口罩佩戴检测实验报告

姓名：禹相祐　　学号：2312900

一. 问题重述

1.1 实验背景

今年一场席卷全球的新型冠状病毒给人们带来了沉重的生命财产的损失。而有效防御这种传染病毒的方法就是积极佩戴口罩。我国对此也采取了严肃的措施，在公共场合要求人们必须佩戴口罩。在本次实验中，我们要建立一个目标检测的模型，可以识别图中的人是否佩戴了口罩。

1.2 实验要求

1）建立深度学习模型，检测出图中的人是否佩戴了口罩，并将其尽可能调整到最佳状态。

2）学习经典的模型 MTCNN 和 MobileNet 的结构。

3）学习训练时的方法。

1.3 实验环境

可以使用基于 Python 的 OpenCV 、PIL 库进行图像相关处理，使用 Numpy 库进行相关数值运算，使用 Pytorch 等深度学习框架训练模型等。

二．设计思想

对于此次实验，我们采用的是 MobileNet 模型进行针对是否佩戴口罩的检测的目标检测的任务，进而实现对是否佩戴口罩的精确判断。

鉴于 mo 上已经有了一个比较详尽的代码框架，我们所需要做的就只剩下对几个关键的参数进行调整，使得学习时将 loss 尽量降低并使得最后提交时得到的分数尽量高即可。

下面对几个关键参数进行说明：

1. Epochs： 模型训练轮数，每次训练完后会输出当前轮次对应的平均 loss 值；

2. Lr: 学习率，此处设置为 1e-3 为一个比较常见的数值，lr 越大，收敛速度越快，但可能跳过潜在的最优解；lr 越小，收敛速度越慢，但同样也有可能陷入局部最优从而忽略全局最优；

3. Factor: 衰减因子。此处设为 0.5 意味着当准确率不再提升时候，会将目前的学习率*0.5 赋值为新的学习率再进行一轮学习；

4. Patience: 忍耐的周期。此处指的是若连续 patience 个 epochs 准确率没有进一步提升，就将学习率*factor 进行新一轮次的学习。

三．代码

修改后的参数如下：

```
epochs = 40
model = MobileNetV1(classes=2).to(device)
optimizer = optim.Adam(model.parameters(), lr=1e-3)  # 优化器
# 学习率下降的方式，acc 三次不下降就下降学习率继续训练，衰减学习率
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer,
                                                 'max',
                                                 factor=0.22,
                                                 patience=12)
# 0.22 12 40 95points
# 损失函数
criterion = nn.CrossEntropyLoss()
```

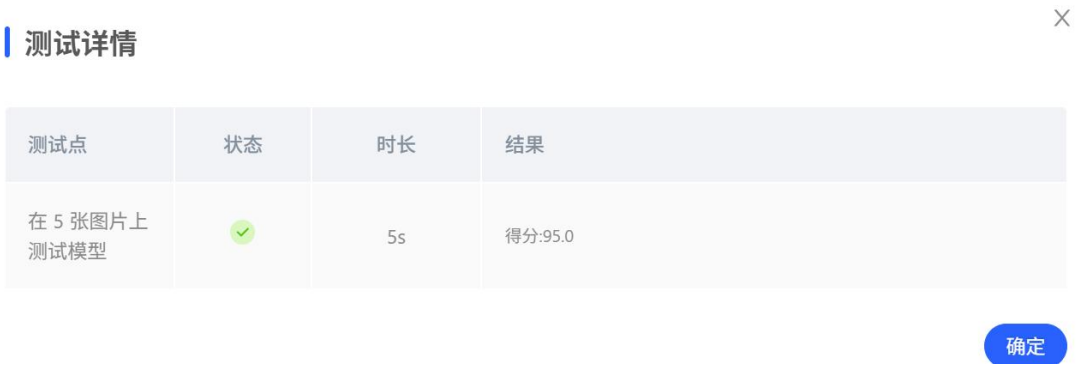
main.py 完整代码如下：

```
import warnings
# 忽视警告
warnings.filterwarnings('ignore')

import cv2
from PIL import Image
import numpy as np
import copy
import matplotlib.pyplot as plt
from tqdm.auto import tqdm
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision.datasets import ImageFolder
import torchvision.transforms as T
from torch.utils.data import DataLoader


from torch_py.Utils import plot_image
from torch_py.MTCNN.detector import FaceDetector
from torch_py.MobileNetV1 import MobileNetV1
from torch_py.FaceRec import Recognition


from torch_py.Utils import plot_image
```

```python
from torch_py.MTCNN.detector import FaceDetector
from torch_py.MobileNetV1 import MobileNetV1
from torch_py.FaceRec import Recognition
from torch_py.FaceRec import Recognition
from PIL import Image
import cv2

# -------------------------- 请加载您最满意的模型 --------------------------
# 加载模型(请加载你认为的最佳模型)
# 加载模型,加载请注意 model_path 是相对路径, 与当前文件同级。
# 如果你的模型是在 results 文件夹下的 dnn.h5 模型，则 model_path = 'results/temp.pth'
# 此处将最佳模型存在了 results 文件夹下的 my_own_temp.pth
model_path = 'results/my_own_temp.pth'
# --------------------------------------------------------------------------

def predict(img):
    """
    加载模型和模型预测
    :param img: cv2.imread 图像
    :return: 预测的图片中的总人数、其中佩戴口罩的人数
    """
    # -------------------------- 实现模型预测部分的代码 --------------------------
    # 将 cv2.imread 图像转化为 PIL.Image 图像，用来兼容测试输入的 cv2 读取的图像（勿删！！！）
    # cv2.imread 读取图像的类型是 numpy.ndarray
    # PIL.Image.open 读取图像的类型是 PIL.JpegImagePlugin.JpegImageFile
    if isinstance(img, np.ndarray):
        # 转化为 PIL.JpegImagePlugin.JpegImageFile 类型
        img = Image.fromarray(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))

    recognize = Recognition(model_path)
    img, all_num, mask_num = recognize.mask_recognize(img)
    # --------------------------------------------------------------------------
    return all_num,mask_num
```

四．实验结果

最后运行结果如图：



五．总结

此次实验还算是比较简单的，因为这次实验需要的也就是手动调整下超参数 epoch、lr、patience 等。这次的实验也让我想起了上学期我在 python 课上为了实现新闻分类模型，一直在不断手动地调整参数，最后还是调用了 optuna 库实现的精准调参。