程序报告

学号：2112514　　姓名：辛浩然

1　问题重述

本实验的核心目标是构建一个目标检测模型，以识别图像中的人员是否佩戴口罩。为了达到这一目标，实验要求建立一个深度学习模型，能够有效地检测出人脸图像中是否佩戴口罩，并进一步优化模型性能。具体思路是利用现有的人脸检测模型，并在此基础上进行口罩识别模型的训练，以提高整体模型的准确性。

2　设计思想

充分使用已有的人脸检测的模型，再训练一个识别口罩的模型，从而提高训练的开支、增强模型的准确率。

人脸检测：直接使用现有的表现较好的 MTCNN 的三个权重文件，通过搭建 MTCNN 网络实现人脸检测。

口罩识别：加载预训练模型 MobileNet，手动调整学习率，调整参数，训练模型。

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 Pytorch 等深度学习框架训练模型。

3　代码内容

3.1　数据处理

首先，加载图像数据集并进行预处理和划分。使用 ImageFolder 加载数据集，并应用一系列预处理操作（调整大小、随机翻转、转换为张量和归一化处理）对图像进行处理。然后，根据给定的测试集比例将数据集随机划分为训练集和测试集。最后，创建训练集和测试集的 DataLoader，用于批量加载数据，并返回这两个数据加载器。这样，我们可以使用这些数据加载器在训练模型时轻松地迭代和处理图像数据。

```
def processing_data(data_path, height, width, batch_size, test_split=0.1):
    mtcnn_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(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]),  # 归一化处理
```

```
        ])

        # 加载数据集并进行划分
        dataset = ImageFolder(data_path, transform=mtcnn_transforms)
        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DataLoader
        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=128)
```

3.2 创建模型

使用现有的人脸检测模型，加载预训练模型 MobileNet 作为分类模型，并设置训练参数和优化器。

```
mtcnn_model = FaceDetector()  # 人脸检测模型

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

epochs = 70  # 训练的总轮数

optimizer = optim.Adam(model.parameters(), lr=1e-3)
# 创建Adam优化器，学习率为0，001

scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'max',
    factor=0.7, patience=2)
# 创建学习率调整策略，当评价指标不再提升时，将学习率乘以0.7，等待2个单位后再次调整

criterion = nn.CrossEntropyLoss()
# 创建交叉熵损失函数，用于计算预测值和目标值之间的差异
```

3.3 训练模型

对人脸口罩检测模型进行训练和验证。其中使用了 MTCNN 模型进行人脸检测和裁剪，使用 MobileNet 模型进行口罩检测。通过不断更新最佳损失和最佳准确率来保存表现最好的模型权重，以及记录每个训练周期的损失函数值。

```
best_loss = 1e9  # 初始化最佳损失
best_acc = 0  # 初始化最佳准确率为0
best_model_weights = copy.deepcopy(model.state_dict())
# 保存最佳模型权重的副本
best_model_weights_acc = copy.deepcopy(model.state_dict())
# 保存最佳准确率对应的模型权重的副本
loss_list = []  # 存储损失函数值的列表

for epoch in range(epochs):
    model.train()  # 将模型设置为训练模式

    for batch_idx, (images, labels) in tqdm(enumerate(train_data_loader, 1)):
        images = images.to(device)
        labels = labels.to(device)

        # 使用MTCNN检测人脸并裁剪出人脸区域
        faces = []  # 存储裁剪后的人脸图像
        for image in images:
            to_pil = transforms.ToPILImage()  # 创建一个将张量转换为PIL类型的转换器
            pil_image = to_pil(image.cpu())  # 将张量类型的图像转换为PIL类型的图像

            # 在图像上绘制人脸边界框
            detect_face_img = mtcnn_model.draw_bboxes(pil_image)
            faces.append(detect_face_img)

        # 将每个人脸图像转换为张量类型
        transform = ToTensor()
        # 创建一个将PIL类型图像转换为张量类型的转换器
        faces = [transform(face) for face in faces]
        # 转换每个人脸图像为张量类型
        faces = torch.stack(faces).to(device)
        # 将人脸图像堆叠为一个张量，并移动到设备上进行加速计算

        # 使用MobileNet进行口罩检测
        outputs = model(faces)  # 前向传播计算输出
```

```
        loss = criterion(outputs, labels)  # 计算损失函数值

        optimizer.zero_grad()  # 清零优化器的梯度
        loss.backward()  # 反向传播计算梯度
        optimizer.step()  # 更新模型参数

        if loss < best_loss:  # 判断当前损失是否是最佳损失
            best_model_weights = copy.deepcopy(model.state_dict())
            # 更新最佳模型权重的副本
            best_loss = loss  # 更新最佳损失

        loss_list.append(loss.item())  # 将损失函数值添加到损失列表中

    print('Epoch [{}/{}], Loss: {:.4f}'.format(epoch + 1, epochs, loss.
item()))
    # 打印当前训练周期的信息

    # 在验证集上评估模型
    model.eval()  # 将模型设置为评估模式
    with torch.no_grad():  # 不计算梯度
        total_correct = 0  # 总正确数
        total_samples = 0  # 总样本数

        for images, labels in valid_data_loader:
            images = images.to(device)
            labels = labels.to(device)

            faces = []
            for image in images:
                to_pil = transforms.ToPILImage()
                pil_image = to_pil(image.cpu())
                detect_face_img = mtcnn_model.draw_bboxes(pil_image)
                faces.append(detect_face_img)
            transform = ToTensor()
            faces = [transform(face) for face in faces]
            faces = torch.stack(faces).to(device)

            outputs = model(faces)  # 前向传播计算输出
            _, predicted = torch.max(outputs, 1)
            # 获取预测结果中的最大值及对应的索引
            total_samples += labels.size(0)
            # 更新总样本数
```

```
            total_correct += (predicted == labels).sum().item()
            # 更新总正确数

        accuracy = 100.0 * total_correct / total_samples
        # 计算准确率
        print('Validation Accuracy: {:.2f}%'.format(accuracy))
        # 打印验证集准确率

        if accuracy > best_acc:
         # 判断当前准确率是否是最佳准确率
            best_model_weights_acc = copy.deepcopy(model.state_dict())
            # 更新最佳准确率对应的模型权重的副本
            best_acc = accuracy
            # 更新最佳准确率

    scheduler.step(accuracy)  # 根据准确率更新学习率

# 训练结束，保存模型
torch.save(best_model_weights, './results/model.pth')
torch.save(best_model_weights_acc, './results/model_acc.pth')
```

3.4 模型预测

加载模型，使用加载的模型对图像进行分析，返回预测结果。

```
model_path = 'results/model.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
    """

    # 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
    # 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
```

4 实验结果

提交模型进行测试，实验结果如下：

5 总结

本次实验创建并训练了口罩佩戴检测模型，达到了较好的预期效果，实现了有效分类。可能有以下的优化方向：

1. 使用更高效的优化算法：当前使用的是 Adam 优化算法，可以尝试其他的优化算法，例如 SGD、RMSprop 等，并调整学习率和动量等超参数，以获得更好的模型性能。

2. 模型结构优化：考虑使用更深层次的网络结构或引入先进的视觉模型（如 ResNet、EfficientNet、MobileNet 等），以提高模型的表示能力和检测性能。

3. 数据平衡：确保训练数据集中的正负样本均衡，避免类别不平衡问题对模型性能的影响。可以采用随机采样、重采样或者引入类别权重等方法来解决这个问题。