**程序报告**

学号：2112060 姓名：孙蕗

1. **问题重述**

（简单描述对问题的理解，从问题中抓住主干，必填）

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 Keras 等深度学习框架训练模型。

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。

将口罩佩戴检测这个目标检测的任务分为两个部分，目标识别和位置检测。

====================================================================

1. **设计思想**

（所采用的方法，有无对方法加以改进，该方法有哪些优化方向（参数调整，框架调整，或者指出方法的局限性和常见问题），伪代码，理论结果验证等… **思考题，非必填**）

====================================================================

1. 学习经典的模型 MTCNN 和 MobileNet 的结构，学习训练时的方法。
2. 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 Keras 等深度学习框架训练模型等。
3. 针对目标检测的任务，可以分为两个部分：目标识别和位置检测。特征提取需要由特有的特征提取神经网络来完成，如 VGG、MobileNet、ResNet 等，这些特征提取网络往往被称为 Backbone 。而在 BackBone 后面接全连接层(FC)就可以执行分类任务。
4. 对图片进行预处理，并制作训练时所需的批量数据集。图片生成器 ImageDataGenerator: keras.preprocessing.image 模块中的图片生成器，主要用以生成一个 batch 的图像数据，支持实时数据提升。训练时该函数会无限生成数据，直到达到规定的 epoch 次数为止。同时也可以在 batch 中对数据进行增强，扩充数据集大小，增强模型的泛化能力，比如进行旋转，变形，归一化等等。
5. 使用现有的表现较好的 MTCNN 的三个权重文件，加载预训练模型MobileNet。
6. 手动调整学习率，调整早停法的参数，避免训练次数过多导致的过拟合问题。
7. **代码内容**

（能体现解题思路的主要代码，有多个文件或模块可用多个"===="隔开，必填）

# 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。

import warnings

# 忽视警告

warnings.filterwarnings('ignore')

import os

import matplotlib

import cv2 as cv

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

from tensorflow.keras.applications.imagenet\_utils import preprocess\_input

from tensorflow.keras import backend as K

from tensorflow.keras.optimizers import Adam

#from tensorflow.keras.utils import np\_utils

#from tensorflow.keras.utils import get\_file

K.image\_data\_format() == 'channels\_last'

from keras\_py.utils import get\_random\_data

from keras\_py.face\_rec import mask\_rec

from keras\_py.face\_rec import face\_rec

from keras\_py.mobileNet import MobileNet

# 数据集路径

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

# mask\_num = 4

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

# for i in range(mask\_num):

# sub\_img = cv.imread(basic\_path + "/image/mask/mask\_" + str(i + 101) + ".jpg")

# sub\_img = cv.cvtColor(sub\_img, cv.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 = cv.imread(basic\_path + "/image/nomask/nomask\_" + str(i + 130) + ".jpg")

# sub\_img = cv.cvtColor(sub\_img, cv.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 = cv.resize(image, size, interpolation=cv.INTER\_AREA)

return new\_image

read\_img = cv.imread("test1.jpg")

print("调整前图片的尺寸:", read\_img.shape)

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

print("调整前图片的尺寸:", read\_img.shape)

# 导入图片生成器

from tensorflow.keras.preprocessing.image import ImageDataGenerator

def processing\_data(data\_path, height, width, batch\_size=16, test\_split=0.1):

"""

数据处理

:param data\_path: 带有子目录的数据集路径

:param height: 图像形状的行数

:param width: 图像形状的列数

:param batch\_size: batch 数据的大小，整数，默认32。

:param test\_split: 在 0 和 1 之间浮动。用作测试集的训练数据的比例，默认0.1。

:return: train\_generator, test\_generator: 处理后的训练集数据、验证集数据

"""

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

)

# 接下来生成测试集，可以参考训练集的写法

test\_data = ImageDataGenerator(

rescale=1. / 255,

validation\_split=test\_split)

train\_generator = train\_data.flow\_from\_directory(

# 提供的路径下面需要有子目录

data\_path,

# 整数元组 (height, width)，默认：(256, 256)。 所有的图像将被调整到的尺寸。

target\_size=(height, width),

# 一批数据的大小

batch\_size=batch\_size,

# "categorical", "binary", "sparse", "input" 或 None 之一。

# 默认："categorical",返回one-hot 编码标签。

class\_mode='categorical',

# 数据子集 ("training" 或 "validation")

subset='training',

seed=0)

test\_generator = test\_data.flow\_from\_directory(

data\_path,

target\_size=(height, width),

batch\_size=batch\_size,

class\_mode='categorical',

subset='validation',

seed=0)

return train\_generator, test\_generator

# 数据路径

data\_path = basic\_path + 'image'

# 图像数据的行数和列数

height, width = 160, 160

# 获取训练数据和验证数据集

train\_generator, test\_generator = processing\_data(data\_path, height, width)

# 通过属性class\_indices可获得文件夹名与类的序号的对应字典。 (类别的顺序将按照字母表顺序映射到标签值)。

labels = train\_generator.class\_indices

print(labels)

# 转换为类的序号与文件夹名对应的字典

labels = dict((v, k) for k, v in labels.items())

print(labels)

pnet\_path = "./datasets/5f680a696ec9b83bb0037081-momodel/data/keras\_model\_data/pnet.h5"

rnet\_path = "./datasets/5f680a696ec9b83bb0037081-momodel/data/keras\_model\_data/rnet.h5"

onet\_path = "./datasets/5f680a696ec9b83bb0037081-momodel/data/keras\_model\_data/onet.h5"

# 读取测试图片

# img = cv.imread("test.jpg")

# # 转换通道

# img = cv.cvtColor(img, cv.COLOR\_RGB2BGR)

# # 加载模型进行识别口罩并绘制方框

# detect = face\_rec(pnet\_path,rnet\_path,onet\_path)

# detect.recognize(img)

# # 展示结果

# fig = plt.figure(figsize = (8,8))

# ax1 = fig.add\_subplot(111)

# ax1.set\_xticks([])

# ax1.set\_yticks([])

# ax1.set\_title('mask\_1')

# ax1.imshow(img)

# 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...')

def save\_model(model, checkpoint\_save\_path, model\_dir):

"""

保存模型，每迭代3次保存一次

:param model: 训练的模型

:param checkpoint\_save\_path: 加载历史模型

:param model\_dir:

:return:

"""

if os.path.exists(checkpoint\_save\_path):

print("模型加载中")

model.load\_weights(checkpoint\_save\_path)

print("模型加载完毕")

checkpoint\_period = ModelCheckpoint(

# 模型存储路径

model\_dir + 'ep{epoch:03d}-loss{loss:.3f}-val\_loss{val\_loss:.3f}.h5',

# 检测的指标

monitor='val\_accuracy',

# ‘auto’，‘min’，‘max’中选择

mode='max',

# 是否只存储模型权重

save\_weights\_only=False,

# 是否只保存最优的模型

save\_best\_only=True,

# 检测的轮数是每隔2轮

period=2

)

return checkpoint\_period

checkpoint\_save\_path = "./results/temp1.h5"

model\_dir = "./results/"

checkpoint\_period = save\_model(model, checkpoint\_save\_path, model\_dir)

# 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='val\_accuracy', # 检测的指标

factor=0.1, # 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

)

early\_stopping = EarlyStopping(

monitor='val\_loss', # 检测的指标

min\_delta=0.00001, # 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

)

# 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=5e-6), # 优化器

metrics=['accuracy']) # 优化目标

# 训练模型

history = model.fit(train\_generator,

epochs=20, # 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=max(1, 641 // batch\_size),

validation\_data=test\_generator,

validation\_steps=max(1, 71 // 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\_new.h5')

plt.plot(history.history['loss'],label = 'train\_loss')

plt.plot(history.history['val\_loss'],'r',label = 'val\_loss')

plt.legend()

plt.show()

plt.plot(history.history['accuracy'],label = 'acc')

plt.plot(history.history['val\_accuracy'],'r',label = 'val\_acc')

plt.legend()

plt.show()

img = cv.imread("./test1.jpg")

img = cv.cvtColor(img, cv.COLOR\_RGB2BGR)

# 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)

# 最佳模型路径

model\_path = "results/temp\_new.h5"

# 加载训练模型并进行口罩识别

detect = mask\_rec(model\_path)

img, all\_num, mask\_num = detect.recognize(img)

# 展示图片口罩识别结果

fig = plt.figure(figsize=(8, 8))

ax1 = fig.add\_subplot(111)

ax1.set\_xticks([])

ax1.set\_yticks([])

ax1.set\_title('test\_mask')

ax1.imshow(img)

plt.show()

print("图中的人数有：" + str(all\_num) + "个")

print("戴口罩的人数有：" + str(mask\_num) + "个")

====================================================================

from keras\_py.utils import get\_random\_data

from keras\_py.face\_rec import mask\_rec

from keras\_py.face\_rec import face\_rec

from keras\_py.mobileNet import MobileNet

from PIL import Image

import cv2

import numpy as np

import matplotlib.pyplot as plt

# -------------------------- 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.h5'

model\_path ='results/temp\_new.h5'

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

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))

detect = mask\_rec(model\_path)

img, all\_num, mask\_num = detect.recognize(img)

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

return all\_num,mask\_num

1. **实验结果**

（实验结果，必填）

====================================================================



1. **总结**

（自评分析（是否达到目标预期，可能改进的方向，实现过程中遇到的困难，从哪些方面可以提升性能，模型的超参数和框架搜索是否合理等），**思考题，非必填**）

====================================================================

1. 一次的训练集大小调整为8
2. 数据的迭代总轮数调整为30
3. 当acc不下降时将学习率下调的比例factor=0.1