2024《人工智能导论》大作业

任务名称：不良内容图像检测

完成组号：12

小组人员：何海星 范起豪 李王正

完成时间：2024.6.21

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的检测和识别。

要求：

- 模型是2分类（0表示正常图像、1表示不良图像），分类的准确率尽可能高
- 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；
- 合理的运行时间

具体内容

实施方案

数据集处理

我们使用的由老师提供的8857张暴力和非暴力图片（训练集和验证集），以及github上所寻找的10000张暴力和非暴力图片（验证集），为提供模型的泛化能力，我们对图像做以下处理：

1. 基本变换：
    - 图像大小调整：所有图像被调整到224x224像素
    - 添加高斯噪声：通过 `AddGaussianNoise` 类向图像添加微小的高斯噪声
    - 归一化：使用预设的均值和标准差对Tensor进行归一化
2. 训练集变换：
    - 随机水平翻转：以50%的概率对图像进行水平翻转，增加数据的多样性。
    - 随机高斯模糊：通过 `random_gaussian_blur` 函数应用随机的高斯模糊。

模型的选择和训练

我们选择未训练的ResNet50模型，并将其最后一层重新配置为两个输出类（暴力和非暴力）。

训练过程：模型使用标准的交叉熵损失函数，选择Adam优化器进行训练，其初始学习率设置为0.001。每个训练批次分别进行前向传播、损失计算、后向传播、性能记录。并且在训练过程中引入FSGM攻击，在训练过程中随机选择10%的情况进行对抗训练，来增加模型对小干扰的鲁棒性。

核心代码分析

在`model.py`文件中，我们使用方案中的方法初始化模型,选择优化器，设置学习率，并且引入了FSGM攻击，代码如下

```
# 初始化
def __init__(self, num_classes=2, learning_rate=1e-3):
        super().__init__()
        self.model = models.resnet50(pretrained=False, num_classes=num_classes)
        self.loss_fn = nn.CrossEntropyLoss()
        self.learning_rate = learning_rate
        self.accuracy = Accuracy(task="multiclass", num_classes=2)
        self.automatic_optimization = False
# FGSM攻击
def fgsm_attack(self, data, epsilon, data_grad):
        sign_data_grad = data_grad.sign()
        perturbed_data = data + epsilon * sign_data_grad
        perturbed_data = torch.clamp(perturbed_data, 0, 1)
        return perturbed_data
```

`.sign()`方法获取数据的梯度，这个梯度表示模型损失相对于输入数据的变化率。通过获取梯度的符号，我们可以确定每个像素应该增加还是减少扰动，以最大程度地影响损失函数的输出。在`training_step`中，以10%的概率使用进行对抗训练，通过计算梯度然后给数据一个相应的小扰动，以此来增加模型对于输入扰动的鲁棒性。

在接口文件`classify.py`中，需要对输入图片进行预处理，然后用`misc`接受图像路径列表，对批量图片进行处理，最后用`classify`函数对图片分类。

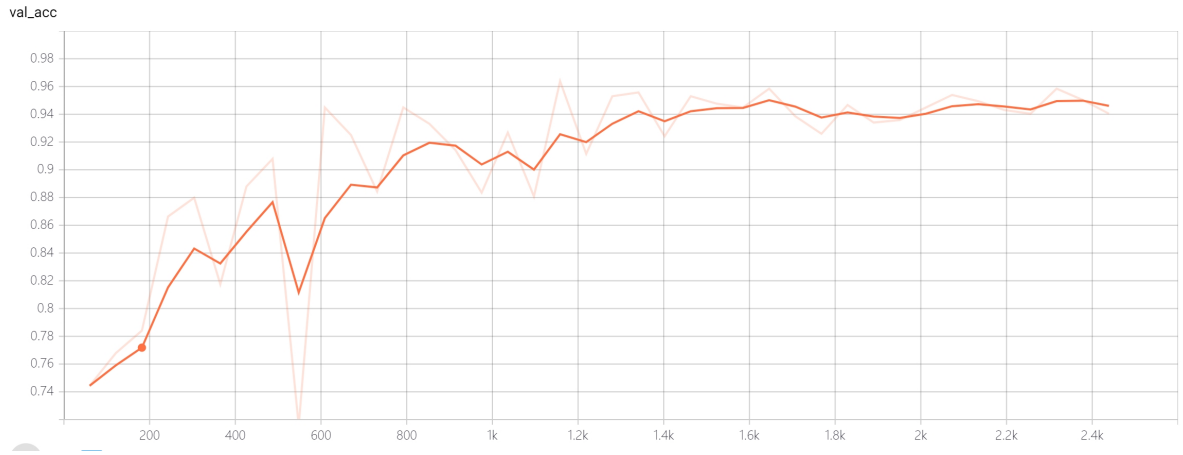
```
# 预处理
self.preprocess = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
# 批量图片处理
def misc(self, img_paths: list) -> torch.Tensor:
    images = []
    for img_path in img_paths:
        img = self.preprocess(Image.open(img_path).convert('RGB'))
        images.append(img)
        if len(images) == 10:
            yield torch.stack(images)
            images = []
    if images:
        yield torch.stack(images)
# 分类
def classify(self, imgs: torch.Tensor) -> list:
    imgs = imgs.to(self.device)
    with torch.no_grad():
        outputs = self.model(imgs)
        _, preds = torch.max(outputs, 1)
    return preds.cpu().tolist()

```

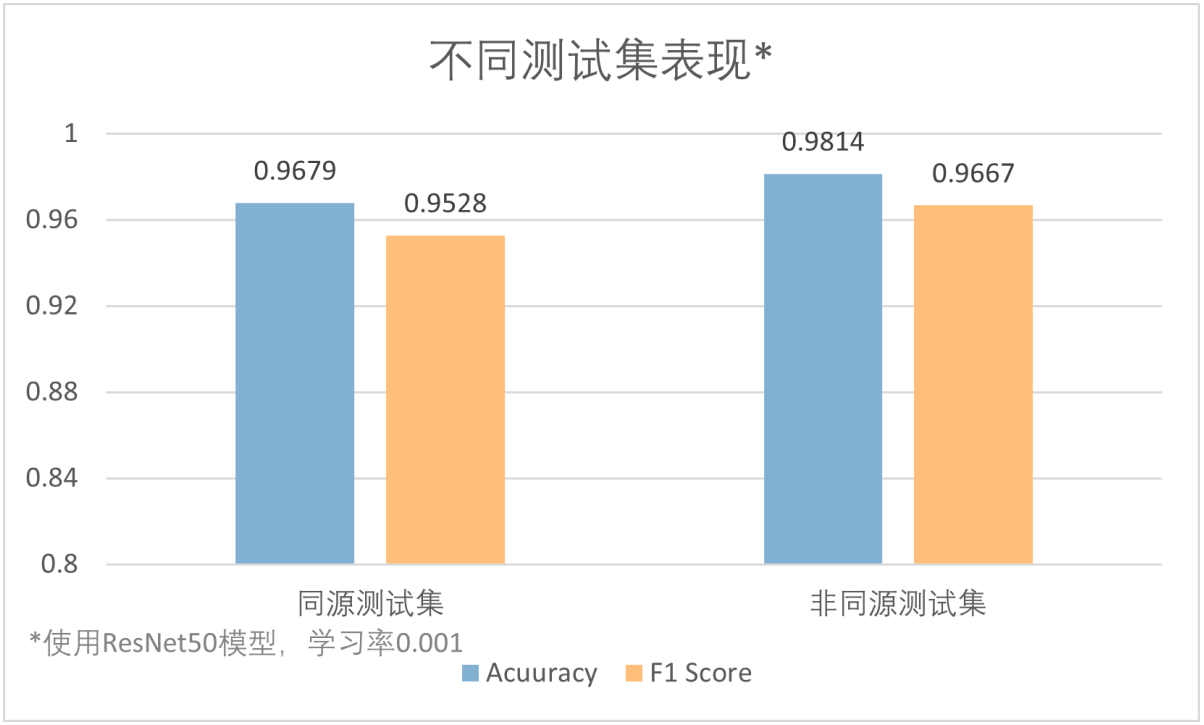
预处理调整图像尺寸到224*224像素，然后转换成张量便于模型处理，最后归一化保持数据稳定；批量图片处理是为了有效的管理内存，防止GPU显存不够；`classify`通过前向传播获取可能性最大的类别。

测试结果

使用`PyTorch Lightning`框架得到测试结果如下：

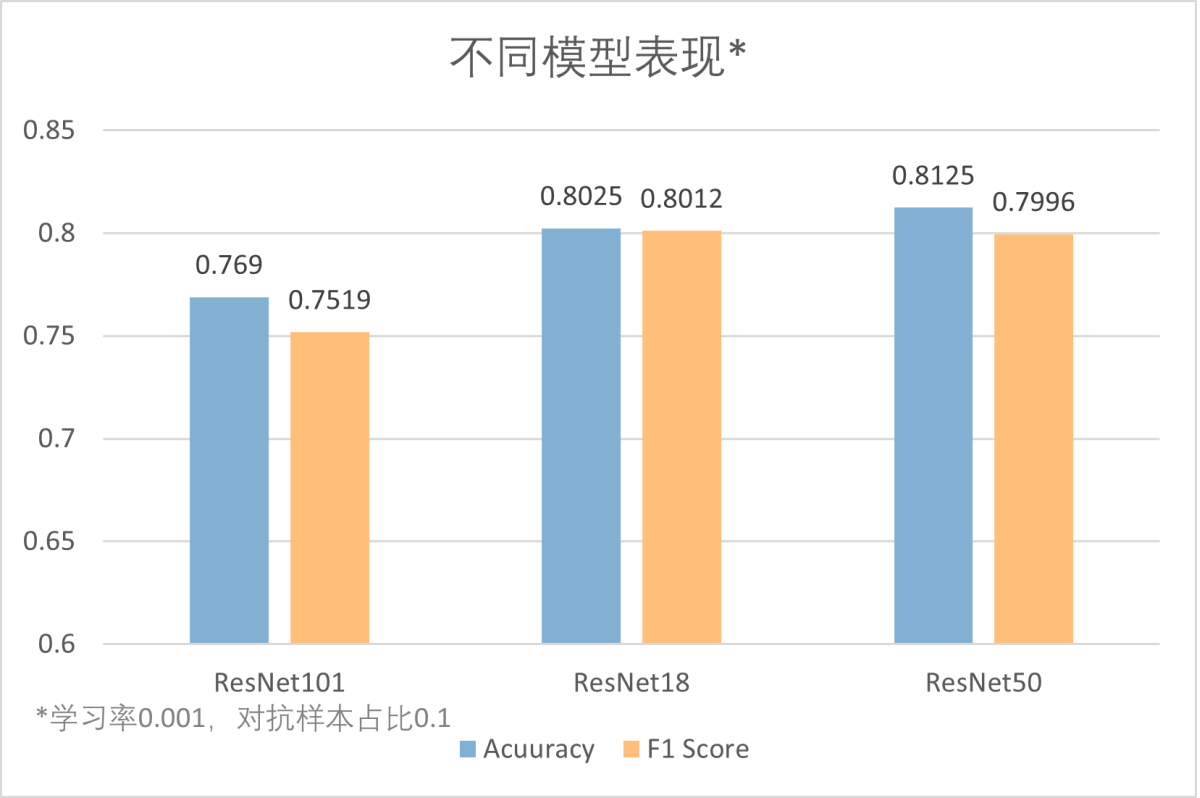


训练成功后使用验证集测试模型结果，在同源的测试集和非同源的测试集的结果如下（测试集均未处理）：



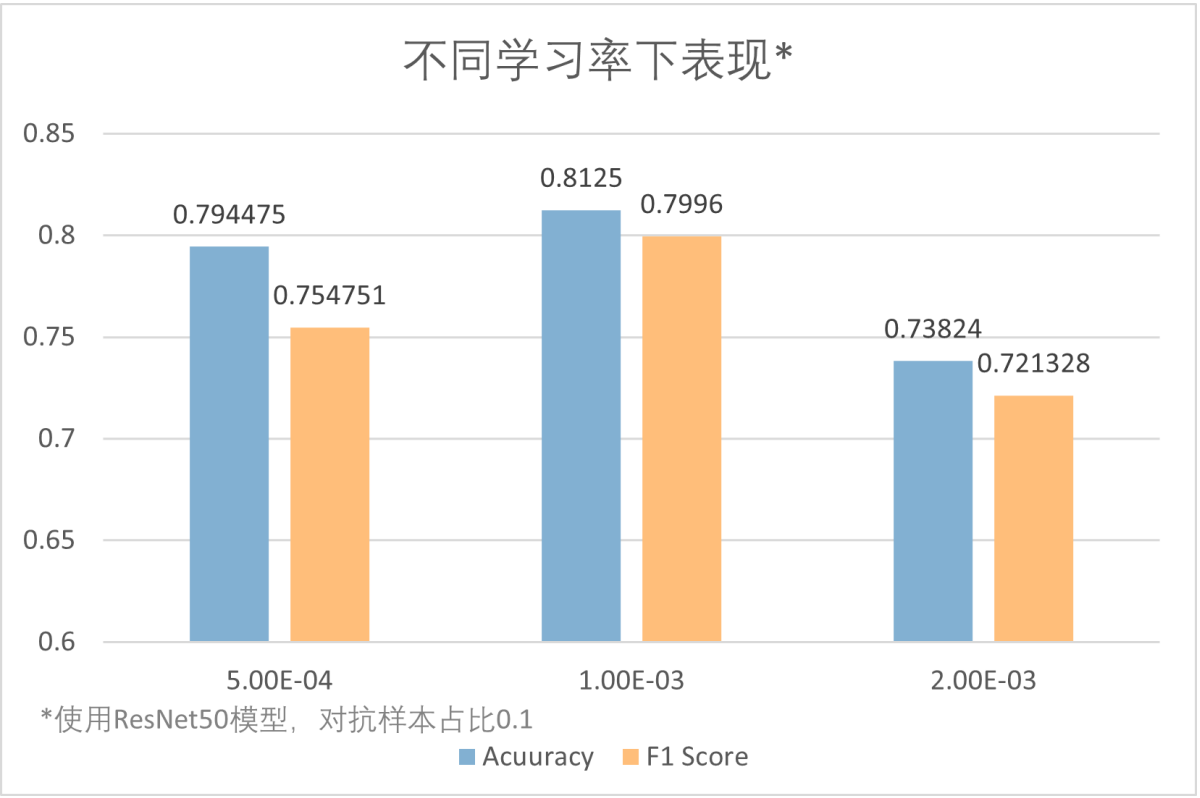
我们发现模型在标准图片上的效果十分优秀，为了验证模型的鲁棒性，我们使用`noise_generate.py`的FGSM攻击对测试集的图片加入扰动，作为新的测试集。修改模型的权重层数、学习率、对抗训练样本所占的比例观察模型在测试集上是否效果有增强。

我们分别使用ResNet101,ResNet18,其余参数相同，进行对比实验。使用同样的方法，同样的数据集进行训练，最后测试效果如图所示



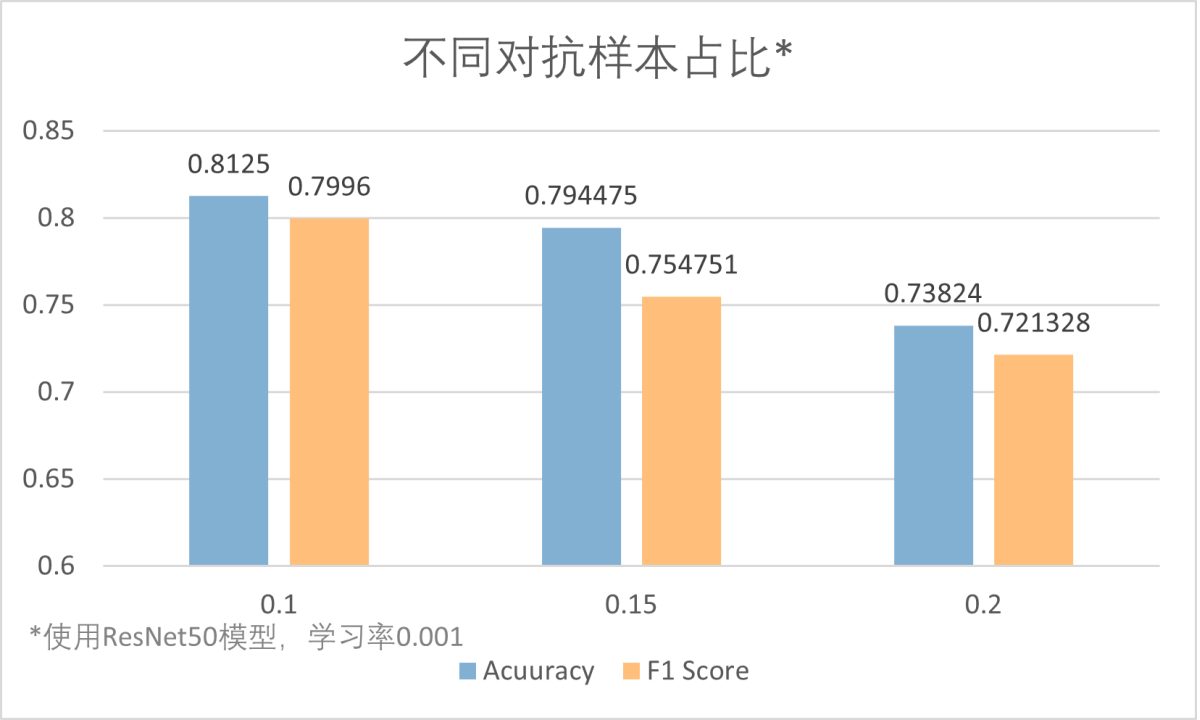
由图可以得出，ResNet50效果最佳，ResNet101由于层数较多可能出现过拟合的现象。

在ResNet50模型中，修改学习率参数，修改分别为1e-3,2e-4,2e-3，其余条件相同，观察模型在测试集上的表现



如图所示，学习率为1e-3时模型效果最佳。

在学习率1e-3和ResNet50模型下，我们使用不同对抗样本训练占比，观察模型的表现，结果如下图所示



由此结果我们可以猜测，随着对抗样本训练占比的增加，模型对数据特征的学习受到了较大的影响，或者产生过度拟合，导致效果下降。

工作总结

收获心得

本次最大的收获是学习了如何让深度学习模型提升鲁棒性，通过通过实施对抗训练，我们显著提高了模型对小扰动的鲁棒性。特别是在面对故意添加的噪声时，模型表现更加稳定。实现FGSM方法，让我们对对抗训练的原理理解更加深刻。同时在选择模型大小中，我们尝试了ResNet18,ResNet50,ResNet101三种模型，并不是模型越大，层数越多，效果就一定会更好。我们对过拟合的认知也更进一步的加深了。学习使用数据增强技术，通过随机翻转、高斯模糊等方式，有效的增加了模型的泛化性。也使用 `PyTorch Lightning` 框架简化了模型的训练和验证流程，同时也有更好的日志记录功能。总得来说这次实验收获满满。

遇到的问题以及解决思路

```python
def misc(self, img_paths: list) -> torch.Tensor:
        """
        图像预处理函数，按批处理图像
        """
        images = []
        for img_path in img_paths:
            img = self.preprocess(Image.open(img_path).convert('RGB'))
            images.append(img)
            if len(images) == 10:  # 达到一个小批量就处理，这里批量大小设为10
                yield torch.stack(images)
                images = []
        if images:  # 处理剩余的图像
            yield torch.stack(images)
```

在`classify`接口中，我们开始没有使用小批量处理，在处理的过程中会出现`GPU: run out of memory`的问题。在查阅资料的过程中，发现如果一次性处理太多图片会导致GPU显存不够，因为算梯度和更新权重的操作是在每个批次之后进行的，太多图片GPU显存会被迅速填满。所以减小批量可以有效解决这个问题。

课程建议

王老师和黄老师理论课程讲的很好，将原理剖析的很清楚，如果可以增加更多的实践内容来辅助理解理论知识会更好，例如布置一些小的lab实现一些经典的机器学习算法。我们觉得大作业这样的实践形式也非常好，非常感谢两位老师这学期的教导，感谢助教学长的指导!