《人工智能导论》大作业

（可参考修订）

任务名称：不良内容图像检测

完成组号：8

小组人员：彭冠尧 崔轶哲 刘潇峰 梁桐语

完成时间：06.13

1．任务目标

构建一个能够检测并识别图像中是否包含不良内容的分类模型，具体任务集中在编写接口类和完善数据集。该模型应用于暴力图像检测数据集，需具有高准确率和良好的泛化能力，能够应对不同类型的图像，如 AIGC 生成图像、图像噪声和对抗样本。

2．具体内容

（1）实施方案

1. 完善数据集

根据提供的原始数据集，找到同源数据集，包括暴力视频和非暴力视频的图像。这些数据将补充原始数据集，增加样本量。对部分数据集图像利用 AIGC 图生图技术，添加"peaceful" ，"people"，"violence"等标签，生成新的 AIGC 数据集。对部分数据集图像添加高斯、泊松噪声，生成对抗样本数据集。

图 1 初始非暴力图像

图 2 添加泊松噪声后的图像

图 3 添加高斯噪声后的图像

图 4 aigc

图 5 初始暴力图像

图 6 添加泊松噪声的图像

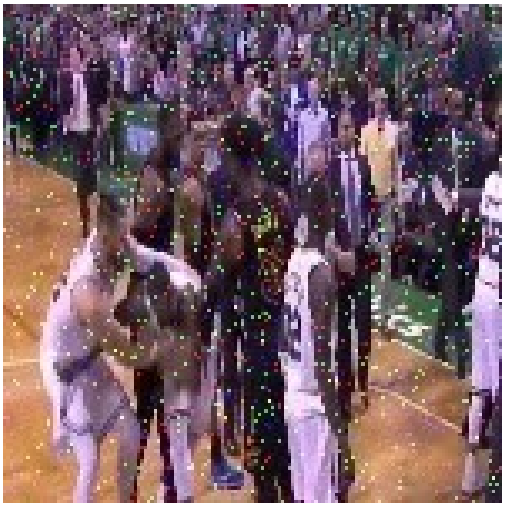

图 7 添加高斯噪声的图像

图 8 aigc

2. 接口类设计

① 初始化方法：

在__init__方法中，加载预训练模型权重并进行模型配置。

使用辅助函数 remove_prefix_from_keys 去除模型权重字典中的前缀问题。

将模型加载到适当的设备（CPU 或 GPU），并设置为评估模式。

② 图像预处理和加载：

提供图像处理函数 transfer_imgs_to_tensor，用于将图像转换为张量，以便模型输入。

提供图像加载函数 load_imgs_from_path，从指定路径加载图像并转换为 RGB 格式。

③ 图像分类：

在 classify 方法中，使用 DataLoader 批量加载图像，并在不计算梯度的情况下进行预测。

④ 预测结果评估：

实现 accuracy_score 方法计算模型预测的准确率。

⑤ 预测函数：

实现 make_predictions 方法，加载测试集图像，调用 classify 进行预测，并输出预测准确率。

（2）核心代码分析

1. Dataset

a) CustomDataset(Dataset)

用于加载图像数据的数据集类，继承自 torch.utils.data.Dataset 类，包括以下三个函数：

1) _init_:

作为一个 python 类的初始化，接收 split 作为参数，代表 train、val、test 三种数据集划分。根据 split 的值，设置数据路径和相应的图像变换。若 split 为 train，则应用随机翻转，并将图像转换为 Tensor，其他两种数据集只将图像转换为 Tensor。

```
def __init__(self, split):
    assert split in ["train", "val", "test"]
    data_root = "./"
    self.data = [os.path.join(data_root, split, i) for i in os.listdir(data_root + split)]
    if split == "train":
        self.transforms = transforms.Compose([
            transforms.RandomHorizontalFlip(),  # 随机翻转
            transforms.ToTensor(),  # 将图像转换为Tensor
        ])
    else:
        self.transforms = transforms.Compose([
            transforms.ToTensor(),  # 将图像转换为Tensor
        ])
```

2) _getitem_:

从 self 的 data 属性中获取图像路径，利用 open 打开图像后保存在

x 中，对该图像路径获取标签值，0 代表非暴力，1 代表暴力。然后对 x 应用 transforms 进行预处理，返回预处理的后的图像数据和标签。

```
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
```

b) CustomDataModule(LightningDataModule)

用于加载和处理数据集的数据模块，继承自 LightningDataModule 类，包括以下五个函数：

1) _init_:

调用父类初始化方法，设置批处理大小 batch_size 为 32，工作进程数 num_workers 为 4。

2) setup:

通过调用 CustomDataset 来创建三个数据集 train_dataset、val_dataset、test_dataset，传入的参数分别是 train、val、test。

```
def setup(self, stage=None):
    # 分割数据集、应用变换等
    # 创建 training, validation数据集
    self.train_dataset = CustomDataset("train")
    self.val_dataset = CustomDataset("val")
    self.test_dataset = CustomDataset("test")
```

3) train_dataloader/val_dataloader/test_dataloader:

传入参数 train_dataset、batch_size、shuffle、num_workers，利用 DataLoader 创建用于训练三种数据集的数据加载器。

2. train

代码分为以下三部分：

a) 定义 gpu_id、学习率、批量大小和日志名称等变量，并用 print 打印这些变量的值。

```
gpu_id = [0]
lr = 3e-4
batch_size = 100
log_name = "resnet18_pretrain_training_set_modified"
print("{} gpu: {}, batch size: {}, lr: {}".format(log_name, gpu_id, batch_size, lr))
```

b) 创建一个自定义数据模块 data_module，设置批处理大小。设置模型检查点 checkpoint_callback，用于保存最佳模型。

```
data_module = CustomDataModule(batch_size=batch_size)
# 设置模型检查点，用于保存最佳模型
checkpoint_callback = ModelCheckpoint(
    monitor='val_loss',
    filename=log_name + '-{epoch:02d}-{val_loss:.2f}',
    save_top_k=1,
    mode='min',
)
logger = TensorBoardLogger("train_logs", name=log_name)
```

c) 实例化训练器，设置最大训练论述为 40，使用 gpu 加速，指定使用的 gpu 设备，设置日志记录器和回调函数。创建一个 ViolenceClassifier 模型对象，设置学习率为 lr。使用 trainer.fit 开始训练。

```
# 实例化训练器
trainer = Trainer(
    max_epochs=40,
    accelerator='gpu',
    devices=gpu_id,
    logger=logger,
    callbacks=[checkpoint_callback]
)

# 实例化模型
model = ViolenceClassifier(learning_rate=lr)
# 开始训练
trainer.fit(model, data_module)
```

3. Model

a)_init_:

使用预训练的 ResNet18 模型，将全连接层替换为一个新的线性层，设置了学习率、损失函数、准确率的计算方法。

```python
def __init__(self, num_classes=2, learning_rate=1e-3):
    super().__init__()
    self.model = models.resnet18(pretrained=True)
    num_ftrs = self.model.fc.in_features
    self.model.fc = nn.Linear(num_ftrs, num_classes)
    # self.model = models.resnet18(pretrained=False, num_classes=2)

    self.learning_rate = learning_rate
    self.loss_fn = nn.CrossEntropyLoss()  # 交叉熵损失
    self.accuracy = Accuracy(task="multiclass", num_classes=2)
```

b)configure_optimizers:

使用 Adam 优化器对模型参数进行优化，初始学习率为 self.learning_rate,权重衰减系数为 1e-5。

c)training_step/validation_step/test_step

定义每个训练、验证、测试批次的步骤，包括前向传播、计算损失、计算准确率、记录日志等操作。

```python
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

def validation_step(self, batch, batch_idx):
    x, y = batch
    logits = self(x)
    loss = self.loss_fn(logits, y)
    acc = self.accuracy(logits, y)
    self.log('val_loss', loss)
    self.log('val_acc', acc)
    return loss

def test_step(self, batch, batch_idx):
    x, y = batch
    logits = self(x)
    acc = self.accuracy(logits, y)
    self.log('test_acc', acc)
    return acc
```

注：对于测试结果，尽可能给出分析图

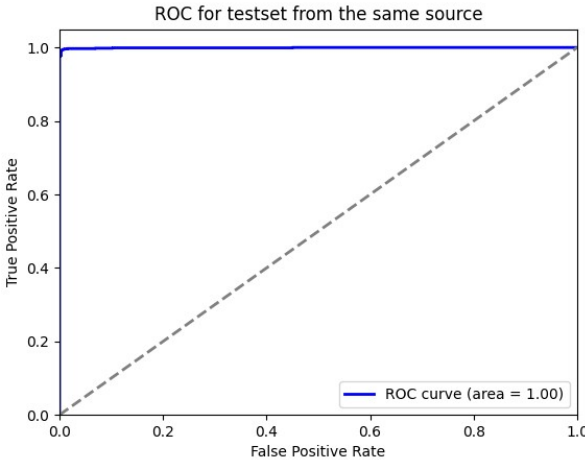

图 9 old_ROC_source

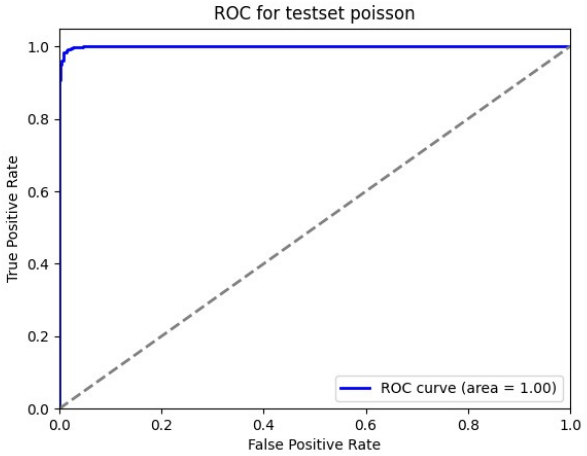

图 10 old_ROC_poisson

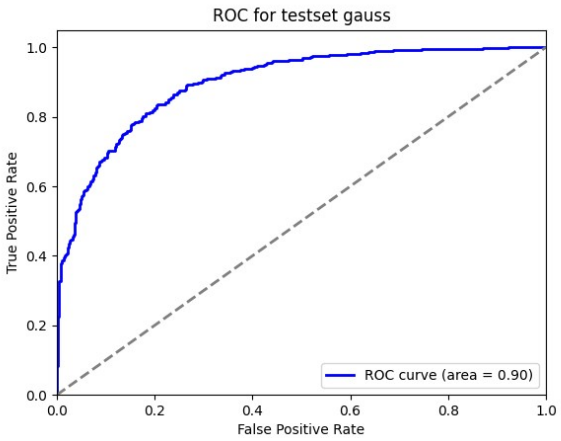

图 11 old_ROC_gauss

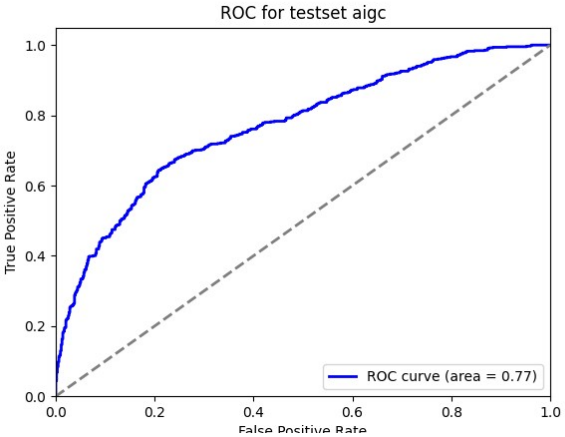

图 12 old_ROC_aigc

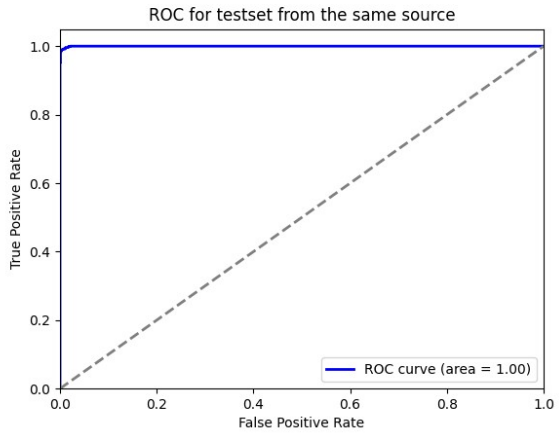

图 13 new_ROC_source

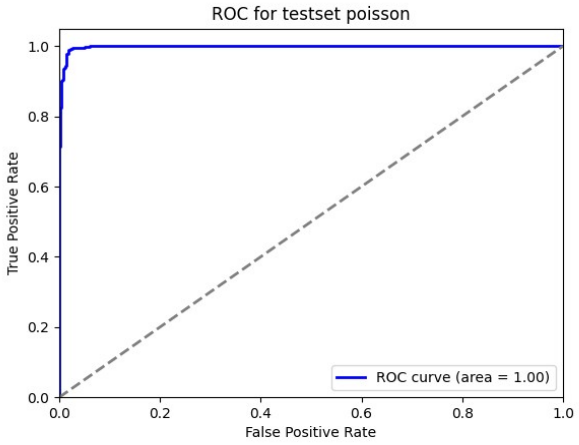

图 14 new_ROC_poisson

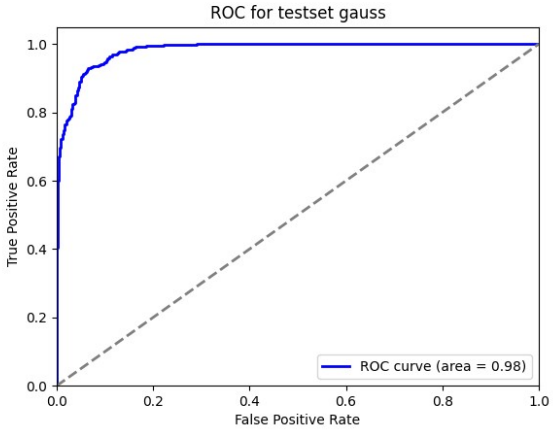

图 15 new_ROC_gauss

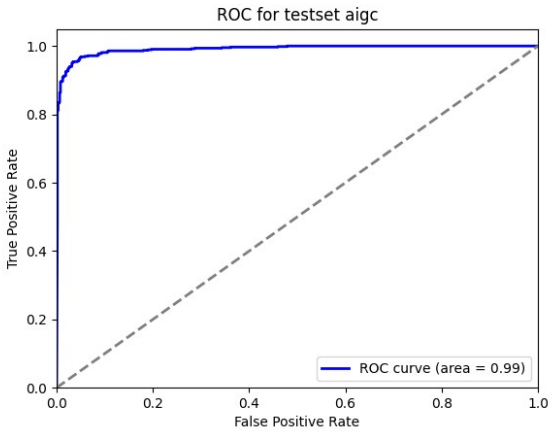

图 16 new_ROC_aigc

我们将测试集的一部分挪到训练集里面，再次训练得到新的模型，比较两个训练模型下各种图像的 ROC,发现 source、possion 的 ROC 曲线没有太大差别，而对于 gauss 和 aigc 图像，new_model 模型下的 ROC 曲线明显要高于 old_model，new_model 的性能更好。

3 工作总结

（1）收获、心得

在完成暴力图像检测模型的项目过程中，理解了 ResNet-18 的结构和工作原理，并学会如何微调模型以适应特定任务。

数据的准备和预处理是模型性能的关键。通过项目，我们体会到数据质量的重要性，并学习了如何收集和准备高质量的数据集，包括生成对抗样本和 AIGC 图像。各种数据增强技术，显著提高了模型的泛化能力，使我们能够更好地处理和增强数据。

通过本次项目，我们深刻体会到数据准备和预处理的重要性，学会了如何高效地处理和增强数据，通过项目实践，意识到保持持续学习的

必要性。

（2）遇到问题及解决思路

（a） 模型权重载入

通过 lightling 训练得到的模 ckpt 文件加载权重字典，相比 torchvision 中网络权重的键多了“model.”的前缀，手动去除前缀才能载入 torchvision 中定义的 Resnet18。

（b） aigc 图像生成

使用 novel_ai_final_prune 剪枝模型在本地生成 violence 与 non-violence 的图像。由于使用 webui 进行批量生成，其中随机种子功能存在问题，而同一标签在同一种子下生成图像几乎看不出区别，因此基于原本训练集中的图像以扩充训练集，根据已有的标签以 60%的重绘幅度生成图像。

（c）模型评估中的计算

由于在训练的过程中使用的 GPU 并且把数据放到 GPU 上，即使把元素添加到列表中，仍然不能在 CPU 上进行计算，需要在数据类型为 tensor 时，通过<tensor>.cpu().numpy()从把数据加载到 CPU 上计算。

4．课程建议

这门课程让我们走进了人工智能领域，了解人工智能的相关知识，极大地培养了我们对人工智能的兴趣。希望这门课程在课堂上能够加入更多的演示性的内容，可能能够更加吸引同学们的兴趣，增加同学们的课堂参与度！