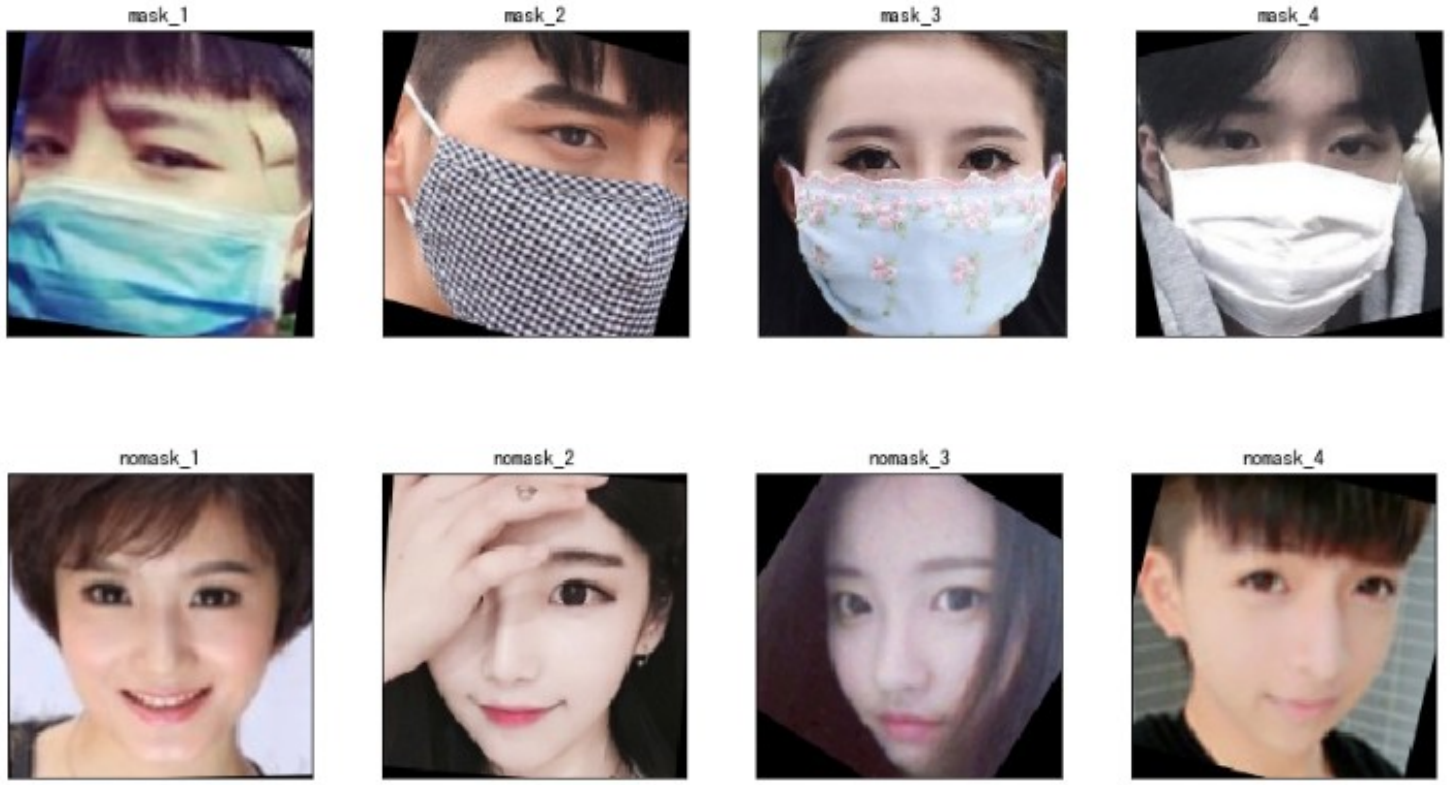
# 口罩佩戴检测

Mask Wearing
Recognition

2024 年 5 月 14 日 - 2024 年 5 月 20 日

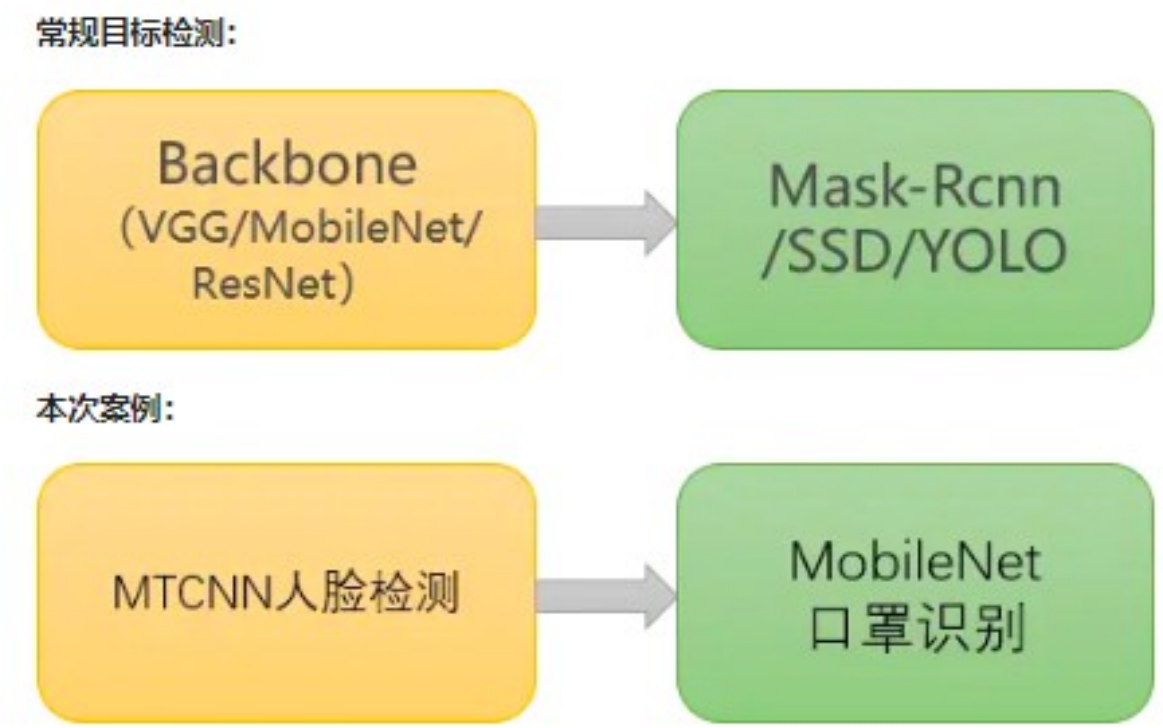
# 实验介绍

The steps of experiment



# 实验思路

The steps of experiment



# 实验思路

The steps of experiment


预处理及数据增强

创建数据集

利用 MTCNN 进行人脸识别

利用 MobileNet 进行口罩识别

训练得到最终模型

# 数据预处理

The steps of experiment

```
train_data = ImageDataGenerator(
        # 对图片的每个像素值均乘上这个放缩因子，把像素值放缩到0和1之间有利于模型的收敛
        rescale=1. / 255,
        # 浮点数，剪切强度（逆时针方向的剪切变换角度）
        shear_range=0.1,
        # 随机缩放的幅度，若为浮点数，则相当于[lower,upper] = [1 - zoom_range, 1+zoom_range]
        zoom_range=0.1,
        # 浮点数，图片宽度的某个比例，数据提升时图片水平偏移的幅度
        width_shift_range=0.1,
        # 浮点数，图片高度的某个比例，数据提升时图片竖直偏移的幅度
        height_shift_range=0.1,
        # 布尔值，进行随机水平翻转
        horizontal_flip=True,
        # 布尔值，进行随机竖直翻转
        vertical_flip=True,
        # 在 0 和 1 之间浮动。用作验证集的训练数据的比例
        validation_split=test_split
)
```

# MTCNN： 人脸检测

The steps of experiment

```
torch.set_num_threads(1)
# 读取测试图片
img = Image.open("test.jpg")
# 加载模型进行识别口罩并绘制方框
recognize = Recognition()
draw = recognize.face_recognize(img)
plot_image(draw)
```

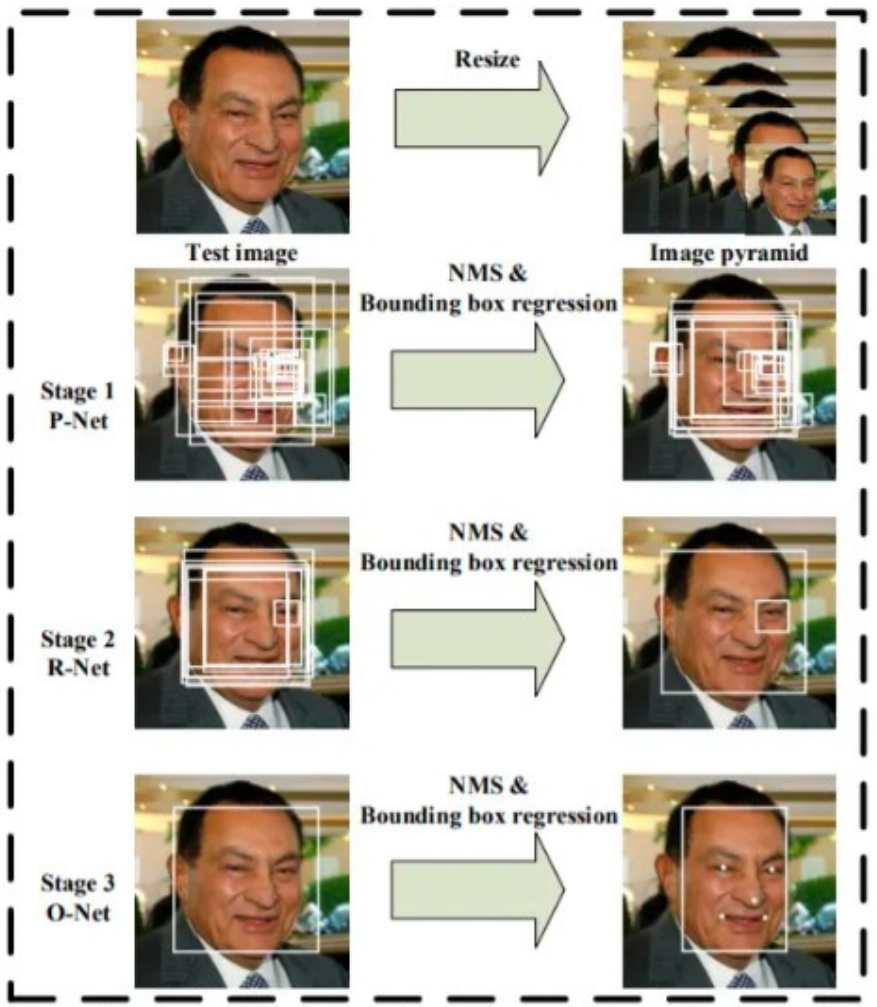



Fig. 1. Pipeline of our cascaded framework that includes three-stage multi-task deep convolutional networks. Firstly, candidate windows are produced through a fast Proposal Network (P-Net). After that, we refine these candidates in the next stage through a Refinement Network (R-Net). In the third stage, The Output Network (O-Net) produces final bounding box and facial landmarks position.

# 《Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks》

# MobileNet： 口罩识别

The steps of experiment

## 4.1 预训练模型 MobileNet

```
# 加载 MobileNet 的预训练模型权重
weights_path = basic_path + 'keras_model_data/mobilenet_1_0_224_tf_no_top.h5'
# 图像数据的行数和列数
height, width = 160, 160
model = MobileNet(input_shape=[height,width,3],classes=2)
model.load_weights(weights_path,by_name=True)
print('加载完成...')
```

# 实验思路

The steps of experiment

```
# 一次的训练集大小
batch_size = 8
# 图片数据路径
data_path = basic_path + 'image'
# 图片处理
train_generator,test_generator = processing_data(data_path, height=160, width=160, batch_size=batch_size, test_split=0.1)
# 编译模型
model.compile(loss='binary_crossentropy',  # 二分类损失函数
              optimizer=Adam(lr=0.1),              # 优化器
              metrics=['accuracy'])        # 优化目标
# 训练模型
history = model.fit(train_generator,
                    epochs=3, # epochs: 整数，数据的迭代总轮数。
                    # 一个epoch包含的步数,通常应该等于你的数据集的样本数量除以批量大小。
                    steps_per_epoch=637 // batch_size,
                    validation_data=test_generator,
                    validation_steps=70 // batch_size,
                    initial_epoch=0, # 整数。开始训练的轮次（有助于恢复之前的训练）。
                    callbacks=[checkpoint_period, reduce_lr])
# 保存模型
model.save_weights(model_dir + 'temp.h5')
```

```
early_stopping = EarlyStopping(
                                monitor='val_loss',  # 检测的指标
                                min_delta=0,         # 增大或减小的阈值
                                patience=10,         # 检测的轮数频率
                                verbose=1            # 信息展示的模式
                               )
```

```
# 学习率下降的方式，acc三次不下降就下降学习率继续训练
reduce_lr = ReduceLROnPlateau(
                              monitor='acc',  # 检测的指标
                              factor=0.5,     # 当acc不下降时将学习率下调的比例
                              patience=2,     # 检测轮数是每隔两轮
                              verbose=2       # 信息展示模式
                             )
```

# 评分标准
Notifications

人数误差 / 口罩误差： (| 预测值 - 实际值 |/ 实际值 )*10

每张图片的分数： 20- 人数误差 - 口罩误差

# 注意事项
Notifications

1.使用上述学到的方法，训练自己的口罩识别模型，尽可能提高准确度。将训练好的模型保存在 results 文件夹下。
2.点击左侧栏 提交结果 后点击【生成文件】则需要勾选与预测 predict() 函数的 cell相关的其它cell ，并将其转化成为 main.py 文件。
3.请导入必要的包和第三方库以及该模型所依赖的 py 文件 (包括此文件中曾经导入过的)。
4.请加载你认为训练最佳的模型，即请按要求填写模型路径。
5.predict() 函数的输入输出及函数名称请不要改动。

# 注意事项
Notifications

In [2]:
```
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
```

In [3]:
```
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
```

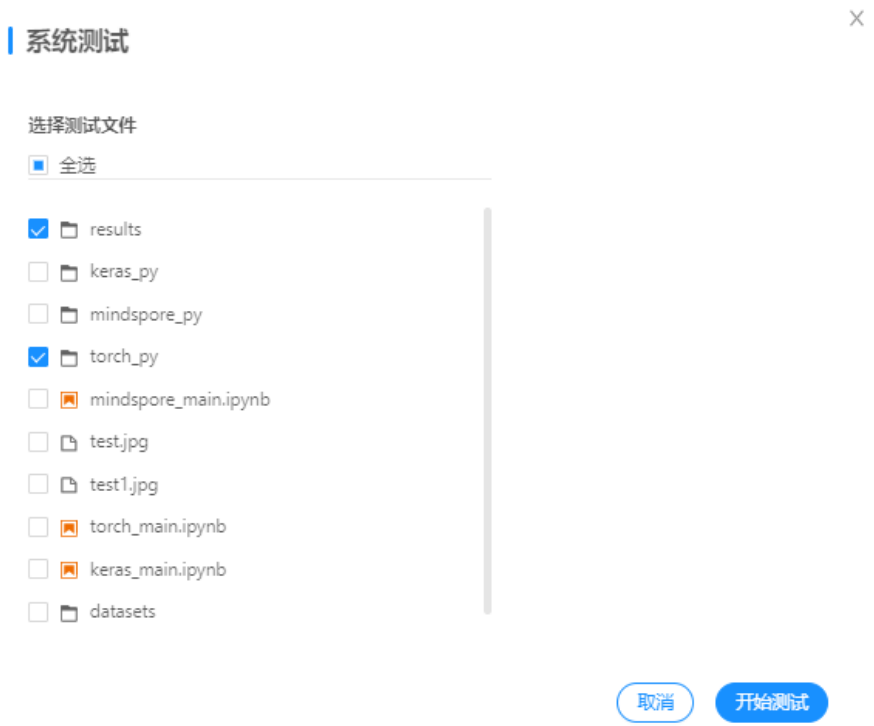
In [3]:
```
from torch_py.Utils import plot_image
from torch_py.MTCNN.detector import FaceDetector
from torch_py.MobileNetV1 import MobileNetV1
from torch_py.FaceRec import Recognition
from torch_py.FaceRec import Recognition
from PIL import Image
import cv2

# -------------------------- 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'
model_path = 'results/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
```

# 注意事项
Notifications




# 谢谢