9.1 必要的环境依赖和项目克隆

9.1.1 YOLOv5框架搭建

YOLOv5的代码是开源的，因此我们可以github上克隆其源码。访问开源网站https://github.com/ultralytics/yolov5并下载其yolov5框架。以下为其代码的整体目录介绍。

data：用于存放超参数的配置文件，用来配置训练集和测试集还有验证集的路径。其中包括目标检测的种类数和种类的名称。

models：主要存放网络构建的配置文件和函数，其中包含了该项目的四个不同的版本，分别为是s、m、l、x。这几个版本的检测测度分别都是从快到慢，但是精确度分别是从低到高。

utils：用于存放工具类的函数，里面有loss函数，metrics函数，plots函数等。

weights：用于存放训练好的权重参数。

detect.py：利用训练好的权重参数进行目标检测，可以进行图像、视频和摄像头的检测。

train.py：用于训练自己的数据集的函数。

test.py：测试训练的结果的函数。

requirements.txt：环境安装文本文件，记录使用yolov5项目的环境依赖包的一些版本，可以利用该文本导入相应版本的包。

9.1.2 环境配置

为该项目创建一个虚拟环境后，打开requirements.txt 可以看到里面有很多环境依赖包的版本，在终端运行pip install -r requirements.txt命令即可自动完成环境所需的依赖包的安装。

9.2 数据与预训练权重的准备

9.2.1数据集的准备

数据集采用 https://toscode.gitee.com/hanqikai/Real-World-Masked-Face-Dataset 提供的图片，从中选取大约2000张图片，并利用labelimg来制作自己的数据集。接下来将数据集划分为训练集和验证集。由于我们会发现部分识别系统对于使用手遮挡嘴部会导致误判情况，对数据集我们增加了部分类似数据进行训练，以降低误判的情况。

9.2.2 获得预训练权重

使用预训练权重，可以缩短网络的训练时间，达到更好的精度。而yolov5的5.0版本给我们提供了几个预训练权重，我们可以对应我们不同的需求选择不同的版本的预训练权重。通过如下的图可以获得权重的名字和大小信息，可以预料的到，预训练权重越大，训练出来的精度就会相对来说越高，但是其检测的速度就会越慢。本次训练自己的口罩数据集所采用的预训练权重为yolov5s.pt。

9.3 模型训练

9.3.1 文件配置

在开始训练需要对项目里的相关文件进行修改，一个是数据配置文件，另一个是模型配置文件。在data目录下新建一个imaskpro.yaml文件用来配置之前准备好的数据集作为数据配置文件，文件格式可以参考项目中的VOC.yaml，在其基础上进行修改。在model目录下同样新建一个imaskpro.yaml文件用来作为模型配置文件，我这里在yolov5s.yaml的模型文件的基础上进行修改。

9.3.2 模型参数配置和训练

打开train.py文件，将“weights”、“cfg”、“data”的参数设置为我们之前创建好的文件路径。下面介绍train.py中部分参数的含义。

--weights：初始化的权重文件的路径地址

--cfg：模型yaml文件的路径地址

--data：数据yaml文件的路径地址

--hyp：超参数文件路径地址

--epochs：训练轮次

--batch-size：批次文件的多少

--img-size：输入图片尺寸

--rect:是否采用矩形训练，默认False

--resume:接着打断训练上次的结果接着训练

--cache-images:是否提前缓存图片到内存，以加快训练速度，默认False

--image-weights：使用加权图像选择进行训练

--device:训练的设备，cpu；0(表示一个gpu设备cuda:0)；0,1,2,3(多个gpu设备)

--multi-scale:是否进行多尺度训练，默认False

--workers：最大工作核心数

配置好参数后就可以运行train.py进行模型的训练了，模型训练时间较长，需要耐心等待。训练完成后，在run/train/exp目录下就可以查看训练的结果了。模型训练结果如下图9-1所示。

|  |
| --- |
| 9-1 模型训练结果 |

9.4 推理测试

训练结束后，会产生一个runs的文件夹，在runs/train/exp/weights会产生两个权重文件，其中best.pt(最好的权重)，last.pt(是最后一轮的权重)，我们推理利用最好的权重(best.pt)。

打开detect.py文件同样存在许多参数设置，下面将介绍部分参数的含义。

--weights:权重的路径地址

--source:测试数据，可以是图片/视频路径，也可以是'0'(电脑自带摄像头),也可以是rtsp等视频流

--output:网络预测之后的图片/视频的保存路径

--img-size:网络输入图片大小

--conf-thres:置信度阈值

--iou-thres:做nms的iou阈值

--device: 推理的设备，cpu；0(表示一个gpu设备cuda:0)；0,1,2,3(多个gpu设备)

--save-txt:是否将预测的框坐标以txt文件形式保存，默认False

--agnostic-nms:进行nms是否也去除不同类别之间的框，默认False

--augment:推理的时候进行多尺度，翻转等操作(TTA)推理

修改weights参数为我们之前训练好的模型路径，修改source参数为待推理的路径后运行detect.py即可在runs/detect/exp路径下找到推理后的结果。测试结果如下图所示9-2所示。

|  |
| --- |
| 9-2 推理测试结果 |

9.5 UI界面的制作与优化。

由于推理测试每次都需要修改源码中的路径再重复运行较为繁琐。我编写并改进了一个UI界面，使得用户可以方便的自由选取需要用来推理预测的模型，以及自由的调整IoU、置信度等参数，并且支持从文件夹选择图片和视频。点击运行后可以直接将图片和视频展示给用户，并且能够实时二点反馈当前预测的类别数量，增强了用户体验。