2024《人工智能导论》大作业

0. 导言概况

任务名称：数据集构建+接口实现

完成组号：　6

小组人员：　阮亿康

完成时间：2024/6/21

1. 任务目标

1. **构建测试集**：从训练集中抽取数据构建测试集（test1），构建一个AIGC生成图像的测试集（test2），构建一个一个加上了图像噪声的测试集（test3）
2. **实现Classify.py接口文件**：提供一个接口classify，接收形状为[n,3,224,224]的tensor输入，输出对应长度为n的列表（每个值对应预测的类别，即0或1）

2. 实施方案

part0：模型的训练

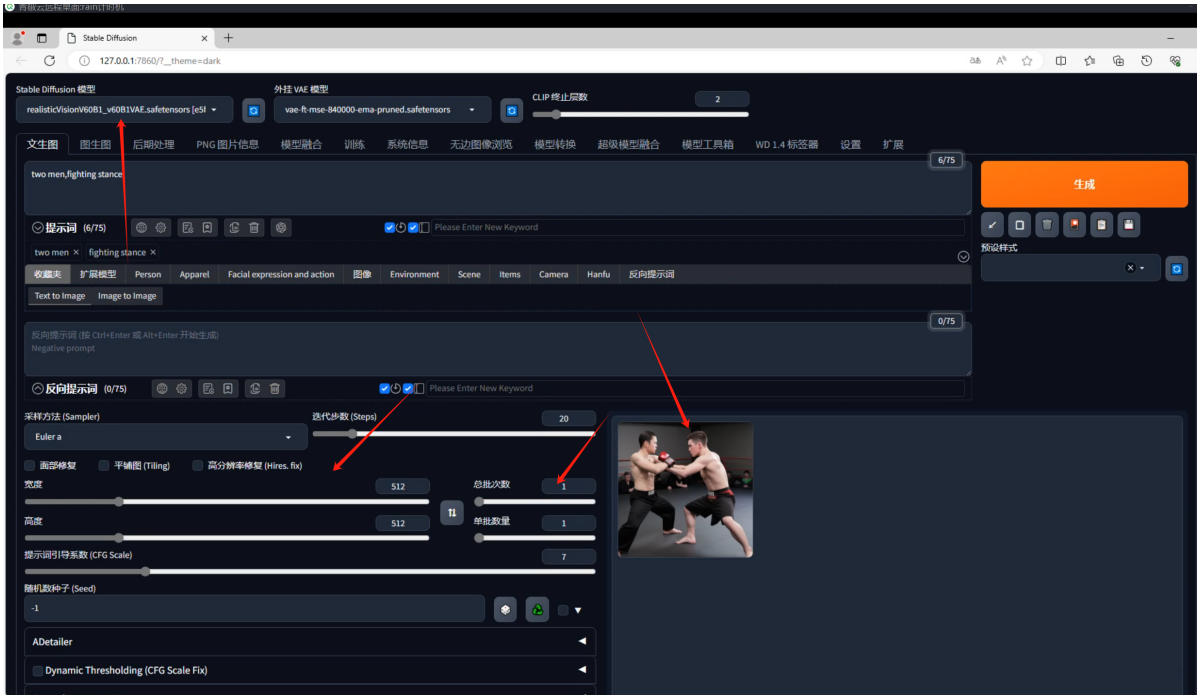
在已有代码的基础上，调整训练模型使用的device以及batch_size、max_epochs，lr等具体参数即可，这里我使用gpu进行训练，并设置lr=3e-4，batch_size=128，max_epochs=40。

训练方式：在train.py文件中调整参数，运行train.py文件即可。

part1：测试集的构建

1.AIGC测试集的构建

总体上，使用 **stable diffusion**(这里主要选择了**realistic vision**的生成模型)，通过较为精准的prompt生成需要的暴力图像和非暴力图像。



补充说明：由于**[224, 224]**的比例不是2的次方相乘，故直接stable diffusion生成效果较差，这里先生成**[512, 512]**的图像，该经过转换得到**[224, 224]**。

2.加噪后生成测试集

通过给图像添加高斯噪声得到test3。

part2：classify接口的实现

外部接口调用方式：

```
python classify.py --classify img_path(图片所在文件夹)
```

3. 代码文件功能说明

见Readme.md

4. 核心代码分析：

part1：已有代码部分分析

1. dataset.py

```
    def __getitem__(self, index):
        img_path = self.data[index]
        x = Image.open(img_path)
        y = int(img_path.split("/")[-1][0])  # 获取标签值，0代表非暴力，1代表暴力
        x = self.transforms(x)
        return x, y
```

通过读取图片的名的方式来获取label，而不是图片文件和标签文件独立的方式。这里还将jpg文件进行transform操作，转换为tensor（train时还进行随机翻转）。

2. model.py

```
    def training_step(self, batch, batch_idx):
        x, y = batch
        logits = self(x)
        loss = self.loss_fn(logits, y)
        self.log('train_loss', loss)
        return loss
```

以training_step为例，self(x)指向self.forward()也即self.model()方法，调用前向过程获得logits，再根据logits进行loss计算，返回loss给模型内部进行优化过程。

3. train.py和test.py

```
trainer = Trainer(
    max_epochs=40,
    accelerator='gpu',
    devices=gpu_id,
    logger=logger,
    callbacks=[checkpoint_callback]
)
```

train.py中设置模型检查点checkpoint用来保存训练过程中出现的最佳模型。使用Trainer加载训练器，确定训练过程的轮次（epoch）,使用加速器，训练过程记录的logger，以及最佳模型指向的checkpoint。调用trainer的fit方法。

test.py中也设置trainner，但调用trainer的test方法。

part2：添加代码部分分析

核心代码文件（**classify.py**）分析如下：

整体逻辑：

1. 从命令获取参数，解析出待分类图片路径
2. 将路径传递给transAndclassify函数进行预处理
3. classify进行分类

核心函数：

1. transAndclassify（数据预处理工作）

```
def transAndclassify(self,imgs_dir):
        #定义一个transform操作，把图片转换为tensor并归一化
        transform = transforms.Compose([
            transforms.ToTensor(),
            #归一化，为了简单这里就用基于ImageNet得到的标准化参数
            #transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
0.224, 0.225])
        ])

        image_tensors = [] #用来存储单个图片张量
        #遍历目录下的所有文件
        for imgname in os.listdir(imgs_dir):
            img_path = os.path.join(imgs_dir,imgname) #获取图片完整路径
            image = Image.open(img_path).convert("RGB")
            image_tensor = transform(image) #单个图片tensor
            image_tensors.append(image_tensor)
        #把图片张量列表堆叠成一个批次张量
        batch_tensor = torch.stack(image_tensors).to(self.device)
        return self.classify(batch_tensor)
```

该函数功能：读取图片，将图片转化为tensor形式，组织起一个batch_tensor传递给classify函数

2. classify（**核心函数**）

```
    def classify(self, imgs : torch.Tensor) -> list:
        # 图像分类
        #走一个forward过程，得到一个logits作为元素形成的tensor
        imgs = imgs.to(self.device)
        vio_possibility = self.model(imgs)
        _, preds =torch.max(vio_possibility,1)#找到对应的预测类别
        return preds.cpu().tolist()#结果运转回到cpu
        #return preds.tolist()
```

a) 首先，将位于cpu上的图像数据，转移到gpu上，因为我们将模 型挂载到了gpu上。

b) 然后，进行一次模型的forward过程，将imgs传递给模型，返 回一个logits组成的张量，其形状为torch.Size([n, 2]) (n代表样本 数，2表示暴力/非暴力2个类别)。

c) 然后，对logits每行两个元素进行比较大小，大的元素对应的位 置即为预测的类别。preds的形状为torch.Size([n])

d) 最后，将预测的preds转换为待输出的列表形式，并运转回cpu。

5. 测试结果及分析

1. test1（同源数据集共4000张jpg）：

```
sjtu@sjtu:~/ryk_file_private_job/my_job$ python test.py
/home/sjtu/anaconda3/lib/python3.9/site-packages/torchvision/io/
image.py:13: UserWarning: Failed to load image Python extension:
 libtorch_cuda_cu.so: cannot open shared object file: No such fi
le or directory
  warn(f"Failed to load image Python extension: {e}")
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
You are using a CUDA device ('NVIDIA GeForce RTX 3090') that has
 Tensor Cores. To properly utilize them, you should set `torch.s
et_float32_matmul_precision('medium' | 'high')` which will trade
-off precision for performance. For more details, read https://p
ytorch.org/docs/stable/generated/torch.set_float32_matmul_precis
ion.html#torch.set_float32_matmul_precision
Missing logger folder: test_logs/resnet18_pretrain
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0,1,2,3]
Testing DataLoader 0: 100%|          | 32/32 [00:02<00:00, 13.48it/s]
```

Test metric	DataLoader 0
test_acc	0.9987509250640869

2. test2（AIGC测试集,共400张jpg）：

```
Testing DataLoader 0: 100%|          | 4/4 [00:01<00:00,  2.78it/s]
```

Test metric	DataLoader 0
test_acc	0.6439909338951111

3. test3（加噪测试集，和训练集同等大小 ）：

```
  return F.conv2d(input, weight, bias, self.stride,
Testing DataLoader 0: 100%|          | 61/61 [00:04<00:00, 13.92it/s]
```

Test metric	DataLoader 0
test_acc	0.5766451358795166

可以见的，模型在test1上表现非常出色，在test2上表现良好，在test3上表现最差。

结果分析：

1. prompt的不准确，stable diffusion使用的不熟悉，导致部分AIGC图片质量很低，标签和实际内容不匹配。图片在[512,512]向[224,224]转化的过程中可能存在失真。
2. 噪声对图像分类的干扰很大。

6. 工作总结

收获和心得

本次实验中，我的收获和心得在于：

1. 初步学会了stable diffusion的部署和使用
2. 掌握了给图片加噪声的方法
3. 学会了模型的基本训练方法和测试方法（Trainer）
4. 学会了图片文件+标签文件之外的打标签的方法——图片命名中包含标签的方法
5. 学会了更细粒度操作模型的方式（见classify文件）
6. 学会了在编程时关注数据是在cpu上还是gpu上

遇到的问题及解决方式

1. stable diffusion模型选择问题

   刚上手sd的时候，无论怎么调整prompt，生成的效果都是二次元风格的。后来，注意到此时sd挂载到就是一款二次元风格的模型，后面改换更偏向于写实风格的realist vision模型，图片生成效果大大提升。

2. 数据位置问题

   在train和test过程中，有封装好的Trainer，只需指定device即可。而在classify时，需要自主处理图片转化为tensor，并且自主将模型挂载到cpu/gpu上。当我把模型挂载到gpu上后，就需要将部分数据也移植到gpu上。

   在本次作业中，图片读取，图片转化为tensor过程都是在cpu上进行的；转化出tensor后，我将tensor转移到gpu上，此时模型已经挂载到gpu上，这样才能有效利用数据。

7. 课程建议和感悟

其实，在上课的过程中我感觉这门课的东西都太浅了，但现在全部学完了之后，感觉这样的安排确实可以让我们对AI有一个较为全面的了解，有助打一个基础吧。本次大作业过程中，我选择一个人成组就是想体验一下从模型训练，到数据集制作，再到模型调用的全过程，说实话，完成这一项大作业，我觉得我有了不小的提升。

至于建议，我建议可以在日常教学周中，布置一些小的实训作业吧。