屏幕中间显示`Classificatiion`表示已经进入分类状态，再按一次BOOT按键自动消失提示。

此时将摄像头拍摄刚刚训练的三个类别的其中一个，屏幕左下角会显示对应类别序号和分数。


ESP 3.3V 1.8V
ESP-RST
ESP-BOOT
EN
Uart
IO13
IO14
IO15
YAHBOOM
YB-DKA01 VER:1.0
class:1,score:98.5


ESP 3.3V 1.8V
ESP-RST
ESP-BOOT
EN
Uart
IO13
IO14
IO15
YAHBOOM
YB-DKA01 VER:1.0
LCD / DVP



如果识别效果不好，或者图像采集出错，可以按一下复位，或者长按BOOT按键重新开始。

8.10.5、实验总结

人脸识别需要用的内存卡加载模型文件，所以需要提前将模型文件导入内存卡，再将内存卡插入K210开发板的内存卡卡槽里，如果无法读取到内存卡里的模型文件，则会报错。

由于人脸识别需要用到BOOT按键，所以不要在CanMV IDE运行人脸识别代码，请将代码作为main.py下载到K210芯片上，然后按复位键 开始运行。

物体分类训练时，最好训练的物体与背景颜色差别一些，这样准确率会更好一些。由于图片采集数量有限，分类产品一般以正面为例，反面可能就识别不出。