**程序报告**

学号： 2011937 姓名：姜志凯

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）

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

**利用python等深度学习框架实现口罩佩戴检测功能，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。**

**任务分为两部分：目标识别和位置检测。**

**通常情况下，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。**

**选择已有的人脸检测模型，再训练一个识别口罩的模型，通过调整参数使模型的表现更好。**

**首先，使用基于 Python 的 OpenCV 、PIL 库进行图像相关处理；**

**利用经典模型MTCNN实现人脸检测；MobileNet实现口罩识别模型。**

**本实验可以选择三个框架，分别为：keras 框架、mindspore 框架和torch框架。**

**我采用torch框架实现实验要求功能。**

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. **导入相关的包，得到训练集的正负样本；**
2. **利用CV调整图片的尺寸；**
3. **制作数据集：创建一个 Dataset 对象dataset=torchvision.datasets.ImageFolder(root, transform=None, target\_transform=None, loader=, is\_valid\_file=None)，\_\_int\_\_初始化数据集，\_\_getitem\_\_可以通过索引值对数据集进行访问，\_\_len\_\_返回数据集的大小。创建一个DataLoader对象包装数据，不停循环这个对象；完成数据的加工处理得到想要的数据集；**
4. **利用MTCCN模型进行人脸检测：直接使用表现较好的三个MTCCN权重文件，搭建 MTCNN 网络实现人脸检测：读取测试图片，加载模型进行识别口罩并绘制方块；**
5. **口罩识别：加载预训练模型MobileNet，通过优化器迭代得到更好的模型，这里迭代次数为epochs=10；设置学习率optimizer,’max’,factor=0.75, patience=3;进行模型训练并进行简单的小测试，测试正确性；**
6. **将模型保存在result下，在预测函数中直接加载模型即可。**
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）

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

数据处理：

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

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

人脸检测：

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)

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

口罩识别：

# 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 = 10

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

optimizer = optim.Adam(model.parameters(), lr=1e-3) # 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=0.75,

patience=3)

# 损失函数

criterion = nn.CrossEntropyLoss()

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

模型训练：

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/temp.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/temp.pth')

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

plt.imshow(draw)

plt.show()

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

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

预测函数：

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

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

结果：

# 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

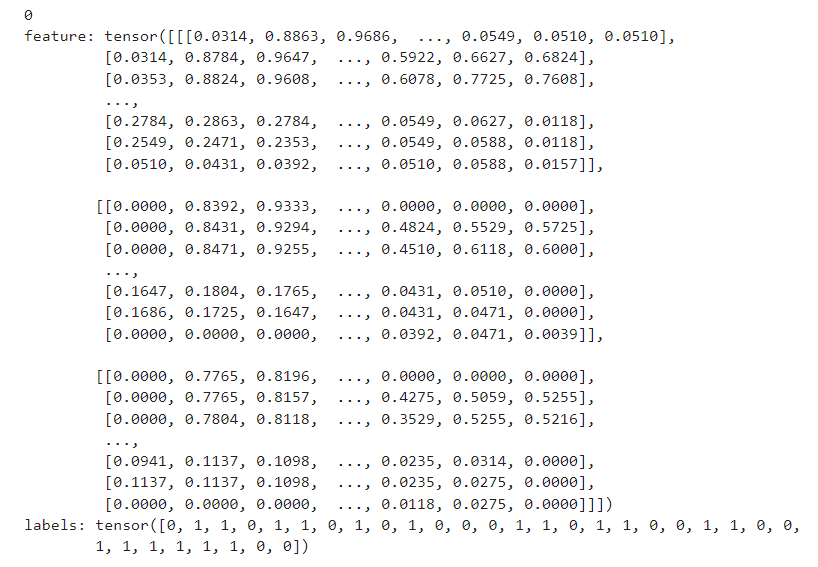
print(all\_num, mask\_num)

1. **实验结果**

（实验结果，必填）

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

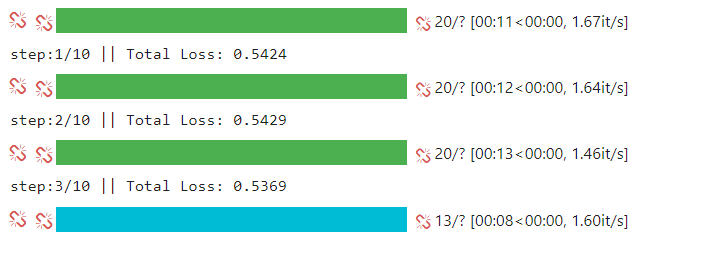
数据处理结果：



识别人脸：

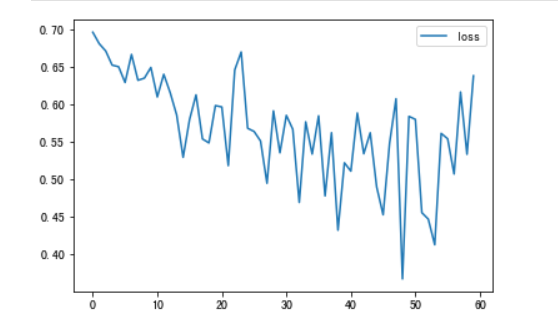


训练模型：

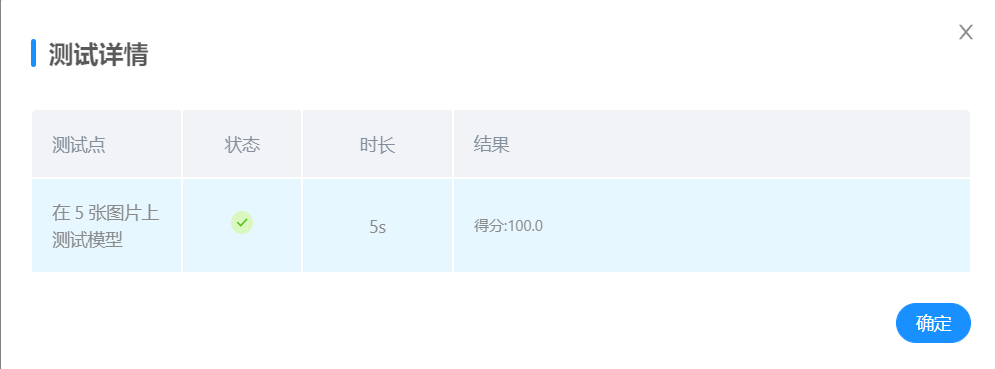


使用官方GPU可以迭代10次，这个不知道为什么不行，不过问题应该不大，最后跑出来了

训练过程展示：



最后结果：



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**）

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

基本达到预期结果，实现口罩佩戴检测；

对模型训练迭代的次数加以控制，过少模型效果不好，过多则会占用大量的资源，要找到平衡；

要防止深度学习的过拟合问题acc三次不下降就降低学习率；

在notebook上由于空间时间限制无法迭代10次，得不到结果，但如果将notebook导出为python脚本，用官方GPU则可以实现迭代10次，这样得到的模型用于预测效果好得多。