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

```
import os
import torch
from torcimport os
import torch
from torch import nn
from PIL import Image
from torchvision import transforms
from torchvision import models

class ViolenceClass:
    def init(self):

        self.model = models.resnet18(pretrained=True)
        num_ftrs = self.model.fc.in_features
        self.model.fc = nn.Linear(num_ftrs, 2)

    def classify(self, output_list):
        """
        将张量传入并分类为预测列表
```

```
        参数：output_list (list): 列表张量
        返回：pred_list: 预测列表
        """
        # 获取每个图像的预测类别

        probs_list = [nn.Softmax(dim=0)(logits) for logits in
output_list]

        pred_list = [torch.argmax(probs).item() for probs in probs_list]

        return pred_list

    def images_to_tensor(self, image_list):
        """
        将图片转换为张量
        参数: image_list (list): 图片列表
        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()

        # 定义一个归一化函数

        def normalize(tensor):
            min_val = torch.min(tensor)
            max_val = torch.max(tensor)
            normalized_tensor = (tensor - min_val) / (max_val - min_val)

            return normalized_tensor

        # 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 = [normalize(tensor) for tensor in tensor_list]

        # 为每个 tensor 添加批次维度

        input_list = [tensor.unsqueeze(0) for tensor in standard_list]

        return input_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 label_true(self, split):
        """
        获得数据集真实标签。
        参数：split：数据集类型名 + '/'
        返回：label：真实标签列表
        """

        #获得文件路径

        assert split in ["train/", "val/", "test/"]
        #data_root = "/your/path/to/violence_224/"
        data_root = "C:/Users/wyc/Desktop/violent_test/violence_224/"
        data_list = [os.path.join(data_root, split, i) for i in
os.listdir(data_root + split)]

        label = [int(data.split("/")[-1][0]) for data in data_list]  # 获
取标签值，0 表示正常，1 表示暴力

        return label

    def accuracy_score(self, 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

```
import torch
import os
from classify import ViolenceClass
from model import ViolenceClassifier

if __name__ == "__main__":

    # 实例化模型和接口

    model = ViolenceClassifier(learning_rate=3e-4)
    using_example = ViolenceClass()

    # checkpoint_path = 'path/to/your/checkpoint.pth'
    checkpoint_path =
'C:/Users/wyc/Desktop/violent_test/train_logs/resnet18_pretrain_test/ve
rsion_0/checkpoints/resnet18_pretrain_test-epoch=08-val_loss=0.05.ckpt'

    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=torch.device('cpu'))

    # 加载模型
```

```
    model.load_state_dict(checkpoint['state_dict'])
    model.eval()

    # 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')
    img_example_list =
using_example.load_images_from_path('C:/Users/wyc/Desktop/violent_test/
violence_224/test')

    # 转换图片列表到 tensor 格式

    imgs_example = using_example.images_to_tensor(img_example_list)

    # 使用模型进行推理
    with torch.no_grad():
        output_list = [model(img) for img in imgs_example]

    # 调用模型获得真实和预测标签
    label_true = using_example.label_true("test/")
    label_predict = using_example.classify(output_list)

    # 计算正确率并打印

    examples_accurancy = using_example.accuracy_score(label_true,
label_predict)
    print(examples_accurancy)

    # 获得预测列表并打印
    print(label_predict)
```

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)
```

3. 工作总结

（1）收获、心得

1.学习了 anaconda 安装以及虚拟环境配置

2.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

思路：单独卸载 torchvision 使用 pip 重新安装

问题 2：如何调用训练后模型

思路：先加载 checkpoint 再将状态导入实例化模型

问题 3：如何获得预测标签

思路：使用 softmax 将模型输出的分类分数转换为概率，再获得概率最大值的索引

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