**程序报告**

学号：2111454 姓名：李潇逸

1. **问题重述**

通过机器学习进行口罩佩戴的识别。

1. **设计思想**

利用机器学习进行模型构建和训练

1. **代码内容**

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

# 数据集路径

data\_path = "./datasets/5f680a696ec9b83bb0037081-momodel/data/"

mask\_num = 4

fig = plt.figure(figsize=(15, 15))

for i in range(mask\_num):

sub\_img = cv2.imread(data\_path + "/image/mask/mask\_" + str(i + 101) + ".jpg")

sub\_img = cv2.cvtColor(sub\_img, cv2.COLOR\_RGB2BGR)

ax = fig.add\_subplot(4, 4, (i + 1))

ax.set\_xticks([])

ax.set\_yticks([])

ax.set\_title("mask\_" + str(i + 1))

ax.imshow(sub\_img)

nomask\_num = 4

fig1 = plt.figure(figsize=(15, 15))

for i in range(nomask\_num):

sub\_img = cv2.imread(data\_path + "/image/nomask/nomask\_" + str(i + 130) + ".jpg")

sub\_img = cv2.cvtColor(sub\_img, cv2.COLOR\_RGB2BGR)

ax = fig1.add\_subplot(4, 4, (i + 1))

ax.set\_xticks([])

ax.set\_yticks([])

ax.set\_title("nomask\_" + str(i + 1))

ax.imshow(sub\_img)

def letterbox\_image(image, size):

"""

调整图片尺寸

:param image: 用于训练的图片

:param size: 需要调整到网络输入的图片尺寸

:return: 返回经过调整的图片

"""

new\_image = cv2.resize(image, size, interpolation=cv2.INTER\_AREA)

return new\_image

# 使用 PIL.Image 读取图片

read\_img = Image.open("test1.jpg")

read\_img = np.array(read\_img)

print("调整前图片的尺寸:", read\_img.shape)

read\_img = letterbox\_image(image=read\_img, size=(50, 50))

read\_img = np.array(read\_img)

print("调整前图片的尺寸:", read\_img.shape)

def processing\_data(data\_path, height=224, width=224, batch\_size=32,

test\_split=0.1):

"""

数据处理部分

:param data\_path: 数据路径

:param height:高度

:param width: 宽度

:param batch\_size: 每次读取图片的数量

:param test\_split: 测试集划分比例

:return:

"""

transforms = T.Compose([

T.Resize((height, width)),

T.RandomHorizontalFlip(0.1), # 进行随机水平翻转

T.RandomVerticalFlip(0.1), # 进行随机竖直翻转

T.ToTensor(), # 转化为张量

T.Normalize([0], [1]), # 归一化

])

dataset = ImageFolder(data\_path, transform=transforms)

# 划分数据集

train\_size = int((1-test\_split)\*len(dataset))

test\_size = len(dataset) - train\_size

train\_dataset, test\_dataset = torch.utils.data.random\_split(dataset, [train\_size, test\_size])

# 创建一个 DataLoader 对象

train\_data\_loader = DataLoader(train\_dataset, batch\_size=batch\_size,shuffle=True)

valid\_data\_loader = DataLoader(test\_dataset, batch\_size=batch\_size,shuffle=True)

return train\_data\_loader, valid\_data\_loader

data\_path = './datasets/5f680a696ec9b83bb0037081-momodel/data/image'

train\_data\_loader, valid\_data\_loader = processing\_data(data\_path=data\_path, height=160, width=160, batch\_size=32)

def show\_tensor\_img(img\_tensor):

img = img\_tensor[0].data.numpy()

img = np.swapaxes(img, 0, 2)

img = np.swapaxes(img, 0, 1)

img = np.array(img)

plot\_image(img)

for index, (x, labels) in enumerate(train\_data\_loader):

print(index, "\nfeature:",x[0], "\nlabels:",labels)

show\_tensor\_img(x)

break

pnet\_path = "./torch\_py/MTCNN/weights/pnet.npy"

rnet\_path = "./torch\_py/MTCNN/weights/rnet.npy"

onet\_path = "./torch\_py/MTCNN/weights/onet.npy"

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)

# 加载 MobileNet 的预训练模型权

device = torch.device("cuda:0") if torch.cuda.is\_available() else torch.device("cpu")

train\_data\_loader, valid\_data\_loader = processing\_data(data\_path=data\_path, height=160, width=160, batch\_size=32)

modify\_x, modify\_y = torch.ones((32, 3, 160, 160)), torch.ones((32))

epochs = 50

model = MobileNetV1(classes=2).to(device)

optimizer = optim.Adam(model.parameters(), lr=0.0002) # 优化器

print('加载完成...')

# 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=0.6,

patience=6)

# 损失函数

criterion = nn.CrossEntropyLoss()

best\_loss = 1e9

best\_model\_weights = copy.deepcopy(model.state\_dict())

loss\_list = [] # 存储损失函数值

for epoch in range(epochs):

model.train()

for batch\_idx, (x, y) in tqdm(enumerate(train\_data\_loader, 1)):

x = x.to(device)

y = y.to(device)

pred\_y = model(x)

# print(pred\_y.shape)

# print(y.shape)

loss = criterion(pred\_y, y)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

if loss < best\_loss:

best\_model\_weights = copy.deepcopy(model.state\_dict())

best\_loss = loss

loss\_list.append(loss)

print('step:' + str(epoch + 1) + '/' + str(epochs) + ' || Total Loss: %.4f' % (loss))

torch.save(model.state\_dict(), './results/temp2.pth')

print('Finish Training.')

plt.plot(loss\_list,label = "loss")

plt.legend()

plt.show()

img = Image.open("test.jpg")

detector = FaceDetector()

recognize = Recognition(model\_path='results/temp2.pth')

draw, all\_num, mask\_nums = recognize.mask\_recognize(img)

plt.imshow(draw)

plt.show()

print("all\_num:", all\_num, "mask\_num", mask\_nums)

# 1.加载数据并进行数据处理

# 2.如果有预训练模型，则加载预训练模型；如果没有则不需要加载

# 3.创建模型和训练模型，训练模型时尽量将模型保存在 results 文件夹

# 4.评估模型，将自己认为最佳模型保存在 result 文件夹，其余模型备份在项目中其它文件夹，方便您加快测试通过。

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 = None

# ---------------------------------------------------------------------------

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

# 输入图片路径和名称

img = cv2.imread("test1.jpg")

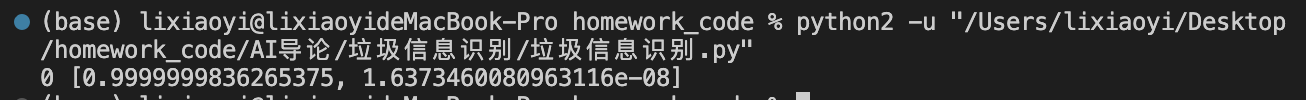
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

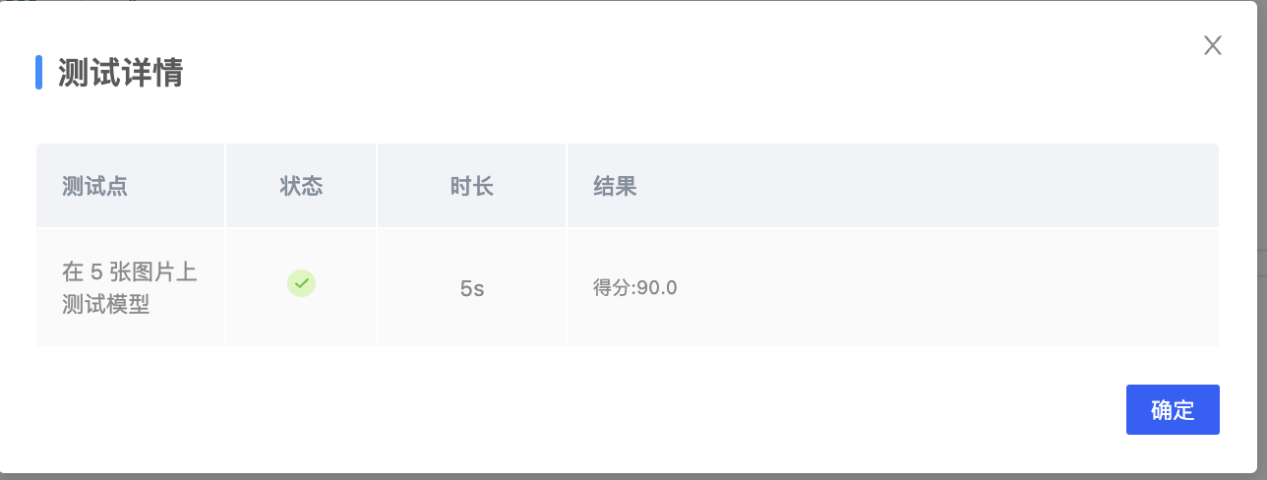
all\_num, mask\_num = predict(img)

# 打印预测该张图片中总人数以及戴口罩的人数

print(all\_num, mask\_num)

1. **实验结果**





1. **总结**

达到了与其设计目的。

改进方向有如下几点：

* 选择更好的模型，使准确率提升。
* 考虑使用深度学习。
* 考虑使用resnet
* 继续调整超参数