**程序报告**

学号：2211290 姓名：姚知言

1. **问题重述**

建立深度学习模型，检测图中的人是否佩戴口罩，并通过学习经典的MTCNN人脸识别和Mobilenet口罩识别，不断进行参数调优使得模型达到更好状态，得到训练模型。

1. **设计思想**

该实验的总体框架为预处理及数据增强-创建数据集-MCTNN人脸识别-MobileNet口罩识别-得到最终模型。

首先对未进行参数调优的测试组进行一次测试，得到结果为keras组77.5分，touch组56.67分，mindspore组75.0分，以便后续对比。

针对keras组进行优化，修改epochs=10/100后，分数都没有变化，为权衡运行时间，选用epoch=20。针对patience=3，batch\_size = 16/32进行尝试后，未能取得更理想的分数。

先对touch组进行优化，修改patience=3，并修改epochs=10，增加学习次数，得分为90.0分。在此之后，我尝试了更多的参数组合，最终取得了较好的模型实现。

1. **代码内容**

#touch组代码

epochs = 9

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

optimizer = optim.Adam(model.parameters(), lr=1e-3) # 优化器

print('加载完成...')

scheduler = optim.lr\_scheduler.ReduceLROnPlateau(optimizer,

'max',

factor=0.5,

patience=3)

# 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)

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/temp.pth')

print('Finish Training.')

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

#torch组提交结果

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

model\_path = 'results/temp.pth'

def predict(img):

if isinstance(img, np.ndarray):

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

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

#kera组

reduce\_lr = ReduceLROnPlateau(

monitor='accuracy', # 检测的指标

factor=0.5, # 当acc不下降时将学习率下调的比例

patience=3, # 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, # 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 = 32

# 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=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=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')

1. **实验结果**



1. **总结**

在本次实验中，通过对MTCNN人脸识别和Mobilenet口罩识别构建，了解了深度学习的基本原理，对深度学习有了基础的认识，也对这些库函数有了一定的理解。随着实验的进行，我对于训练模型的开销有了认识，也认识到一些细微的参数对于模型整体的影响。因此，对模型的细节认知理解是非常重要的。