# 《人工智能导论》大作业

任务名称.	其于 Resnet	的暴力图像检测
	AS I MESHEL	1171 M 181 121 138 119 119 119 119 119 119 119 119 119 11

完成组号: \_\_\_\_\_5

小组人员: 见纸质报告或云盘报告

完成时间: 2024.6.20

# 1. 任务目标

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,且具有一定的泛化能力,即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,同时有合理的运行时间。

# 2. 具体内容

# (1) 实施方案

模型创建、训练与测试: 完善支持文档中的代码,利用 resnet18 模型对图像进行二分类;利用 classify. py 实现接口类 ViolenceClass,类 的 初 始 化 函 数 中 完 成 模 型 加 载 等 初 始 化 工 作; 利 用 run\_classify. py 脚本调用接口,实现模型的简便训练、测试和图像分类功能

**数据集创建**:对于AIGC数据集,我们选择使用 stable diffusion 生成暴力图像样本;对于对抗样本数据集,我们利用自己写的 augmentation.py 脚本对原数据集分别进行了加高斯噪声和加椒盐噪声的处理。

# (2) 核心代码分析

模型训练、测试、图像分类部分:

### dataset.py:

该部分主要实现了 lightning 模块中数据的加载。

CustomDataset 是用以保存数据集的类,该保存了所有图片的路径,在加载时的 getitem 函数中再打开对应的图片,同时通过文

件名的尾缀标定图像的实际类别。实际输出为转化为 tensor 的图像 及其标签。

CustomDataModule 是用以训练的数据模块类,继承了LightningDataModule,里面包含三个dataset,分别为训练集train,验证集val和测试集test。每一个数据集是一个CustomDataset类,而需要时,返回对应dataset类的dataloader。

PredDataset 与 CustomDataset 类似,但在加载时进行了形式转换,将图像标准化为 3\*224\*224 的格式,同时不含有图像标签。用以进行图像的类别预测。而 pred\_dataloader 函数则作为 PredDataset 的调用接口,返回其对应的 dataloader。

# classify.py:

该文件实现了接口类 ViolenceClass。

在初始化函数\_\_init\_\_中,调用了 config\_get 函数进行了模型、 路径配置与模型加载工作等,同时定义了图像预处理变换 transform。

config\_get 函数作为初始化的一部分,读取了 config.txt 的数据,从中获得当前模型的保存路径,训练 gpu 编号以及学习率、batch大小等信息。

train\_model 函数通过 lightning 的 Trainer 模块实现了对模型的训练。

test 函数同样通过 Trainer 模块对模型进行测试,用以分析模型的正确率。

classify 函数实现了对输入图像进行类别预测,其输入为n\*3\*224\*224 的 torchTensor,代入模型进行预测分析得到每种类型的可能性值,再使用 max 函数比较返回可能性值更高的类别。

classify2 函数通过 lightning 的 Trainer 对图像进行预测,由于使用了 PredDataset 保存数据集,故不需要一次性读入全部的图像数据,适用于更大规模的数据集。

# run\_classify.py:

该文件实现了对 ViolenceClass 的调用。通过 ArgumentParser 实现了对模型的不同调用方式。

- --Train 和--test 分别是对模型的训练与测试,其直接调用 violenceclass 的对应模块。
- ——classify 是对图像输入的类别预测,需要同时输入包含所有图像路径的目录文件路径,程序将会对文件中的路径进行图像加载,使用 violenceclass 中的 transform 将数据转化为符合要求的 Tensor,调用 classify 函数进行预测,并输出对应的类别。
- --classify2 则是另一种类别预测接口,通过--img 参数或者 config 文件中的 data-pred 输入待预测图像的文件夹路径,随后程序将调用 classify2 函数进行预测。

# 数据集部分:

# AIGC 数据: 利用 stable diffusion 生成

在生成阶段, Stable Diffusion 模型从随机噪声开始,逐步去噪生成清晰图像。这个过程如下:

- 1. 初始化: 从随机噪声开始。
- 2. 逆向扩散:通过神经网络预测当前噪声并去除相应噪声,生成下一步的图像。
- 3. 多次迭代: 重复上述过程, 直至生成最终的清晰图像。

通过更换具体模型,修改关键词等,我们生成了不同组不同画风的不同场景的暴力图像,但是由于部分模型过于简易未经过大量的训练,生成图片效果可能欠佳。

# 对抗样本数据: 使用 augmentation.py 脚本处理原数据集

为了增强模型的泛化能力,通过添加噪声的方式生成了对抗样本,进行数据增强。在本次大作业中,我们使用了常用的椒盐噪声和高斯噪声。

椒盐噪声又称脉冲噪声,它随机改变一些像素值,在二值图像上 表现为使一些像素点变白,一些像素点变黑。是由图像传感器,传输 信道,解码处理等产生的黑白相间的亮暗点噪声。

在代码中,使用 prob 和 thres 变量分别控制了白点、黑点产生概率,而 prob 和 thres 变量的取值则是由 max\_noise 和 min\_noise 变量控制,这两者分别决定了白点与黑点产生概率取值的上下限,使得

对不同图像样本而言噪点密度略有不同。

高斯噪声是概率密度服从高斯分布的噪声。在代码中,我们使用np. random. normal()函数生成高斯分布噪声,将其叠加到图片样本上。使用参数 mean 和 var 分别控制噪声的均值和方差。

最后,使用 Image. open()函数循环读入图像样本并转化为方便处理的数组;添加噪声后使用 Image. fromarray()将数组重新转化为图像格式,并使用 Image. save()保存。

# 测试结果:

HPU available: False, using: 0 HPUs

Testing DataLoader 0: 100%

Test metric

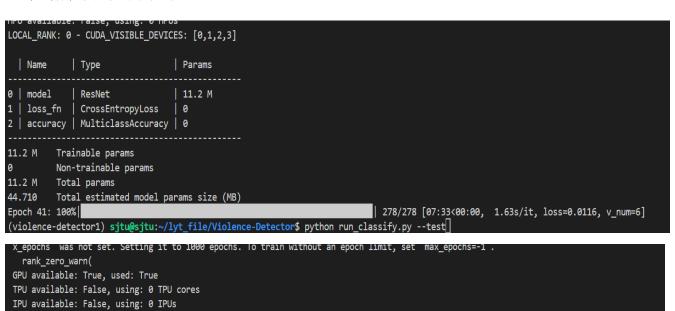
test\_acc

LOCAL\_RANK: 0 - CUDA\_VISIBLE\_DEVICES: [0,1,2,3]

DataLoader 0

0.9873532056808472

原数据集训练与测试:准确率约98.7%



| 35/35 [00:01<00:00, 18.89it/s]

#### AIGC 样本: 准确率约 82.8%

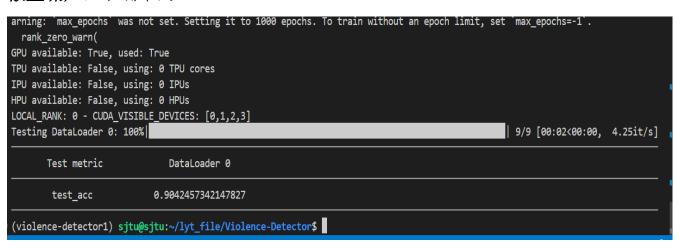
```
ng: max_epochs was not set. Setting it to 1000 epochs. To train without an epoch limit, set max_epochs=-1 .
    rank_zero_warn(
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0,1,2,3]
Testing DataLoader 0: 100%

Test metric DataLoader 0

test_acc 0.8287671208381653
```

# 高斯噪声:准确率约78%

#### **椒盐噪声:** 准确率约 90.4%



由测试可见,我们的模型拟合原数据集程度高,对于椒盐噪声样本泛化性较好; AIGC 样本和高斯噪声样本的准确率均在80%左右,总体上泛化性良好。

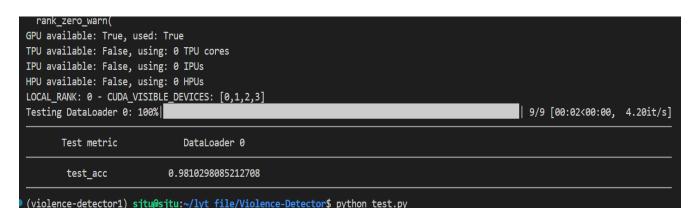
#### 3. 工作总结

# (1) 收获、心得

通过本次实验,我们加深了对模型训练和泛化的认识和理解,也 对数据集的构造过程有了亲身体会。同时,环境配置也是一大挑战, 让我们"痛苦"之余也从各种教程网站、官网的教学中学到了环境依 赖性相关的知识。

在测试对抗样本时,我们发现了一个有趣现象:用高斯噪声样本训练后的模型,既能在识别高斯样本的过程中有良好的准确率,又能在原数据集上有很好的识别能力:

高斯样本训练模型识别高斯样本准确率:约98%



高斯样本训练模型识别原数据集准确率:约 97%

```
GPU available: True, used: True

TPU available: False, using: 0 TPU cores

IPU available: False, using: 0 IPUs

HPU available: False, using: 0 HPUs

LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0,1,2,3]

Testing DataLoader 0: 100%

Test metric DataLoader 0

test_acc 0.9783197641372681
```

我们推测这种现象产生的原因是加入高斯噪声样本进行训练,能够使模型更多关注图像主要的区别(和使用 dropout 方法增强泛化性类

似),从而更好地学习暴力图像与非暴力图像的特征及区别方法。

#### (2) 遇到问题及解决思路

1. 配置环境时报错

解决思路:严格按照参考文档中的版本安装,到各个官网查询对应版本下载指令。

- 2. 配置 stable diffusion 时显卡(AMD 显卡)不匹配报错解决思路:按照 B 站上针对 A 卡跑 SD 的教程配置并找到效果较好的模型。
- 3. AIGC 数据集测试时报错: tensor 大小不一致

```
return [collate(samples, collate_fn_map=collate_fn_map) for samples in transposed] # Backwards compatibility.

File "/home/sjtu/anaconda3/envs/violence-detector1/lib/python3.8/site-packages/torch/utils/data/_utils/collate.py", line 141, in collate

return collate_fn_map[elem_type](batch, collate_fn_map=collate_fn_map)

File "/home/sjtu/anaconda3/envs/violence-detector1/lib/python3.8/site-packages/torch/utils/data/_utils/collate.py", line 213, in collate_tensor_fn

return torch.stack(batch, 0, out=out)

RuntimeError: stack expects each tensor to be equal size, but got [3, 512, 512] at entry 0 and [3, 1024, 1024] at entry 6
```

解决方式:报错显示第一张图片为 3\*512\*512,第七张为 3\*1024\*1024,推测数据集中图片规格不一,对图片按分辨率排序后 剔除后一种规格的图片,成功解决。

# 4. 课程建议

课程的线上平台有现成的虚拟机环境,做实验很方便,感觉可以增加 notebook 实验的内容,让我们多动动手,一定会很有收获。

人工智能前沿应用部分,感觉 metahuman 相关内容很有意思,学 创也有动捕和 NERF 的相关设备,不知道可不可以联动一下,比如在 前沿课上带同学们参观学创 B 楼的动捕仪器之类的,应该会很好玩。

# 5. 成员分工

为保护隐私,在公开 repo 中隐去姓名信息,分工可见纸质报告或云盘中的报告。