# 2024《人工智能导论》大作业

（可参考修订）

任务名称：暴力图像二分类检测模型及其接口的实现与实验

完成组号：7

小组人员：邓迪、苏煐泽

完成时间：2024-06-17

# 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。

模型是二分类模型，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。

为便于他人使用模型，我们实现了一个接口类。其提供的函数接受 tensor 向量化的 n 张固定大小图片，并输出以 n 长数组表示的分类结果。

# 2. 具体内容

## 2.1. 实施方案

### 2.1.1. 模型实现

我们利用文档里提供的几部分代码，成功进行了模型的代码实现，并进行了一些调参优化工作，以实现更好性能。

第一次实验的参数如下：
max_epochs=10
batch_size = 128
lr = 3e-4
模型测试结果：

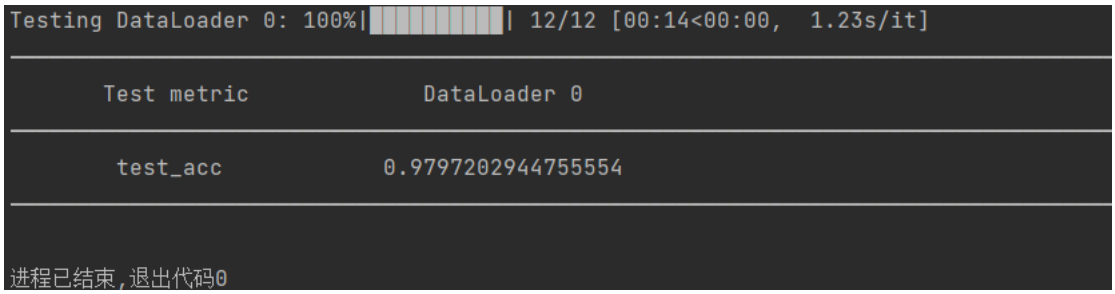
```
Testing DataLoader 0: 100%|██████████| 12/12 [00:12<00:00,  1.06s/it]
──────────────────────────────────────────────────────────────────────
       Test metric             DataLoader 0
──────────────────────────────────────────────────────────────────────
        test_acc            0.9860140085220337
──────────────────────────────────────────────────────────────────────
```

第二次实验的参数如下：
max_epochs=40
batch_size = 128
lr = 3e-4
模型测试结果：

```
Testing DataLoader 0: 100%|██████████| 12/12 [00:12<00:00,  1.07s/it]
──────────────────────────────────────────────────────────────────────
       Test metric             DataLoader 0
──────────────────────────────────────────────────────────────────────
        test_acc            0.9650349617004395
──────────────────────────────────────────────────────────────────────

进程已结束,退出代码0
```

第三次实验的参数如下：
max_epochs=20
batch_size = 128
lr = 3e-4
模型测试结果：

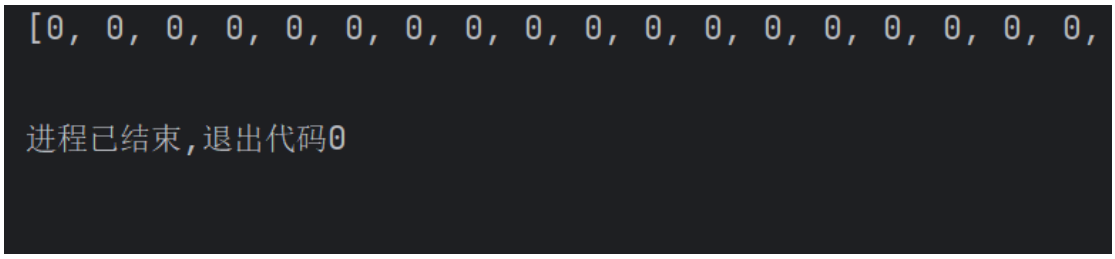



由三次实验可知，max_epochs 设为 10 可以得到一个精确度较高的模型。由于时间因素，我们未尝试对 lr 等参数进行调整。

### 2.1.2. 接口实现

为便于他人使用模型，实现一个接口类就成了模型之外的必要部分。设计中，其提供的函数应接受 tensor 向量化的 n 张固定大小图片，并输出以 n 长数组表示的分类结果。

在之前训练好的模型基础上，我们定义了调用模型的接口。通过给定的 dataset.py，我们得到了 n 张大小的图片数组用作测试。可以看到，其输出了模型对此的预测结果。




（注 1：由于该 batch 的测试集图片顺序未被打乱，模型预测得出连续的相同结果，这是正确的）

（注 2：接口类和使用实例的代码见 2.2.）

### 2.1.3. 复杂测试集测试

先前的测试、调参是使用与训练集同源的图片作为测试集，为充分检验模型效果，现引入更多更复杂的测试集。

#### 2.1.3.1. AIGC 生成图片集

使用 AIGC 生成图片，扩大测试集范围，以实现多样化测试。

通过以下代码，将生成图片处理为模型能接受的形式。

```python
from PIL import Image
import os


1 个用法
def batch_resize(input_folder, output_folder, target_width, target_height):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    files = os.listdir(input_folder)
    image_files = [f for f in files if f.endswith('.jpg') or f.endswith('.png')]
    count = 0
    for image_file in image_files:
        image_path = os.path.join(input_folder, image_file)
        img = Image.open(image_path)
        img_resized = img.resize((target_width, target_height), Image.LANCZOS)
        output_filename = f"1_{count:04d}.jpg"
        output_path = os.path.join(output_folder, output_filename)
        img_resized.save(output_path)
        count += 1


if __name__ == "__main__":
    input_folder = r"C:\Users\HUAWEI\Desktop\AI‘\violence_224\aigcTest"
    output_folder = r"C:\Users\HUAWEI\Desktop\AI‘\violence_224\aigcTest2"
    target_width = 224
    target_height = 224

    batch_resize(input_folder, output_folder, target_width, target_height)
```

结果如下：

```
Testing DataLoader 0: 100%|██████████| 2/2 [00:07<00:00,  3.53s/it]
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃        Test metric        ┃       DataLoader 0        ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
│         test_acc          │    0.8168317079544067     │
└━━━━━━━━━━━━━━━━━━━━━━━━━━━┴━━━━━━━━━━━━━━━━━━━━━━━━━━━┘

进程已结束,退出代码0
```

可见模型对于范围更广泛的图片有一定辨识能力，但仍有改进空间。

### 2.1.3.2. 噪声处理图片集

通过以下代码引入随机噪声，模拟现实中监控设备不清晰的情况。

```python
def add_noise_to_image(image_path, output_path, noise_size):
    image = Image.open(image_path)
    width, height = image.size
    noisy_image = Image.new('RGB', (width, height))
    for x in range(width):
        for y in range(height):
            pixel_color = image.getpixel((x, y))
            noise = (random.randint(-noise_size, noise_size),
                     random.randint(-noise_size, noise_size),
                     random.randint(-noise_size, noise_size))
            new_color = tuple(map(lambda i, j: max(0, min(255, i + j)), pixel_color, noise))
            noisy_image.putpixel((x, y), new_color)
    noisy_image.save(output_path)


1 个用法
def add_noise_to_images_in_folder(input_folder, output_folder, noise_size):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    for file_name in os.listdir(input_folder):
        if file_name.endswith('.jpg') or file_name.endswith('.png'):
            input_file_path = os.path.join(input_folder, file_name)
            output_file_path = os.path.join(output_folder, file_name)
            add_noise_to_image(input_file_path, output_file_path, noise_size)


input_folder = r"C:\Users\HUAWEI\Desktop\AI‘\violence_224\test"
output_folder = r"C:\Users\HUAWEI\Desktop\AI‘\violence_224\noiseTest"
noise_size = 20
add_noise_to_images_in_folder(input_folder, output_folder, noise_size)
```

结果如下：

```
Testing DataLoader 0: 100%|██████████| 3/3 [00:14<00:00,  4.87s/it]
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃        Test metric        ┃       DataLoader 0        ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
│         test_acc          │    0.9356912970542908     │
└━━━━━━━━━━━━━━━━━━━━━━━━━━━┴━━━━━━━━━━━━━━━━━━━━━━━━━━━┘

进程已结束,退出代码0
```

可见其有一定抗噪声能力。

## 2.2. 核心代码分析

### 2.2.1. classify.py 接口类分析

以下是对接口的定义代码。

```python
from pytorch_lightning.loggers import TensorBoardLogger
import torch
from model import ViolenceClassifier
1 个用法
class ViolenceClass:
    def __init__(self, gpu_id, ckpt_root, ckpt_model, batch_size=128, log_name="resnet18_pretrain"):
        self.gpu_id = gpu_id
        self.batch_size = batch_size
        ckpt_path = ckpt_root + ckpt_model
        self.logger = TensorBoardLogger("test_logs", name=log_name)

        self.model = ViolenceClassifier.load_from_checkpoint(ckpt_path)

    1 个用法
    def classify(self, imgs):
        self.model.eval()
        with torch.no_grad():
            preds = []
            for img in imgs:
                output = self.model(img.unsqueeze(0))
                _, predicted = torch.max(output, 1)
                preds.append(predicted.item())

        return preds
```

接口的构造函数首先接受 GPUid、已训练模型文件的位置（ckpt_root+ckpt_model），batch 大小，以此定义接口使用的硬件、模型和接收图片大小。

接口的分类函数 classify 则接受 imgs（一个大小为 n*3*224*224 的 tensor），随后 model.eval()设置模型为评估模式，遍历 imgs 中的所有图片，计算模型的输出量。通过 torch.max 取得 output 中的分类结果，类型转换后存储到 preds 中。

### 2.2.2. main.py 使用实例分析

以下是接口类的使用实例之一。

```python
import torch


import classify
from dataset import CustomDataModule

if __name__ == '__main__':
    batch_size = 128
    data_module = CustomDataModule(batch_size=batch_size)
    data_module.setup()
    imgs = torch.ones(batch_size, 3, 224, 224)
    for i in range(0, batch_size, 1):
        imgs[i] = data_module.test_dataset.__getitem__(i + 128)[0]
    model = classify.ViolenceClass(gpu_id=[0], ckpt_root="C:/Users/HUAWEI/Desktop/AI'/train_logs/",
                                   ckpt_model="resnet18_pretrain_test/version_0/checkpoints/resnet18_pretrain_test"
                                              "-epoch=20-val_loss=0.04.ckpt",
                                   batch_size=batch_size)
    ans = model.classify(imgs)
    print(ans)
```

首先，实例指定了输入图片的数量 batch_size，并调用 dataset.py 中的函数建立了数据集 data_module，将其初始化。随后，使用数据集的既有方法得到 imgs（一个大小为 n*3*224*224 的 tensor），作为标准格式的输入。

随后，调用接口类的构造函数，指定 GPUid、已训练模型文件位置，batch 大小。

调用接口类的分类函数，得到标准格式的结果并输出。

# 3. 工作总结

## 3.1. 收获、心得

### 3.1.1. 邓迪心得

本次实验中，我进行了接口类的实现工作，这对于我是首次考虑模型的易用性问题。在过去浅尝辄止的尝试中，我满足于模型的训练和测试，而没有考虑其如何被使用，也没有考虑其具体的结果产生过程。

这次大作业一方面让我进行了模型接口设计、基于 github 平台的代码共享等崭新的实践，另一方面也促使我不满足于对模型模块的拼凑，而更加深入地理解了模型的运作。

### 3.1.2. 苏煐泽心得

第一次成功训练模型并有好的测试结果的这次尝试，相当于打开人工智能领域的大门，看到了这个领域怎么做以及能得到什么结果，虽然只是一次小的尝试，但对于模型的训练与测试有了初步的认识。希望以后能继续深入的学习与实践。

## 3.2. 遇到问题及解决思路

### 3.2.1. 人员短缺问题

由于项目开始时间较晚，叠加期末复习临近，我们小组出现了人头数和单人工时的双重短缺。项目因而出现了人力上的困难。

对此，我们一方面倒排工期，制定严格时间表和分工；另一方面充分发挥个人能动性，见缝插针进行项目代码的设计开发。

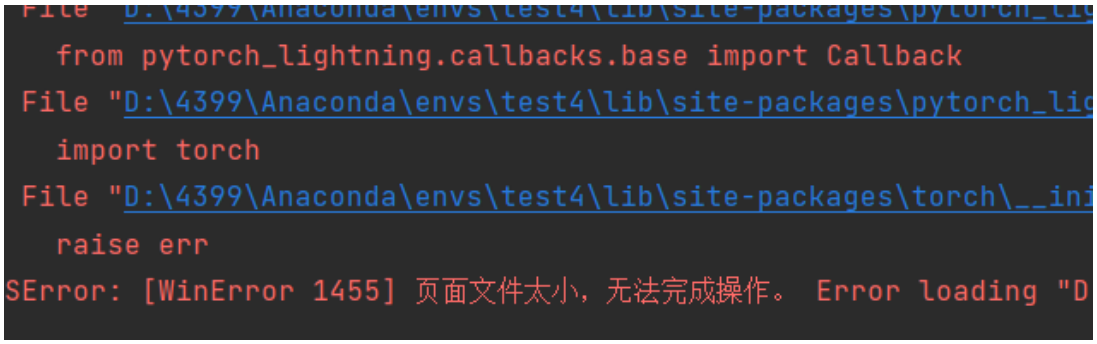
### 3.2.2. 训练速度问题

由于组员邓迪的计算机未安装独立显卡，只能使用 cpu 作为训练的设备，因而出现训练缓慢的问题。一次训练往往需要耗费七至八个小时，严重影响了迭代开发、问题发现与解决的效率。

在组员苏煐泽完成了 cuda 驱动的配置，开始使用其电脑的 NVIDIA 显卡进行训练后，情况有了明显改观。一次训练（10 轮）仅需十分钟即可完成，大大加速了我们完善代码的工作。

由此我们深刻认识到，显卡硬件对于人工智能模型学习的重要性。

### 3.2.3. 虚拟内存问题




运行环境所在的文件夹在计算机 D 盘中，查找发现计算机默认情况下没有给 D 盘分配虚拟内存。程序运行时没有分配虚拟内存，就遇到了上面的问题。

对此，我们在高级系统设置中给 D 盘分配虚拟内存，问题也就得以解决。

## 4. 课程建议

希望能在课堂具体讲或者介绍大作业相关的实践操作，在理论教学的课堂中加上人工智能实践相关的作业或练习。

大作业亲自尝试训练大模型很有趣也很有学习的价值，希望以后的课程能接触更多更新颖的模型。