2024《人工智能导论》大作业

（可参考修订）

任务名称：暴力图像检测

完成组号：10

小组人员：潘徐谦

完成时间：2024.5.31

1．任务目标

基于暴力图像检测数据集，构建一个检测模型。该模型可以对数据集的图像进行不良内容检测与识别。

2．具体内容

（1）实施方案

在 dataset.py 文件中进行数据的加载和处理，在 model.py 文件中初始化模型，在 train.py 和 continue_train.py 文件中进行模型的训练和继续训练，在 test.py 文件中对初步训练好的模型进行准确率的评估。利用 classify.py 文件作为接口调用模型。

（2）核心代码分析

对于 dataset.py 文件的实现，我使用了 PyTorch 和 PyTorch Lightning 库，定义了一个自定义数据集 CustomDataset 和一个数据模块类 CustomDataModule，用于处理图像分类任务。在 CustomDataset 类中，__init__ 方法接受数据目录 data_dir 和可选的图像变换 transform，并扫描目录中的.jpg 文件，将文件名存储在 image_files 列表中，用__len__ 方法返回数据集中图像文件的数量，用__getitem__ 方法根据索引读取图像文件，将其转换为 RGB 格式。在 CustomDataModule 类中，用__init__ 方法接受训练和验证数据目录 train_dir 和 val_dir，以及批量大小 batch_size 和工作线程数 num_workers，用 setup 方法根据阶段（训练、验证或测试）初始化相应的数据集，用 train_dataloader、val_dataloader 和 test_dataloader 方法分别提供训练、验证和测试数据加载器，配置批量大小、是否打乱数据和工作线程数。

对于 model.py 的实现，我定义了一个基于 PyTorch Lightning 的暴力分类器 ViolenceClassifier。__init__ 方法中，使用 ResNet18 架构。load_pretrained_weights 方法加载预训练模型的权重，并过滤掉不匹配的键。forward 方法定义了前向传播过程，直接调用 ResNet18 模型。training_step 方法计算训练批次的交叉熵损失。validation_step 方法计算验证批次的损失和准确率，并记录日志。test_step 方法计算测试批次的损失和准确率，并记录日志和保存输出。on_test_epoch_end 方法计算并记录测试阶段的平均损失和准确率。configure_optimizers 方法定义了 Adam 优化器和学习率调度器。

对于 train.py 的实现，先导入必要模块，包括 PyTorch Lightning 的 Trainer、回调函数 ModelCheckpoint 和日志记录器 TensorBoardLogger，以及自定义的模型 ViolenceClassifier 和数据模块 CustomDataModule，再定义训练和验证数据集的路径，以及 GPU ID、学习率、批量大小和日志名称。后面配置 Trainer，包括最大训练周期数、加速器类型（GPU）、设备 ID、日志记录器和回调函数，调用 data_module.setup(stage='fit') 方法，确保在训练之前正确初始化数据集，调用 trainer.fit(model, data_module) 开始模型训练。

对于 continue_train.py 的实现，先导入必要的模块：包括 PyTorch Lightning、模型检查点回调 ModelCheckpoint、自定义的数据模块 CustomDataModule 和模型 ViolenceClassifier，再定义主函数，包括定义训练和验证数据集的路径、批量大小、类别数量以及预训练模型的路径，创建 ViolenceClassifier 实例，指定类

别数量并加载预训练模型权重；设置 ModelCheckpoint 回调函数，监控验证准确率 val_acc，并保存验证准确率最高的模型；创建并调用 Trainer 实例，设置使用的设备数量（GPU）、最大训练周期数和回调函数。

对于 test.py 的实现，先导入必要模块，然后设置验证数据目录和批量大小，初始化数据模块，使用 ViolenceClassifier.load_from_checkpoint 方法，从指定的检查点路径加载模型权重，再调用 data_module.setup(stage='test') 方法，确保数据模块在测试阶段正确初始化，最后调用 trainer.test(model, data_module.val_dataloader()) 方法，在验证数据集上运行测试并评估模型性能。

对于 classify.py 的实现，先导入必要模块，再 v 定义图像分类函数 classify_image，包括使用 transforms.Compose 定义一系列预处理步骤，包括调整图像大小、转换为张量和归一化，使用 PIL 加载图像和模型推理。在主程序中，从命令行获取检查点路径和图像路径列表，从指定的检查点路径加载模型权重，并指定类别数量，再遍历图像路径列表，调用 classify_image 函数对每张图像进行分类，并输出预测结果。

对于已经训练过两次的模型 epoch=10-val_acc=0.99.ckpt，运行 test.py，输出结果如下图所示：

```
Testing DataLoader 0: 100%|██████████| 9/9 [00:08<00:00,  1.09it/s]
────────────────────────────────────────────────────────────────────
       Test metric             DataLoader 0
────────────────────────────────────────────────────────────────────
      avg_test_acc          0.9882446452476572
      avg_test_loss         0.04775475710630417
        test_acc             0.988256549232159
        test_loss           0.04887654632329941
```

平均测试准确率为 0.9882446452476572,平均测试损失为 0.04775475710630417, 单次测试准确率为 0.988256549232159, 单次测试损失为 0.04887654632329941, 这些结果表明模型在测试集上表现出色，具有高准确率和低损失，能够有效地分类暴力图像。

试运行 classify.py 文件，在命令行中输入 python classify.py "lightning_logs/version_4/checkpoints/epoch=10-val_acc=0.99.ckpt" "E:/test/test(1).jpg" "E:/test/test(2).jpg" "E:/test/test(3).jpg" "E:/test/test(4).jpg" "E:/test/test(5).jpg" "E:/test/test(6).jpg"
结果如下图所示：

```
(.venv) PS C:\Users\Lenovo\Desktop\pythonProject> python classify.py "lightning_logs/version_4/checkpoints/epoch=10-val_acc=0.99.ckpt" "E:/test/test(1).jpg" "E:/test/test(2).jpg" "E:/test/test(3).jpg" "E:/test/test(4).jpg" "E:/test/test(5).jpg" "E:/test/test(6).jpg"
[1, 0, 0, 0, 0, 0]
```

符合预期输出。

注：对于测试结果，尽可能给出分析图

3. 工作总结

（1）收获、心得

通过实现上述暴力图像分类模型，我深入理解了如何使用 PyTorch 加载预训练模型，了解了模型的加载与使用，并在推理阶段应用于新数据。我掌握了使用 torchvision.transforms 对图像进行标准化处理的重要性，确保了模型输入一致性和性能。通过实际应用深度学习技术，进一步理解了模型优化和性能调优的重要性。比如，通过实验调整超参数、优化数据预处理流程等，提升了模型的准确性和效率

（2）遇到问题及解决思路

训练好的模型经过 classify.py 的调用测试时只能输出全是 0 或 1 的数组，而使用 test.py 测试时却显示模型在测试集上有 99%的准确率。检查代码后发现，输入图像的预处理步骤不一致，导致模型推理结果不准确。解决时，修改所有的相关文件，统一使用 torchvision.transforms 进行标准化处理，包括调整图像大小、裁剪、归一化等，确保预处理步骤的一致性。修改后再使用 test.py 测试原模型发现准确率仅有 45%，考虑到这是一个二分类模型，故认为先前训练的模型是失败的，重新训练后成功。

4．课程建议

希望引入更多实际应用案例，帮助理解 AI 在不同领域的具体应用；希望在课堂上增加结合具体代码对训练模型方法的介绍。