智能门锁

陈子翔，孙卓阳，张子阳

第一部分 设计概述

1.1 设计目的

智能门锁利用指纹或人脸等生物信息凭证，解决了传统机械锁的钥匙等物理凭证易丢失、被盗用的问题。然而，市面上的智能门锁往往只注重识别准确度，忽视了被对抗样本等手段攻破的风险，且不为使用者所知。

为了提升门锁安全性，本作品利用 Hi3516DV300 的 AI 视觉套件采集开锁者视频、完成人脸检测识别，并与 Hi3861V100 的 WiFi-IoT 套件互联通信、控制解锁，将试图开锁的实时画面发送到用户移动端，最终形成一套安全有效、智慧互联的门锁解决方案。

1.2 应用领域

在万物互联的时代，智能门锁作为不同领域场景下的第一道“关口”，承担着堡垒的作用。

1. 商务办公场景：门锁只对前期录入脸部信息的员工打开，方便安全。
2. 家用住宅：陌生人无法解锁智能锁且无法通过物理手段强行开锁，当遭遇非法开启时，智能锁将第一时间将警报发送至家庭主人手机中，解决了传统机械锁的不安全性。
3. 公寓酒店场景：公寓中可远程控制对智能门锁进行远程授权，远程控制门的开关；并只有授权人员才能打开进入各单元，保证了业主的安全。酒店环境下门锁为套房管理提供多种入住和开门方式；满足任意开门需求；免去客人和相关工作人员的麻烦。

1.3 主要技术特点

（1） 人脸分类网 Resnet 的 NAIE 训练

我们采用 Resnet 神经网络进行人脸识别。为便于实际验证，我们采集三位队员的人脸图片制作数据集，并尝试利用 LFW、CASIA-FaceV5 等开源人脸数据集提升模型的泛化能力。为提升效率，我们利用 NAIE 平台进行模型训练，并利用 pytorch2caffe 工具自动转换为 Caffe 模型。

（2） Caffe 模型量化、plug 插件开发与板端部署

我们在修改.prototxt.txt 适配网络层、利用 Netscope 确认网络层正确的基础上，通过 RuyiStudio 完成 Caffe 模型量化、生成 wk 文件。我们参考性别分类的插件代码，开发了人脸识别插件的相关接口、制作了相应的 makefile 文件，从而得到 plug 文件。我们在 hiopenais 环境下编译生成文件系统，并通过 HiTool 烧录到 Taurus，从而实现模型的板端部署。

（3） Taurus 与 Pegasus 的互联通信

我们利用 hisignalling 协议完成 Taurus 与 Pegasus 的互联通信。我们参考性别分类的 hisignalling 代码，设计了 Taurus 端发送人脸类别数据的逻辑

与格式、Pegasus 端接收与利用数据解锁的逻辑，从而实现 Taurus 识别特定人脸后引起 Pegasus 控制解锁的功能。

1.4 关键性能指标

（1） 识别准确率

理论准确率由模型在测试集上的推理结果给出，通过推理结果与图片标签一致的比例得到。

实际准确率由摄像头识别结果与实际人脸的相符程度给出，通过一段时间内 Taurus 识别人脸的分类结果与实际人脸一致的比例得到。

（2） 识别稳定性

同一人脸在不同角度、背景、光线下的识别结果保持不变，则可认为具有识别稳定性。

实际识别时，很难保证诸多因素变化下的模型推理结果完全不变，但也绝不能因此而错误地解锁。为了保证系统鲁棒性，我们对离散的分类结果设定了连续性要求（见创新点 2）。

（3） 运行稳定性

门锁系统模块连线、组装稳定，不脱落散乱。

在供电充足的前提下，系统各模块能长期工作。

（4） 响应速度

从拥有权限的人脸对准摄像头时刻，到系统解锁时刻的时间间隔，直接反映响应速度。

1.5 主要创新点

（1） 我们在 python 中调用 CV2 库，对录制好的队员视频进行图片提取，同时指定照片提取大小 256*256，获得所需数量的队员照片和背景图片数据集，效率效果大大提高。

（2） 为防止瞬间识别人脸失误，我们设置了一定的开锁阈值，套件必须在一段连续的时间内成功识别出同一个人，才能执行开锁程序，否则尽管识别出特定个人也无法开锁，有效提高了开锁的精度，防止陌生人随意开锁。

实现途径：对队员识别次数设置三个计数器，每识别一次某位队员，计数器+1，识别另一人时，计数器清零。当连续识别同一位队员多次，即计数器达到某一开锁阈值时，锁成功开启。

（3） 我们精心设计了智能门锁的整体外观。首先我们利用 sketchup 软件绘制草图，对门锁进行 3D 建模，在建模过程中综合考虑走线、零件摆放位置、实际操作可能性和合理性，直至整体效果不错，操作可行。之后我们利用 coredraw 软件对门锁进行平面设计，依据上一步 3D 建模结果测量门锁实际尺寸，得到门锁的粗略设计图。我们采用的材质是塑料玻璃，设计完成后我们对塑料玻璃进行切割、拼装，填充以具体门锁部件查看实际检测效果与整体观感。组员给出修改意见进行下一版制作，直至尺寸合理、功能完善。制作完成后对门锁细节进行手工打磨。

（4） 为优化门锁外观，让锁适配外观设计，同时解决原电子锁管脚部件接触不良的问题。我们更换了大小更为合适、性能更强的电子锁。

第二部分　系统组成及功能说明

2.1 整体介绍

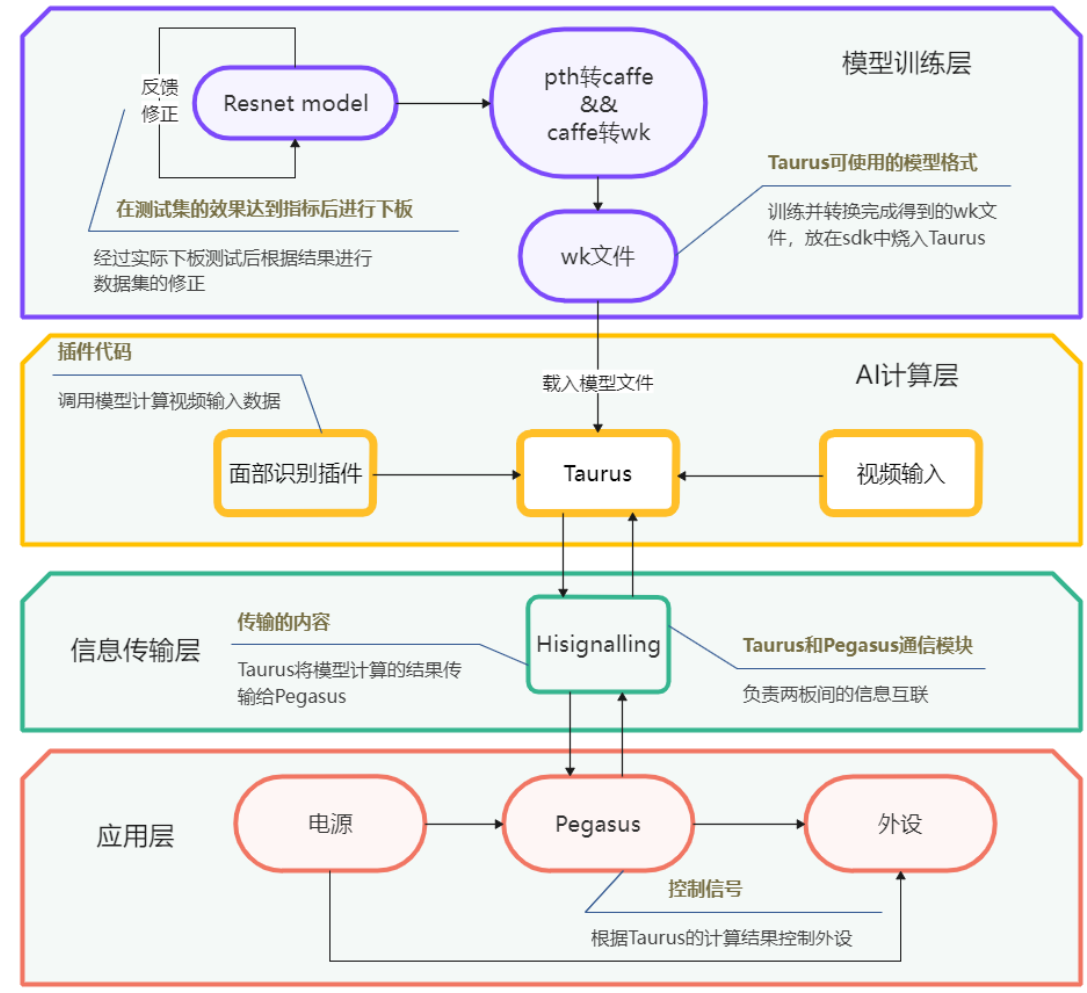
整体系统由四大部分组成，分别是模型训练、AI 计算、信息传输和应用层。

模型训练层主要包括模型训练以及模型转化两部分。模型训练部分我们利用 resnet18 网络对导入的人脸照片数据集进行训练并根据结果反馈修正。模型转化部分为将训练所得模型转为 caffe，进一步形成 wk 文件。

AI 计算层包括模型的调用和结果的计算，视频流的输入以及信息的传输。接收输入的视频数据并调用模型实时计算当前的类别，再将结果给到传输层。

信息传输层主要用于接收 Taurus 计算出的结果并传输给 Pegasus。

应用层为实际操作层，Pegasus 板根据从信息传输层获取的 Taurus 板的计算结果，经过一系列的控制条件判断，对锁的开闭进行控制。



2.2 各模块介绍

（1） 模型训练层

模型训练层包括模型训练与模型转化。在模型训练之前，需要进行数据采集以及数据集制作。我们采用 cv2 库对视频进行人脸照片提取；再对数据集进行分类制作数据集完毕。后续我们创建项目，利用 resnet 网络训练人脸识别模型，接着上传数据集，导入基于 pytorch 的开源代码，配置环境训练模型并进行归档管理。模型训练完毕后，平台上传 pth 模型，将模型

转化为 caffe 并获得模型中的.caffemodel 及.prototxt 文件，再转化得到 wk 文件。

（2） AI 计算层

计算层的核心是模型的推理计算，主要依赖 Taurus 集成的神经网络加速引擎 NNIE。NNIE 加载模型后，不断根据获取的数据进行推理计算，直到卸载模型。所需的数据输入，由我们制作的人脸识别插件从 Taurus 摄像头的视频流实时提供。推理得到的分类结果，经信息传输层发送到 Pegasus 端。

（3） 信息传输层

Taurus 和 Pegasus 的代码是分离的，所以为了做到传输信息的统一，两方的信号标志必须是对应的，所以我们将两板的信息传输相关结构体设计成一样。因为本项目中只需要 Taurus 向 Pegasus 传送计算结果，所以只在 Taurus 端设置了发送功能，Pegasus 端设置了接收功能。

（4） 应用层

应用层是实际操作的模块，由 Pegasus 负责总控。主要构成为电源、Pegasus 以及一个电子锁。Pegasus 控制的内容包括 LED 控制逻辑、计算结果的接收、输出控制信号。LED 用于显示开锁的次数以及工作状态；接收到计算结果后送至控制信号的产生模块；控制信号由多级的判断逻辑给出，优先级最高的是锁的物理开闭状态，在门开的情况下板端需要屏蔽计算结果，只有关锁的状态下正确的人脸识别结果才可以开锁。

第三部分 完成情况及性能参数

3.1 设计与改进

（1）设计初稿

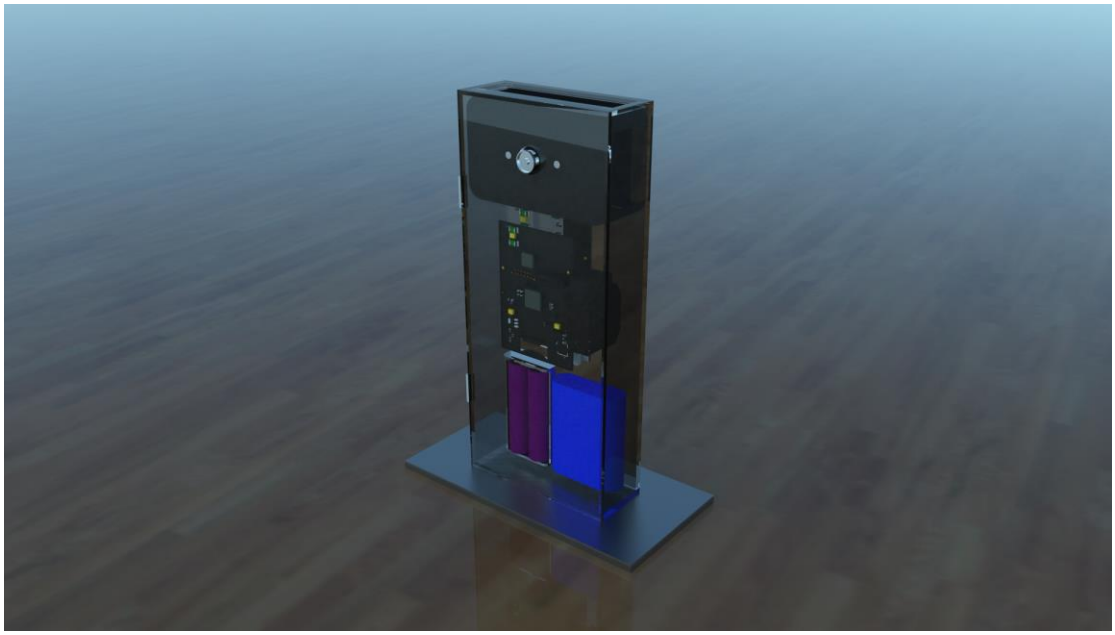
我们设计的初稿为木制快递柜，由左侧的木片作为开锁的门，快递柜内置电子锁。

当摄像头识别到指定开锁人后，电子锁打开，快递柜开门。

（2）外观改进

在初版快递柜完成后，我们确保了基本功能的实现，接下来对外观进行了设计打磨。首先我们利用 SketchUp 软件绘制草图，对门锁进行 3D 建模，在建模过程中综合考虑走线、零件摆放位置、实际操作可能性和合理性，直至整体效果不错，操作可行。

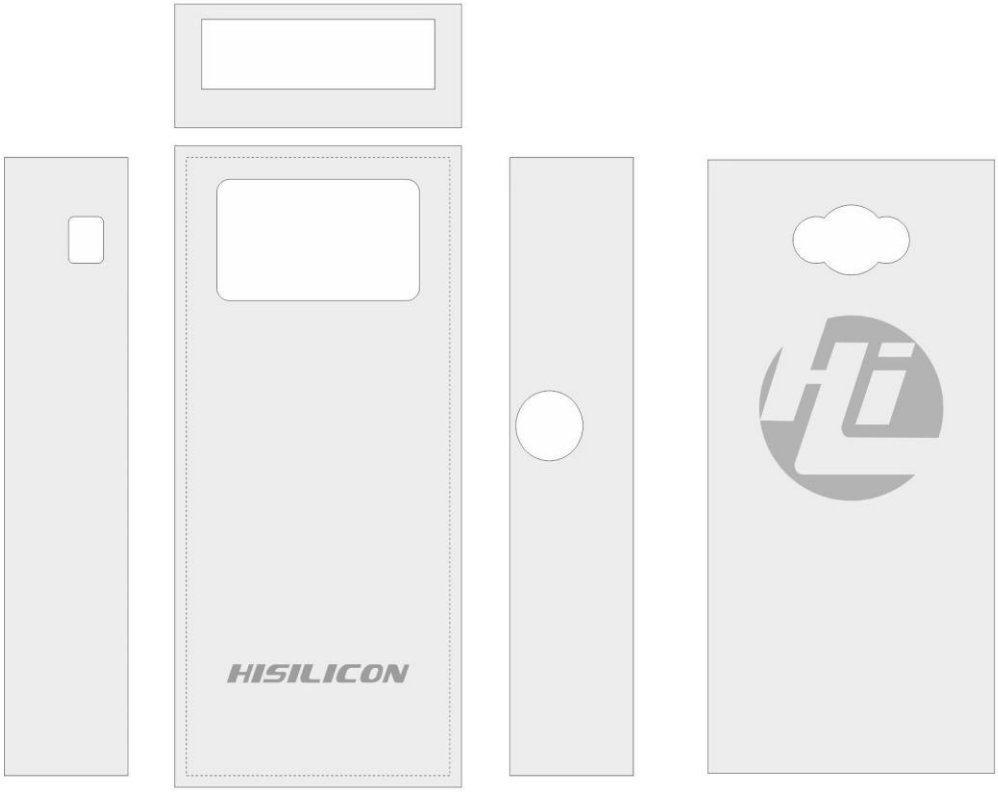
3D 模型渲染图如下所示。



之后我们利用 coredraw 软件对门锁进行平面设计，依据上一步 3D 建模结果

测量门锁实际尺寸，得到门锁的粗略设计图。我们采用的材质是塑料玻璃，设计完成后我们对塑料玻璃进行切割、拼装，填充以具体门锁部件查看实际检测效果与整体观感。组员给出修改意见进行下一版制作，直至尺寸合理、功能完善。制作完成后对门锁细节进行手工打磨。

2D 平面设计图如下所示。



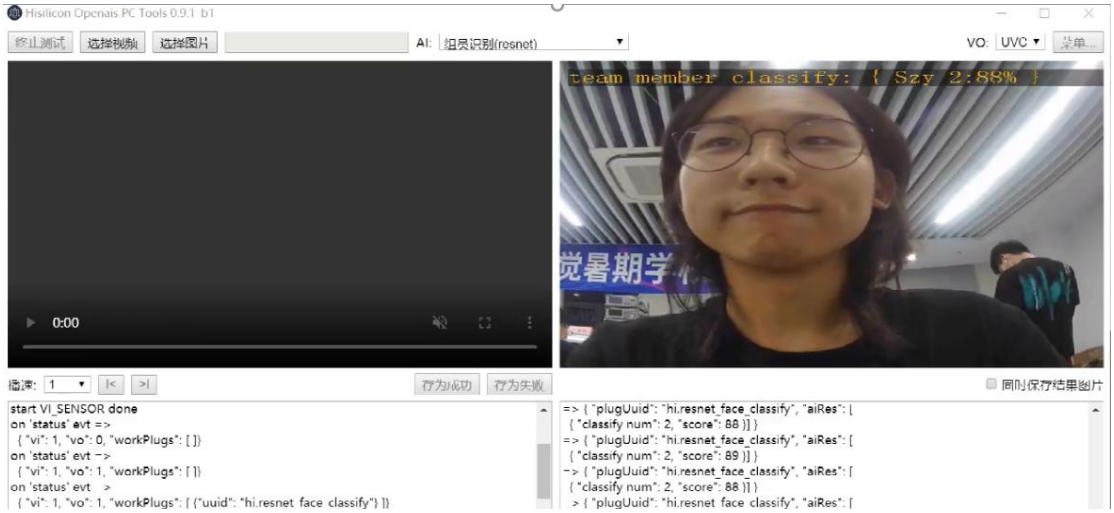
门锁系统的内部分为三大块：最底部安装套件所需电池，中部为 Taurus 开发板以及外围模块，最顶部为 Pegasus 开发套件。

全局正面展示、全局侧面展示如下页所示。

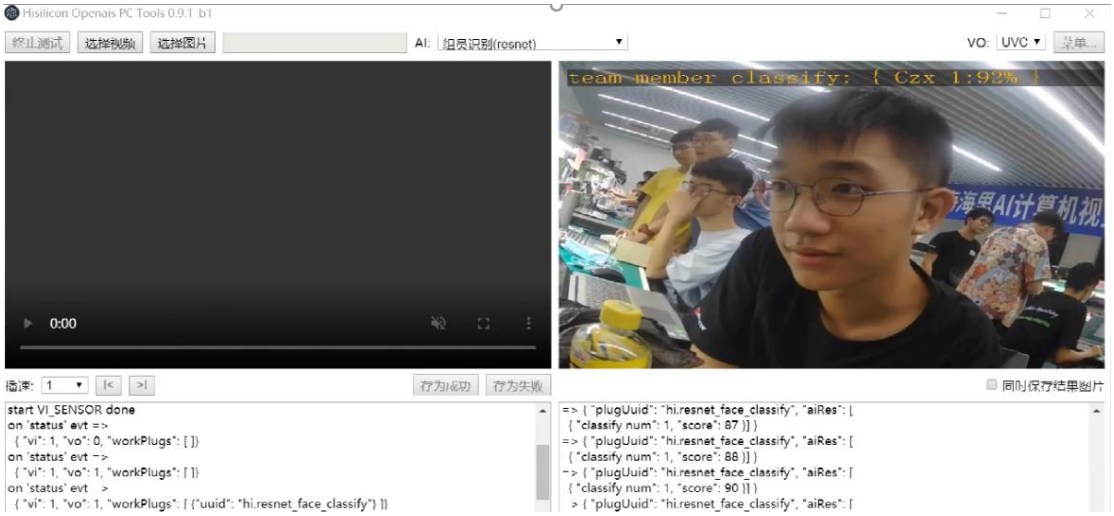

上海海思Hi3Spark开发套件| Taurus AI视频分析仪
MIC

3.2 实现功能

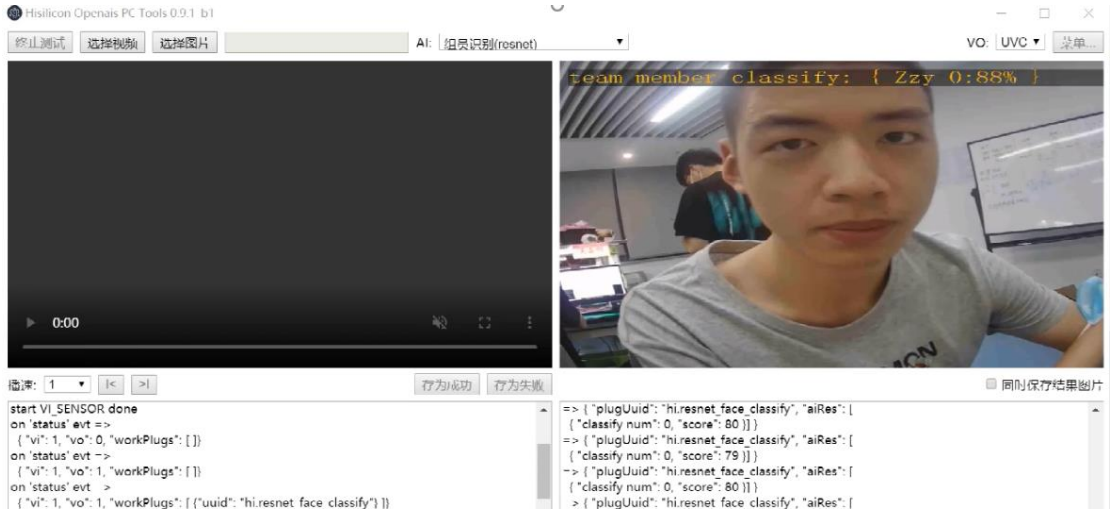
（1）准确识别小组三人



如上所示，识别组员孙卓阳，得分高达 88。

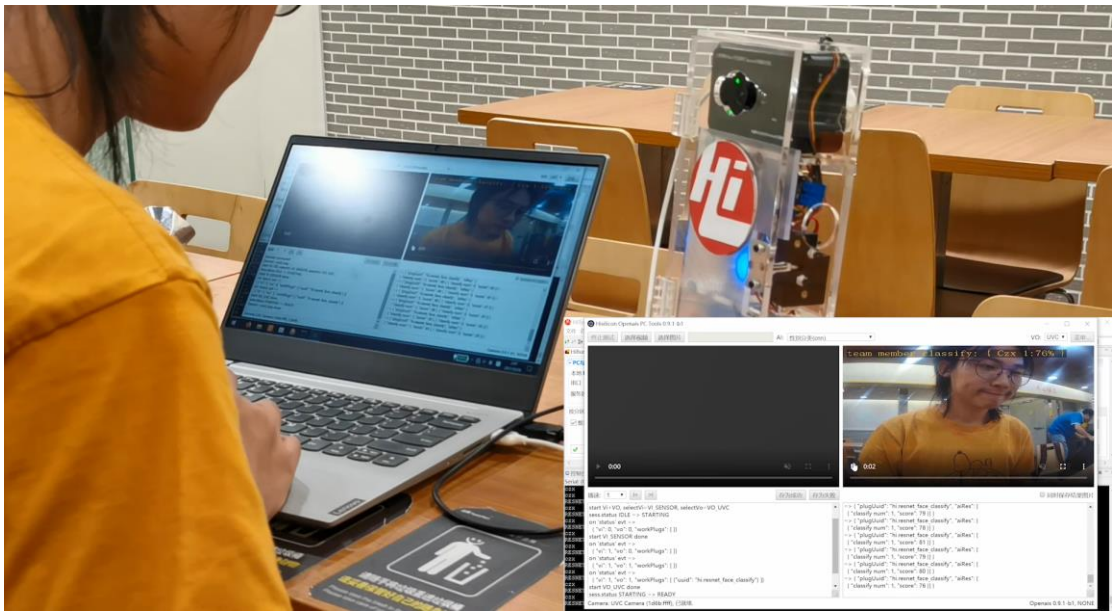


如上所示，识别组员陈子翔，得分高达 92。

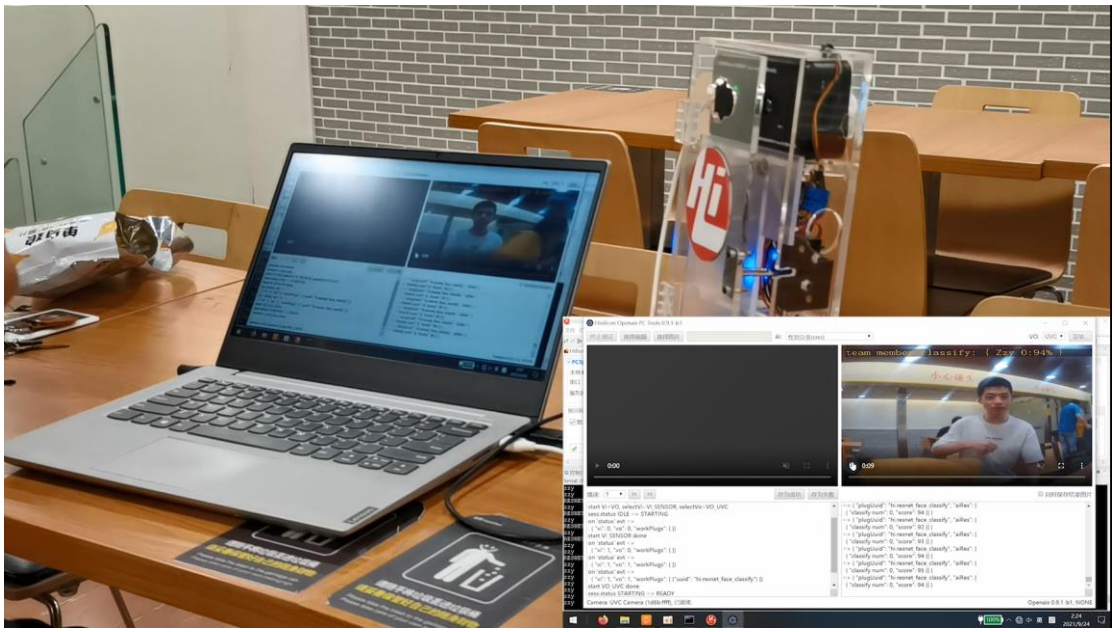


如上所示，识别组员张子阳，得分高达 88。

（2）设置特定开锁人



未识别到指定组员张子阳，门锁无动静，不开锁。



识别到指定组员张子阳，智能门锁开锁。

3.3 性能

根据 1.4 中定义的性能指标，系统性能情况如下。

（1） 识别准确率

理论准确率。我们最终部署的模型，训练时在测试数据集（从总数据集中划出 10%）上的准确率为 95%。

实际准确率 85%。我们利用 hiopenais 的 pc 端工具在人脸识别时打印的日志，观察 1 分钟内 Taurus 识别小组三位成员人脸的结果，并计算与实际人脸一

致的次数占总识别次数的百分比。

（2） 识别稳定性

为保证模型识别稳定性，我们在采集小组成员的人脸数据集时，有意识地让被采集人从各角度面对摄像头，并在数据集的制作脚本中随机调整了图片亮度，从而使数据集中的人脸角度、图片明暗富有变化。

实际识别的结果表明，在侧脸面对摄像头、光线增强或较暗的情况下，系统识别人脸的结果没有明显波动。在改变背景的情况下也是如此。因此可以认为系统具有较好的识别稳定性。

针对偶发波动，我们通过创新点 2 中的设计予以应对，最后也没有发生因为波动而错误解锁的情况。

（3） 运行稳定性

我们通过改进外观设计与组合设计，构建了稳定、不易散乱的系统，所有模块与布线都在装置内部、受外壳保护，不易受外界干扰。对于初始方案中，杜邦线连线接口不稳、接触不良的情况，我们针对性地改用双口排线，从而保证了模块间的通信稳定。

我们持续优化了锁控模块的代码逻辑，只在需要解锁时短暂拉高电平，保证了锁不会因持续工作而过热、烧坏，且更省电。

由于以上诸多设计，系统持续运行较长时间，仍能保持稳定工作。

（4） 响应速度

我们通过手机的秒表计时，测试系统的大致响应速度。

创新点 2 中基于稳定性考虑，设定了连续识别要求，导致解锁所需的时间下限达到 10*20*3 = 600ms。

我们测试了不同成员、不同角度识别的响应速度，得到的结果基本都在 1 秒以内。

第四部分 总结

4.1 可扩展之处

（1）人脸防欺骗

由于准备时间不够充分，本次并未实现防照片、视频欺骗的功能，所以本项目实现的门锁仍不够安全。

（2）更高的准确率

我们的人脸识别模型仍然有优化的余地，可以在训练集的大小和分类上进行改进，以实现更优的识别效果。

（3）更多的 AI 应用

与手机互联的 AI 增强应用可以加强用户体验，例如手机点播和远程操控。

4.2 心得体会

回想整个参赛过程，感触良多，收益颇丰。我们从最初确定参赛，查询资料与导师研究确定选题，制定详细的实施方案。报名成功拿到开发板后进行板端组装，环境配置，外观设计，代码编写，板端互联，模型训练与验证。期间遇到了各式各样的错误，我们一步步解决问题最终实现了相应功能。这是一次让我们得到锻炼与成长的经历。

在刚拿到开发套件后，我们参照海思工程师提供的各个文档操作，前期还算

顺利，遇到的一些问题在咨询了群内老师后也得到了解决。随着项目的推进，我们继续进行了 IOT 设计、板端互联、AI 算法基础应用、AI 分类网实现人脸识别开锁，难度也逐步攀升。在板端互联阶段，我们遇到了比较大的问题与瓶颈：数据传输不到位、数据传输错误等各式各样的错误，我们咨询工程师解决问题，整整花了两天时间才完成了两个套件的互联。再到后续 AI 分类网的设计与训练更是困难重重，一方面我们需要确保模型正常跑通，另一方面我们同时需要兼顾模型准确率，及时对数据集进行分析改进。整个过程比较艰难，但每完成一项目标成就感十足，收获满满。

参加这一次嵌入式大赛，我们对于海思 Hi3516DV300 的 AI 视觉套件以及 Hi3861V100 的 WiFi-IoT 套件有了比较深入的了解，同时对于嵌入式开发的基本流程有了初步的认识并初次尝试。在竞赛的各个阶段，组内成员们目标明确，分工合理，一步一个脚印向着既定目标前进，这次比赛也培养了我们的团队协作能力。总体来说，本次比赛不仅扩充了我们的知识储备，学习到了许多嵌入式开发的知识，同时将所学知识付诸实践，得以应用；同时，还提高了我们诸如资料查询、错误排查与处理、团队协作等软性实力，此次嵌入式大赛我们受益匪浅。

第五部分 参考文献

[1] Jifeng Dai, Yi Li, et al. R-FCN: Object Detection via Region-based Fully Convolutional Networks[C]. Advances in Neural Information Processing Systems. Curran Associates Inc. 2016.

第六部分 附录

重要代码、推导过程等不便于在正文中体现的内容

1. 板端代码

```
1. hi_void *led_control_demo(hi_void* param)
2. {
3.     hi_u32 ret;
4.     hi_gpio_value unlock;
5.
6.     hi_unuse_param(param);
7.     hi_gpio_init();
8.
9.     hi_gpio_get_input_val(HI_GPIO_IDX_12, &unlock);
10.
11.    /**
12.        book the times of one face showing consistantly
13.        记录某一分类的人脸连续出现的次数，超过 3 则视为成功识别并解锁
14.    */
15.    face_type last_face = UNKNOWN;
16.    static int book[4] = {0, 0, 0, 0};
```

```
17.
18.     for (;;)
19.     {
20.         // 通过 GPIO12 的电平高低，判断锁的开关状态 unlock
21.         hi_gpio_get_input_val(HI_GPIO_IDX_12, &unlock);
22.
23.         if(unlock)
24.         {
25.             // 锁开状态下，不执行开锁操作
26.             last_face = UNKNOWN;
27.         }
28.         else
29.         {
30.             // 只有在锁定状态下，进行相关判断
31.
32.             // 判断识别结果是否与上次一致，不一致则清空
33.             if (last_face != face_sort)
34.                 book[last_face] = 0;
35.             book[face_sort]++;
36.             last_face = face_sort;
37.
38.             if (face_sort == ZZY && book[ZZY] > 3) // 设置 ZZY 开
    锁
39.             {
40.                 // 将 GPIO9 电平拉高、开锁后，迅速拉低，防止锁烧坏
41.                 hi_gpio_set_ouput_val(HI_GPIO_IDX_9, HI_GPIO_VALU
    E1);
42.                 hi_sleep(power_sleep_100MS);
43.                 hi_gpio_set_ouput_val(HI_GPIO_IDX_9, HI_GPIO_VALU
    E0);
44.                 // 重置人脸分类结果和计数
45.                 face_sort = UNKNOWN;
46.                 book[ZZY] = 0;
47.             }
48.             else
49.             {
50.                 hi_gpio_set_ouput_val(HI_GPIO_IDX_9, HI_GPIO_VALU
    E0);
51.             }
52.         }
53.
54.         /*Release CPU resources for 20ms*/
55.         hi_sleep(10 * LED_CONTROL_TASK_SLEEP_20MS);
56.     }
```

```
    /*Delete task*/
    ret = hi_task_delete(g_led_control_id);
    if (ret != HI_ERR_SUCCESS)
    {
        printf("Failed to delete led control demo task\r\n");
    }
}
```

2. 数据集采集代码

```
import cv2
import dlib
import os
import sys
import random

video_path = 'member.mp4'
output_dir = './member_faces'
size = 256

if not os.path.exists(output_dir):
    os.makedirs(output_dir)


# 改变图片的亮度与对比度
def relight(img, light=1, bias=0):
    w = img.shape[1]
    h = img.shape[0]
    # image = []
    for i in range(0, w):
        for j in range(0, h):
            for c in range(3):
                tmp = int(img[j, i, c] * light + bias)
                if tmp > 255:
                    tmp = 255
                elif tmp < 0:
                    tmp = 0
                img[j, i, c] = tmp
    return img


# 使用 dlib 自带的 frontal_face_detector 作为我们的特征提取器
```

```
35.detector = dlib.get_frontal_face_detector()
36.# 打开摄像头 参数为输入流，可以为摄像头或视频文件
37.camera = cv2.VideoCapture(video_path)
38.
39.# 读帧
40.success, frame = camera.read()
41.
42.# 设置帧率
43.timeF = 2
44.count = 0
45.index = 0
46.
47.while success:
48.    count = count + 1
49.    if(count % timeF == 0 and index <= 10000):
50.        print('Being processed picture %s' % index)
51.        # 从摄像头读取照片
52.        success, img = camera.read()
53.        # 转为灰度图片
54.        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
55.        # 使用detector进行人脸检测
56.        dets = detector(gray_img, 1)
57.
58.        for i, d in enumerate(dets):
59.            x1 = d.top() if d.top() > 0 else 0
60.            y1 = d.bottom() if d.bottom() > 0 else 0
61.            x2 = d.left() if d.left() > 0 else 0
62.            y2 = d.right() if d.right() > 0 else 0
63.
64.            face = img[x1:y1, x2:y2]
65.            # 调整图片的对比度与亮度， 对比度与亮度值都取随机数，这样能增加样本的多样性
66.            face = relight(face, random.uniform(0.5, 1.5), random.randint(-50, 50))
67.
68.            # img = relight(img, random.uniform(0.5, 1.5), random.randint(-50, 50))
69.
70.            face = cv2.resize(face, (size, size))
71.
72.            cv2.imwrite(output_dir + '/' + 'member_faces_background' + str(index) + '.jpg', face)
73.            index += 1
74.        key = cv2.waitKey(30) & 0xff
```

```
        if key == 27:
            break

print('Finished!')
```

3. 分类网重要代码

```
# -*- coding: utf-8 -*-
from __future__ import print_function  # do not delete this line if you want to save your log file.
import moxing as mox
import zipfile
import os
import subprocess
from naie.context import Context


def load_data():

    from naie.datasets import get_data_reference
    data_reference = get_data_reference(dataset="faces_train", dataset_entity="faces_train")
    file_paths = data_reference.get_files_paths()
    mox.file.copy(data_reference.get_files_paths()[0], "/cache/faces_train.zip")
    rf = zipfile.ZipFile("/cache/faces_train.zip")
    rf.extractall("/cache/")
    rf.close()
    print(file_paths)


def main():
    config = './configs/resnet/resnet18_b32x8_imagenet.py'
    gpus = 8
    workdir = './ckpt'

    cmd = " {} {} --work-dir {} ".format(config, gpus, workdir)
    print(cmd)
    subprocess.call("cd mmclassification-master && pip install -e . ", shell = True)
    subprocess.call("cd mmclassification-master && sh ./tools/dist_train.sh " + cmd, shell = True)
```

```
31.
32.if __name__ == "__main__":
33.    load_data()
34.    main()
35.    mox.file.copy('/cache/user-job-dir/codes/mmclassification-
   master/ckpt/latest.pth', os.path.join(Context.get_model_path(), '
   latest.pth'))
36.    os.path.abspath('.')
37.    print("path:", os.path.abspath('.'))
38.    print("dir:", os.listdir(os.path.abspath('.')))
```