SmartMonitor 项目开发手册

项目负责人：张晨

项目交付日期：**2023** 年 **3** 月 **30** 日

目录

背景介绍

近几年，AI 视觉技术在安防、工业制造等场景在产业智能化升级进程中发挥着举足轻重的作用。【进出管控】作为各行业中的关键场景，应用需求十分迫切。 如在居家防盗、机房管控以及景区危险告警等场景中，存在大量对异常目标（人、车或其他物体）不经允许擅自进入规定区域的及时检测需求。利用深度学习视觉技术，可以及时准确地对闯入行为进行识别并发出告警信息。切实保障人员的生命财产安全。相比传统人力监管的方式，不仅可以实现 7*24 小时不间断的全方位保护，还能极大地降低管理成本，解放劳动力。

但是在真实产业中，要实现高精度的人员进出识别不是一件容易的事，在实际场景中存在着各种各样的问题：摄像头采集到的图像会受到建筑、机器、车辆等遮挡的影响；天气多种多样，要适应白天、黑夜、雾天和雨天等。

针对上述场景，本项目推出了重点区域人员进出管控实践示例，提供从数据准备、技术方案、模型训练优化，到模型部署的全流程可复用方案，有效解决了不同光照、不同天气等室外复杂环境下的图像分类问题，并且极大地降低了数据标注和算力成本，适用于厂区巡检、家居防盗、景区管理等多个产业应用。

项目阶段 I：模型准备

一、方案介绍

1. 模型研发方案

本案例是一个经典的图像二分类问题，即通过算法来判断图片中是否包含人物。对于该问题，最常见的迭代流程和方法是：数据收集->模型选择及研发->预测部署。本小节将介绍数据收集和模型如何选择和研发。

1.1 数据准备

数据集筛选自 MS-COCO 和 Object365 数据，其中图像中人的检测框面积大于 10%则认为该图像包含人，如果图像中没有人的 ID，则认为该图像中没有人。更具体的详细筛选方法可以参考 PaddleClas 文档。筛选出的图片如下图所示：

1.2 模型选择及研发

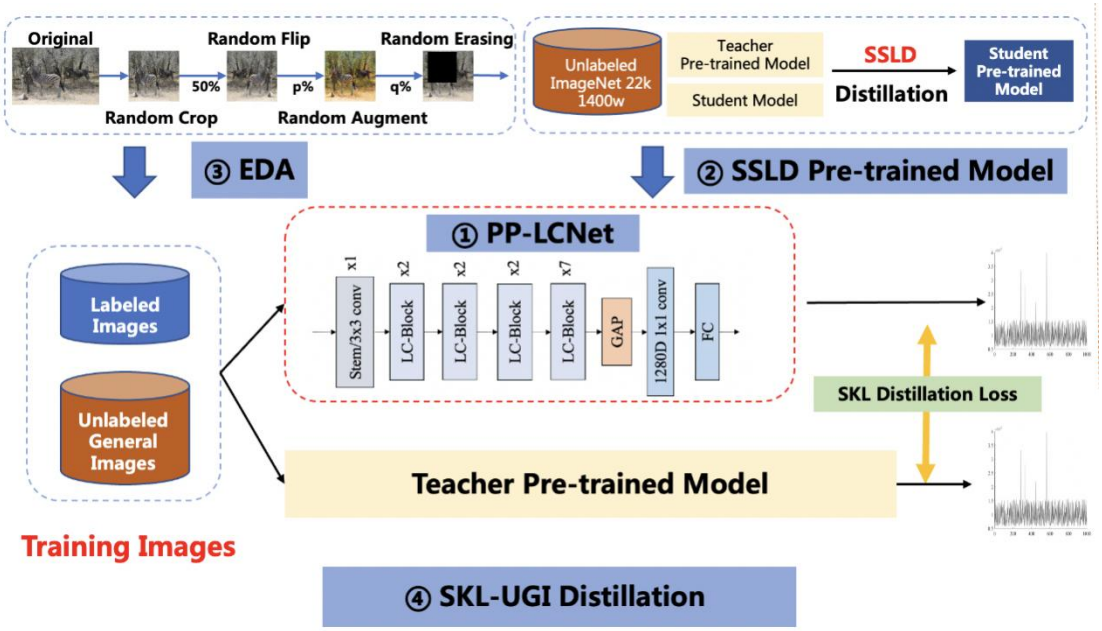
首先，对于该问题首先可以选择一个较好的骨干网络直接训练。

我们选择明星网络 SwinTransformer 进行训练，关于 SwinTransformer 的介绍可以参考 SwinTransformer 介绍。由于 SwinTransformer 的模型较大，推理速度也比较慢，不利于推理部署，所以也可以尝试更加轻量级的模型如 MobileNetV3，关于 MobileNetV3 的介绍可以参阅移动端系列网络介绍。

2. 方案优化思路

2.1 PULC 方案

采用 PaddleClas 的超轻量图像分类方案（Practical Ultra Light Classification，简称 PULC），实现对模型的快速优化，仅用超轻量模型就可实现与 SwinTransformer 模型接近的精度，同时预测速度提高 50 倍，极大地提升了部署性能。整体方案如图所示：



2.2 PP-LCNet 轻量级骨干网络

PP-LCNet 作为针对 CPU 量身打造的骨干网络模型，在速度、精度方面均远超如 MobileNetV3 等同体量算法，在有人/无人场景中，速度较 SwinTransformer 的模型快 40 倍以上，较 MobileNetV3_small_x0_35 快 20%。更多关于 PP-LCNet 的介绍，可以参考 PP-LCNet 介绍。

2.3 SSLD 预训练权重

SSLD 半监督蒸馏算法可以使小模型学习到大模型的特征和 ImageNet-22k 无标签大规模数据的知识。在训练小模型时，使用 SSLD 预训练权重作为模型的初始化参数，可以使有人/无人分类模型有 2.5 个点的精度提升。关于更多 SSLD 的介绍，可以参考 SSLD 介绍。

2.4 EDA 数据增强策略集成

该方案融合了图像变换、图像裁剪和图像混叠 3 种数据增强方法中，并支持自定义调整触发概率，能使模型的泛化能力大大增强，提升模型在实际场景中的性能。模型可以在上一步的基础上，精度再提升 1 个多点。

2.5 SKL-UGI 知识蒸馏算法

SKL(symmetric-KL)在经典的 KL 知识蒸馏算法的基础上引入对称信息，提升了算法的鲁棒性。同时，该方案可以方便的在训练中加入无标签训练数据（Unlabeled General Image），可以进一步提升了模型效果。该算法可以使模型精度继续提升两个多点。

2.6 结果分析

下表列出了此处训练得到所有的数据，通过对比我们可以看到，使用 PaddleClas 提供的 PULC，PP-LCNet_x1_0 模型的精度超越了 SwinTransformer_tiny 模型的精度，而且速度较 MobileNetV3_small_x0_35 更快，存储较 MobileNetV3_small_x0_35 更小，该模型可以方便、快速地部署到各种硬件设备上，在解决人员是否进入的问题基础上大大提升了预测效率。

模型	Tpr（%）	延时（ms）	存储（M）	策略
SwinTranformer_tiny	95.69	95.30	111	使用 ImageNet 预训练模型
MobileNetV3_small_x0_35	68.25	2.85	2.6	使用 ImageNet 预训练模型
PPLCNet_x1_0	89.57	2.12	7.0	使用 ImageNet 预训练模型
PPLCNet_x1_0	92.10	2.12	7.0	使用 SSLD 预训练模型
PPLCNet_x1_0	93.43	2.12	7.0	使用 SSLD 预训练模型+EDA 策略
PPLCNet_x1_0	**96.23**	**2.12**	**7.0**	使用 SSLD 预训练模型+EDA 策略+SKL-UGI 知识蒸馏策略

二、实践教程

1. 环境配置

1.1 新建项目文件夹“Detection”

（1）进入文件夹路径，例如: cd D:\Desktop\Program\Detection

（2）当前目录新建 data 和 work 两个文件夹，然后 ls 查看

1.2 环境要求

（1）PaddlePaddle >= 2.2.2

① 您的机器安装了 CUDA10 或 CUDA11，请运行以下命令安装：

python -m pip install paddlepaddle-gpu -i https://mirror.baidu.com/pypi/simple

② 您的机器是 CPU，请运行以下命令安装：

python -m pip install paddlepaddle -i https://mirror.baidu.com/pypi/simple

（2）PaddleClas >= 2.4

pip install paddleclas

（3）Python >= 3.7

更多的版本需求，请参照飞桨官网安装文档中的说明进行操作。

1.3 代码准备

（1）本案例需要克隆 PaddleClas 代码，首先需要解压：

```
cd work/
git clone -b release/2.4 https://gitee.com/paddlepaddle/PaddleClas.git
```

（2）安装依赖：

```
cd ./PaddleClas/
pip install -r requirements.txt
```

2. 数据准备

2.1 数据集来源：

本案例中所使用的所有数据集均为开源数据，train 集合为 MS-COCO 数据的训练集的子集，val 集合为 Object365 数据的训练集的子集，ImageNet_val 为 ImageNet-1k 数据的验证集。

2.2 数据集获取

在公开数据集的基础上经过后处理即可得到本案例需要的数据，具体处理方法如下：

（1）训练集合：

本案例处理了 MS-COCO 数据训练集的标注文件，如果某张图含有“人”的标签，且这个框的面积在整张图中的比例大于 10%，即认为该张图中含有人，如果某张图中没有“人”的标签，则认为该张图中不含有人。经过处理后，得到 92964 条可用数据，其中有人的数据有 39813 条，无人的数据 53151 条。

（2）验证集合：

从 Object365 数据中随机抽取一小部分数据，使用在 MS-COCO 上训练得到的较好的模型预测这些数据，将预测结果和数据的标注文件取交集，将交集的结果按照得到训练集的方法筛选出验证集合。经过处理后，得到 27820 条可用数据。其中有人的数据有 2255 条，无人的数据有 25565 条。

2.3 下载数据集

经过上述方法处理好的数据，可以直接下载：

（1）下载 person.tar 到 Detection\work\PaddleClas\dataset 路径并解压，该数据是本案例的训练和验证数据，为了加快训练，该数据集是 person 全量数据集的子集，全部集合的获取可以参考 PaddleClas 有人/无人案例。由于不是全量数据，所以最终的训练精度与提供的精度不会一致。

（2）执行上述命令后，dataset/ 下存在 person_exists 目录，该目录中具有以下数据：

```
├── train
│   ├── 000000000009.jpg
│   ├── 000000000025.jpg
...
├── val
│   ├── objects365_01780637.jpg
│   ├── objects365_01780640.jpg
...
├── ImageNet_val
│   ├── ILSVRC2012_val_00000001.JPEG
│   ├── ILSVRC2012_val_00000002.JPEG
...
├── train_list.txt
├── train_list.txt.debug
├── train_list_for_distill.txt
├── val_list.txt
└── val_list.txt.debug
```

其中 train/ 和 val/ 分别为训练集和验证集。train_list.txt 和 val_list.txt 分别为训练集和验证集的标签文件，train_list.txt.debug 和 val_list.txt.debug 分别为训练集和验证集的 debug 标签文件，其分别是 train_list.txt 和 val_list.txt 的子集，用该文件可以快速体验本案例的流程。ImageNet_val/ 是 ImageNet-1k 的验证集，该集合和 train 集合的混合数据用于本案例的 SKL-UGI 知识蒸馏策略，对应的训练标签文件为 train_list_for_distill.txt 。

备注：

①关于 train_list.txt、val_list.txt 的格式说明，可以参考 PaddleClas 分类数据集格式说明 。

②关于如何得到蒸馏的标签文件可以参考知识蒸馏标签获得方法。

3. 模型训练

到 PaddleClas 目录下训练：cd ~/work/PaddleClas/

3.1 基于 SwinTransformer_tiny 训练

在./ppcls/configs/PULC/person_exists/SwinTransformer_tiny_patch4_window7_224.yaml 中提供了基于 SwinTransformer_tiny 训练该场景的配置，训练脚本如下：

```
python tools/train.py
-c ./ppcls/configs/PULC/person_exists/SwinTransformer_tiny_patch4_window7_224.yaml
-o Global.start_eval_epoch=1 -o Optimizer.lr.learning_rate=2.5e-5
-o DataLoader.Train.loader.num_workers=1 -o DataLoader.Eval.loader.num_workers=1
```

3.2 基于 MobileNetV3_small_x0_35 训练

在 ./ppcls/configs/PULC/person_exists/MobileNetV3_small_x0_35.yaml 中提供了基于 MobileNetV3_small_x0_35 训练该场景的配置，训练脚本如下：

```
python tools/train.py
-c ./ppcls/configs/PULC/person_exists/MobileNetV3_small_x0_35.yaml
-o Global.start_eval_epoch=1 -o Optimizer.lr.learning_rate=0.0325
-o DataLoader.Train.loader.num_workers=1 -o DataLoader.Eval.loader.num_workers=1
```

3.3 使用 PP-LCNet_x1_0 模型训练

在./ppcls/configs/PULC/person_exists/PPLCNet_x1_0.yaml 中提供了基于 PP-LCNet_x1_0 训练该场景的配置，训练脚本如下：

```
python tools/train.py
-c ./ppcls/configs/PULC/person_exists/PPLCNet_x1_0.yaml
-o Arch.use_ssld=False -o Global.start_eval_epoch=1 -o Optimizer.lr.learning_rate=0.0025
-o DataLoader.Train.loader.num_workers=1 -o DataLoader.Eval.loader.num_workers=1
```

3.4 基于 SSLD 权重使用 PP-LCNet_x1_0 模型训练

在./ppcls/configs/PULC/person_exists/PPLCNet_x1_0.yaml 中提供了基于 PP-LCNet_x1_0 训练该场景的配置，使用-o Arch.use_ssld=True 表示加载 SSLD 的预训练权重，训练脚本如下：

```
python tools/train.py
-c ./ppcls/configs/PULC/person_exists/PPLCNet_x1_0.yaml
-o Arch.use_ssld=True -o Global.start_eval_epoch=1 -o Optimizer.lr.learning_rate=0.0025
-o DataLoader.Train.loader.num_workers=1 -o DataLoader.Eval.loader.num_workers=1
```

3.5 基于 SKL-UGI Distillation 知识蒸馏得到最终的模型

为了尽可能缩小 PP-LCNetx1_0 和大模型的差距，此处可以通过 SKL-UGI Distillation 来提升模型在该场景的拟合能力，增强模型面对复杂成景的鲁棒性。此过程分为两步，首先训练教师模型，目的是为了得到在该场景中更好的教师模型权重，其次，基于有标签数据和无标签数据，加载教师模型训练得到的权重进行蒸馏训练。

（1）训练教师模型

首先，训练教师模型。复用 ppcls/configs/PULC/person_exists/PPLCNet_x1_0.yaml 中的超参数，训练教师模型，训练脚本如下：

```
python tools/train.py
-c ./ppcls/configs/PULC/person_exists/PPLCNet_x1_0.yaml
-o Arch.name=ResNet101_vd -o Global.start_eval_epoch=1 -o Optimizer.lr.learning_rate=0.0025
-o DataLoader.Train.loader.num_workers=1 -o DataLoader.Eval.loader.num_workers=1
```

当前教师模型最好的权重保存在 output/ResNet101_vd/best_model.pdparams

（2）训练蒸馏模型

配置文件 ppcls/configs/PULC/person_exists/PPLCNet_x1_0_distillation.yaml 提供了 SKL-UGI 知识蒸馏策略的配置。该配置将 ResNet101_vd 当作教师模型，PPLCNet_x1_0 当作学生模型，此处使用 ImageNet 数据集的验证集的子集作为新增的无标签数据。训练脚本如下：

```
python tools/train.py
-c ./ppcls/configs/PULC/person_exists/PPLCNet_x1_0_distillation.yaml
-o Arch.models.0.Teacher.pretrained=output/ResNet101_vd/best_model
-o Global.start_eval_epoch=1 -o Optimizer.lr.learning_rate=0.0025
-o DataLoader.Train.loader.num_workers=1 -o DataLoader.Eval.loader.num_workers=1
```

当前模型最好的权重保存在 output/DistillationModel/best_model_student.pdparams

三、模型导出

1. 通过导出 inference 模型，PaddlePaddle 支持使用预测引擎进行预测推理。接下来介绍如何用预测引擎进行推理：首先，对训练好的蒸馏模型进行转换：

① cd ~/work/PaddleClas/

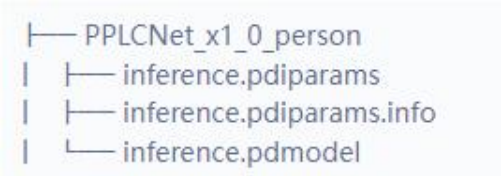
② python tools/export_model.py

-c ppcls/configs/PULC/person_exists/PPLCNet_x1_0.yaml

-o Global.pretrained_model=output/DistillationModel/best_model_student

-o Global.save_inference_dir=deploy/models/PPLCNet_x1_0_person

2. 转换完成后在 deploy/models/下生成 PPLCNet_x1_0_person 文件夹,该文件夹有如下结构：



其中，inference.pdiparams 和 inference.pdmodel 分别对应了模型的参数文件和模型结构文件。

四、推理预测

1. 用下载的模型 person_exists_infer 进行预测

第三节导出的模型可直接用于推理部署，但是由于该模型不是经过全量数据训练得到的模型，模型的精度有限，此时可以下载已经训练好的模型直接预测。进入 deploy 目录下：cd ./deploy，然后下载 person_exists_infer 模型到 deploy/models，解压后返回 deploy 目录。

1.1 预测单张图像

运行下面的命令，对图像 ./images/PULC/person_exists/objects365_02035329.jpg 进行有人/无人分类：

① 使用 GPU 进行预测

python python/predict_cls.py -c configs/PULC/person_exists/inference_person_exists.yaml

② 使用 CPU 进行预测

python python/predict_cls.py -c configs/PULC/person_exists/inference_person_exists.yaml -o Global.use_gpu=False

输出：objects365_02035329.jpg: class id(s): [1], score(s): [1.00], label_name(s): ['someone']

其中，someone 表示该图里存在人，nobody 表示该图里不存在人。

1.2 基于文件夹的批量预测

如果希望预测文件夹内的图像，可以直接修改配置文件中的 Global.infer_imgs 字段，也可以通过下面的 -o 参数修改对应的配置：

① 使用 GPU 进行预测

python python/predict_cls.py -c configs/PULC/person_exists/inference_person_exists.yaml -o Global.infer_imgs="./images/PULC/person_exists/"

② 使用 CPU 进行预测

python python/predict_cls.py -c configs/PULC/person_exists/inference_person_exists.yaml -o Global.infer_imgs="./images/PULC/person_exists/" -o Global.use_gpu=False

终端中会输出该文件夹内所有图像的分类结果，如下所示：

objects365_01780782.jpg: class id(s): [0], score(s): [1.00], label_name(s): ['nobody']

objects365_02035329.jpg: class id(s): [1], score(s): [1.00], label_name(s): ['someone']

在这里，您也可以尝试上传自己的数据进行预测。

2. 用导出模型 PPLCNet_x1_0_person 进行预测

如果希望用第三节导出的模型 PPLCNet_x1_0_person 进行预测，可以修改配置文件 configs/PULC/person_exists/inference_person_exists.yaml 中的 Global.inference_model_dir 字段为"./models/PPLCNet_x1_0_person"，再用同样的方法进行预测。

项目阶段 II：GUI 部署

一、方案介绍

1. 模型准备

（1）PPLCNet_x1_0_person 模型（项目阶段 I 得到）

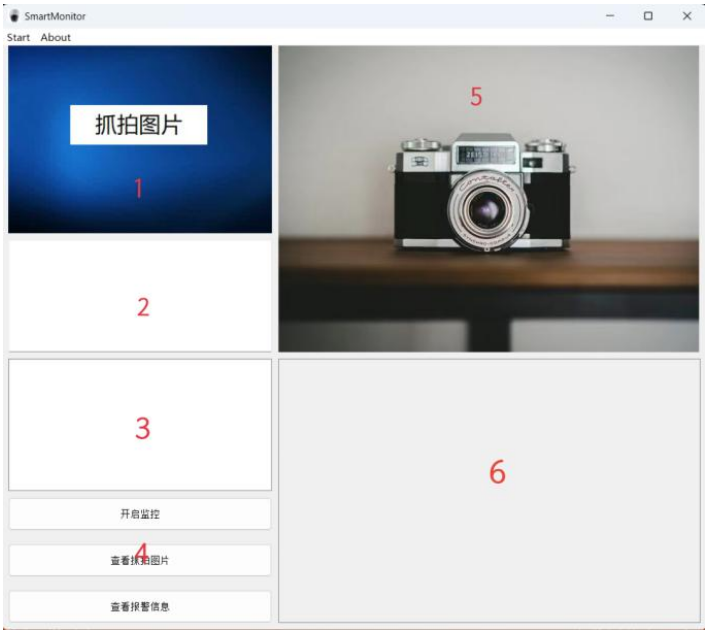
（2）PPLCNet_x1_0_person_attribute 模型

PPLCNet_x1_0_person_attribute 模型用于识别图片中人物的属性，例如性别、年龄范围、有无帽子、有无背包等。关于如何得到 PPLCNet_x1_0_person_attribute 模型，可以访问 GitHub 项目地址：PULC 人体属性识别模型。

2. GUI 开发

2.1 开发目标

制作如图 GUI 界面，实现人机交互：



1：抓拍图片显示区

2：抓拍图片中的人物属性显示区

3：抓拍图片选择列表

4：按钮区

5：实时监控显示区

6：实时报警信息提示区

关于 GUI 界面使用，参照第三节 GUI 示例或产品使用手册。

2.2 开发工具

Python3.8 + PyQt5 + Qt Designer5.11.1

2.3 技术路线

（1）基于 Python 编写功能模块

（2）基于 Qt Designer 设计 Qt 窗体文件

（3）基于 PyQt 把 Qt 窗体文件链接到对应的功能模块中

3. GUI 测试

3.1 Debug

（1）摄像头视频流读取问题：如果网络摄像头不可用，建议换用 USB 摄像头。

（2）监控画面卡顿问题：多线程方式，实现监控与检测报警并发进行。

（3）其它问题：根据 TraceBack 报错，利用断点调试逐步排查。

3.2 可执行文件(.exe)

（1）Python 项目打包

pip install pyinstaller //安装 pyinstaller 包

pyinstaller -D -w main.py //main.py 为项目中的主函数，执行后生成 main.spec 文件

pyinstaller -y main.spec //执行前可以修改 main.spec 中的参数

（2）运行 main.exe 文件（打包生成的 main 文件夹里）

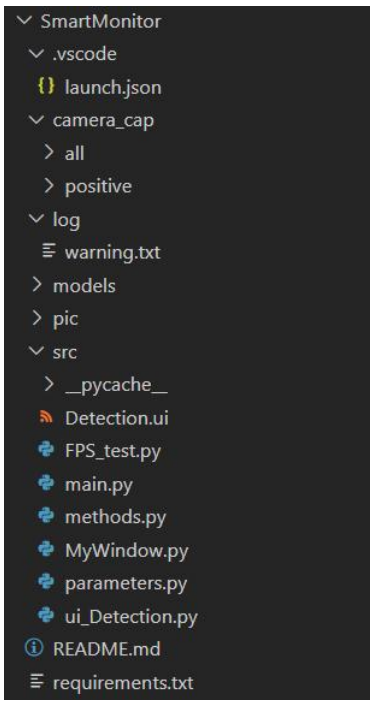
如果运行报错，可能是因为安装的依赖包缺失或不正确，建议安装后重新打包测试。

二、实践教程

1. 环境配置

1.1 新建项目文件夹“SmartMonitor”

按如下结构配置文件夹目录：



1. “.vscode”：python 配置文件夹。
2. “camera_cap”：包含“all”和“positive”两个文件夹（“all”文件夹保存所有监控截图；“positive”文件夹保存有异常人员的监控截图）。
3. “log”：包含“warning.txt”文件，记录报警信息。
4. “models”：保存了项目用到的模型参数文件。
5. “pic”：保存了项目用到的资源文件（图片或视频）。
6. “src”：保存.py 代码文件以及.ui 文件。

有关各文件详细信息，请参阅 README.md。

1.2 环境要求

（1）Python3.8.0

（2）PyQt5

（3）Qt Designer5.11.1

（4）IDE：Visual Studio Code 1.76.1 / PyCharm 2022.2.4

2. 准备资源文件

2.1 模型准备

把阶段Ⅰ得到的 PPLCNet_x1_0_person 文件夹和 PPLCNet_x1_0_person_attribute 文件夹复制到“SmartMonitor/models”文件夹里。

2.2 图片准备

下载几张分辨率合适的图片到“SmartMonitor/pic”文件夹里，用于 GUI 的初始化显示。

3. 代码开发

3.1 代码文件概览

（1）main.py：主函数出口，调用 MyWindow 类测试。

（2）MyWindow.py：新建一个 MyWindow 类继承 QMainWindow，实现自定义窗体的功能。

（3）methods.py：MyWindow 类中所需要的函数或方法。

（4）parameters.py：保存常量参数，如摄像头路径、模型路径、图片路径等。

（5）FPS_test.py：独立的.py 文件，仅用于测试摄像头帧率。

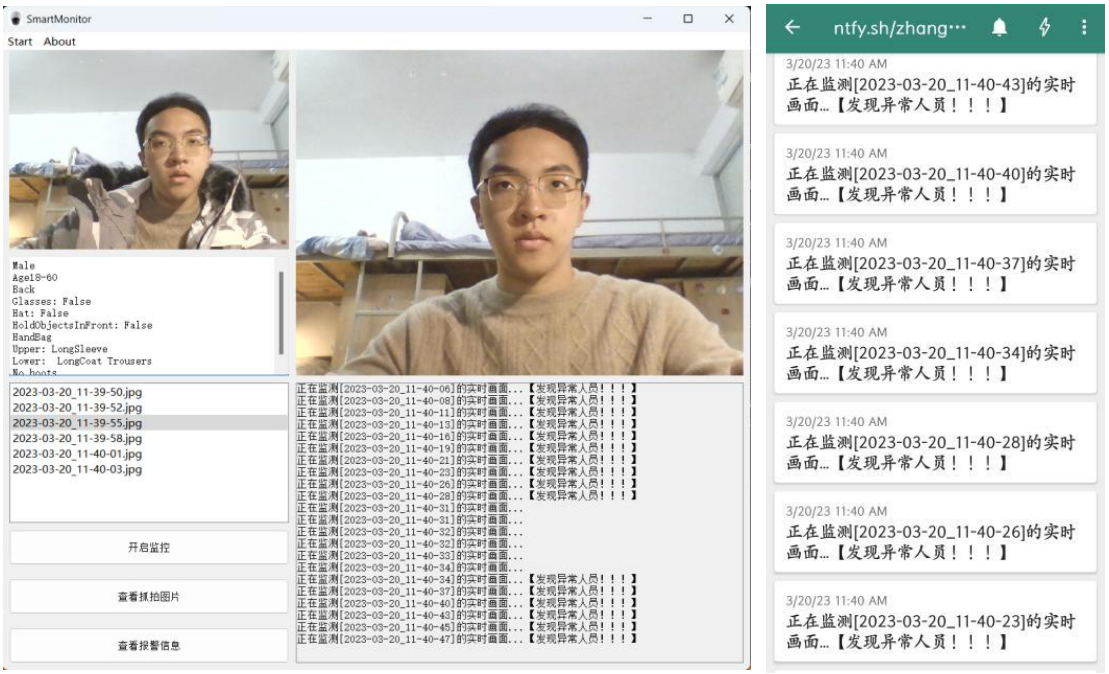
（6）Detection.ui：使用 Qt Designer 自定义设计的窗体文件。

（7）ui_Detection.py：由 Detection.ui 自动转换得到（VS Code 里已下载 Qt for Python 插件）。

关于软件项目详细介绍，请参阅 README.Md

3.2 移动端接收报警短信

由于本产品支持实时接收报警短信提醒，因此用户需要首先在移动端（手机）上下载 ntfy app，然后打开 ntfy app，订阅自己的 topic，下载和使用说明可参阅 ntfy 官网。效果如下：

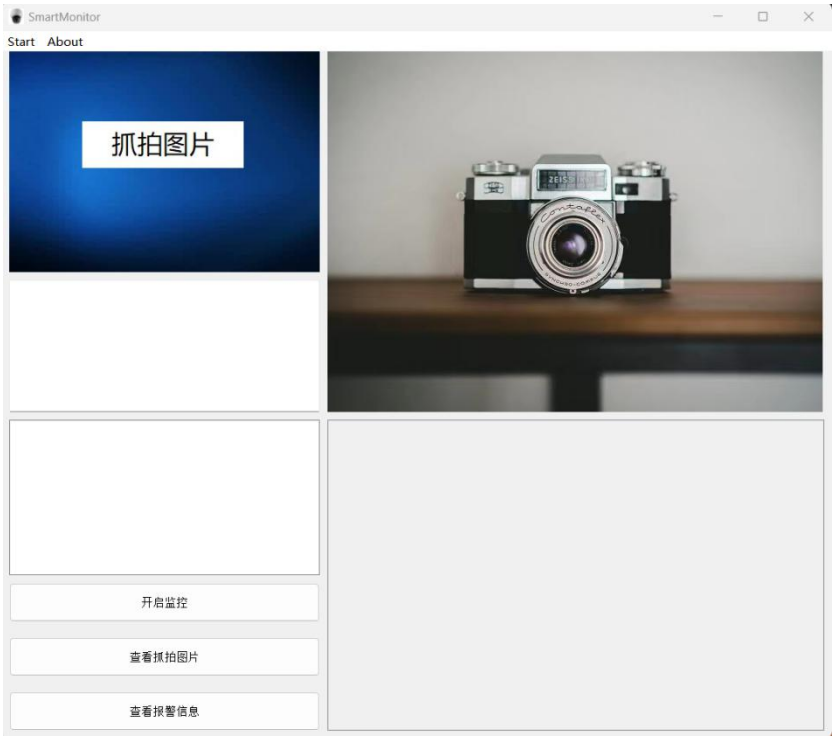


3.3 Debug

运行 main.py 测试，巧用断点调试+过程变量分析。

三、GUI 示例

1. GUI 初始化示例图



2. GUI 使用

请参阅产品使用手册

3. GUI 演示视频

SmartMonitor_demo

4. 项目更多信息

欢迎访问 GitHub 项目地址：SmartMonitor_GitHub