2024《人工智能导论》大作业

任务名称：暴力图片检测模型

完成组号：13

小组人员：王旭升　王跃驰

完成时间：2024-06-21

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。
要求：

- 模型是 2 分类（0 代表正常图像、1 代表不良图像），分类准确率越高越好；
- 模型具有一定的泛化能力：不仅能够识别与训练集分布类似的图像，对于 AIGC 风格变 化、图像噪声、对抗样本等具有一定的鲁棒性；
- 有合理的运行时间。

2．具体内容

（1）实施方案

使用 **dataset.py/model.py/train.py** 建立并训练预估模型
使用 **shengcheng.py** 给图片数据集添加噪声
使用 **classify.py** 实现接口类
使用 **using.py** 接口类实例化

（2）核心代码分析

1. classify.py

接口类的实现

```
from model import ViolenceClassifier
from dataset import CustomDataModule
from dataset import CustomDataSet



class ViolenceClass:
    def init(self):

        #调用 ViolenceClassifier 的初始化函数

        ViolenceClassifier.__init__(self, num_classes=2,
learning_rate=1e-3)

    def load_checkpoint(checkpoint_path, device='gpu'):
        """
        加载 checkpoint 文件并返回模型

        Args:
            ViolenceClassifier: 模型类
```

```
            checkpoint_path: checkpoint 文件路径
            device: 设备类型，例如 'gpu' 或 'cuda'

        Returns:
            model: 加载后的模型
            epoch: 训练的 epoch 数
            loss: 训练的损失值
        """
        if not os.path.exists(checkpoint_path):
            raise FileNotFoundError(f'No checkpoint found at
{checkpoint_path}')

        # 加载 checkpoint

        checkpoint = torch.load(checkpoint_path, map_location=device)

        # 实例化模型

        model = ViolenceClassifier(learning_rate=lr)
        model.load_state_dict(checkpoint['model_state_dict'])

        # 获取其他信息

        epoch = checkpoint['epoch']
        loss = checkpoint['loss']

        print(f'Checkpoint loaded from {checkpoint_path}')
        return model, epoch, loss

    def classify(standard_list):
        """
        将张量传入并分类为预测列表
        参数：tensor_list (list): 列表张量
        返回：pred_list: 预测列表
        """
        output_list = self.model(standard_list)

        # 获取每个图像的预测类别

        pred_list = torch.argmax(output_list, dim=1)

        return pred_list

    def images_to_tensor(image_list):
```

```
        """
        将图片转换为张量
        参数: image_list (list): 图片 liebiao
        返回: standard_list: 转换并归一化后的张量
        """
        # 定义一个 ToTensor 转换

        to_tensor = transforms.ToTensor()

        # 定义一个 ToStandard 标准化

        to_standard = transforms.Normalize(mean=[0.485, 0.456, 0.406],
std=[0.229, 0.224, 0.225])

        # 将列表中的每个图像转换为 tensor，并存储在一个新的列表中

        tensor_list = [to_tensor(image) for image in image_list]

        # 将列表中每个 tensor 归一化， 并存储在一个新的列表中

        standard_list = [to_standard(tensor) for tensor in tensor_list]

        return standard_list

    def load_images_from_path(self, path):
        """
        从指定路径加载图像并转换为 RGB 格式
        参数: path (str): 图像文件夹的路径
        返回: list: 包含所有图像的列表，每个图像都是 PIL.Image 对象
        """
        images = []
        for filename in os.listdir(path):
            file_path = os.path.join(path, filename)
            if os.path.isfile(file_path) and
filename.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif')):
                img = Image.open(file_path).convert('RGB')
                images.append(img)
        return images


    def accuracy_score(y_true, y_pred):
        """
        计算模型预测的准确率。
        参数:
        y_true (list or numpy array): 真实标签。
```

```
        y_pred (list or numpy array): 预测标签。
        返回:float: 准确率，范围在 0 到 1 之间。
        """
        # 确保输入是可迭代对象
        if len(y_true) != len(y_pred):
            raise ValueError("真实标签和预测标签的长度必须相同")

        # 计算匹配的样本数
        correct = sum(yt == yp for yt, yp in zip(y_true, y_pred))

        # 计算准确率
        accuracy = correct / len(y_true)

        return accuracy
```

2. using.py

接口类实例化

```
from classify import ViolenceClass


if __name__ == "__main__":

    # 实例化该接口类

    using_example = ViolenceClass()

    # 加载 checkpoint 并返回模型

    checkpoint_path = 'path/to/your/checkpoint.pth'
    model, epoch, loss = using_example.load_checkpoint(checkpoint_path,
device='gpu')

    # 打印轮次，损失值

    print(f'Model loaded. Epoch: {epoch}, Loss: {loss}')

    # 从指定路径批量加载图片,返回一个图片列表，图片格式为 RGB

    img_example_list =
using_example.load_images_from_path('/path/to/your/image')
//"C:/Users/wyc/Desktop/暴力图片检测模型/output/output"
```

```
    # 转换图片列表到 tensor 格式

    imgs_example = using_example.image_to_tensor(img_example_list)

    # 调用模型分类

    predicted_output = using_example.classify(imgs_example)

    # 计算正确率并打印

    examples_accurancy = using_example.accuracy_score(using_example.label,
predicted_output)
    print(examples_accurancy)

    # 获得预测列表并打印
    examples_pred = using_example.classify(imgs_example)
    print(examples_pred)
```

3. shengcheng.py

为图片数据集添加噪音

```
import os
import cv2
import numpy as np


def add_poisson_noise(image, scale=1.0):
    """
    给图像添加泊松噪声
    :param image: 输入图像，numpy 数组
    :param scale: 噪声强度，默认为 1.0
    :return: 添加噪声后的图像
    """
    # 将图像转换为浮点数类型
    img_float = image.astype(np.float32) / 255.0

    # 生成泊松噪声
    noise = np.random.poisson(img_float * scale) / scale

    # 将噪声添加到图像中
    noisy_image = img_float + noise

    # 将图像值裁剪到[0, 1]范围内
    noisy_image = np.clip(noisy_image, 0, 1)
```

```
    # 将图像转换回 8 位无符号整数类型
    noisy_image = (noisy_image * 255).astype(np.uint8)

    return noisy_image


def process_images_in_directory(input_dir, output_dir, scale=1.0):
    """
    批量处理目录中的所有图片，添加泊松噪声并保存到输出目录
    :param input_dir: 输入目录路径
    :param output_dir: 输出目录路径
    :param scale: 噪声强度，默认为 1.0
    """
    # 确保输出目录存在
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    # 遍历输入目录中的所有文件
    for filename in os.listdir(input_dir):
        # 检查文件是否为图片文件
        if filename.endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif')):
            # 读取图像
            image_path = os.path.join(input_dir, filename)
            image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

            # 添加泊松噪声
            noisy_image = add_poisson_noise(image, scale)

            # 保存添加噪声后的图像
            output_path = os.path.join(output_dir, filename)
            cv2.imwrite(output_path, noisy_image)
            print(f"Processed and saved {output_path}")


# 输入和输出目录路径
input_directory = "D:\workspace\python\input"
output_directory = "D:\workspace\python\output"


# 批量处理图片
process_images_in_directory(input_directory, output_directory,
scale=20.0)
```

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

学习建立一个二分类模型、训练模型、调用实例，提升了自己的代码实践能力。

（2）遇到问题及解决思路

问题 1：**from torchvision import models** 调用失败

4．课程建议

给予更多扩展内容进行学习，结合不同专业提供任务，可以更好提升学生专业水平和实践能力