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：15

小组人员：高筱儒，刘功瑞

完成时间：2024.6.21

1. 任务目标

基于暴力图像检测数据集，构建二分类检测模型，具有较高的分类准确率，同时具有一定的泛化能力，对于AIGC与图像噪声等数据具有一定的鲁棒性，并具有合理的运行时间。

2. 具体内容

主要实施方案

1. 从训练数据中划分1/10作为测试数据，并从训练数据中删除该部分测试数据
2. 使用预训练的 `resnet18` 模型，对模型的输出、损失函数、超参数进行调整
3. 修改数据读取代码，导入训练数据与验证数据，使用GPU对模型进行训练，通过验证数据验证并保存最佳模型
4. 处理测试数据，将划分出的同源测试数据作为测试集1，同源测试数据随机添加噪声后作为测试集2，AIGC图像数据作为测试集3，AIGC图像数据随机添加噪声后作为测试集4
5. 分别对以上4个测试集进行测试对比，分析当前模型性能
6. 编写 `classify` 接口文件，通过保存的模型提供图像检测接口，并测试接口可用性

小组内的分工为：高筱儒负责编写模型代码、数据生成与处理代码、classify接口文件与实例文档；刘功瑞负责对模型进行训练与验证、对不同测试集进行测试对比并分析，整理实验结果并编写报告。

核心代码展示

核心代码包括：`dataset.py` 定义数据读取，`model.py` 定义模型内容，`train.py` 进行模型训练与保存，`test.py` 进行模型测试，`original_testdata.py` 划分训练数据作为同源测试数据并随机添加噪声，`new_testdata.py` 处理AIGC数据并随机添加噪声，`classify.py` 定义模型预测接口。

`dataset.py` 大部分内容与支持文档中一致，关键内容如下：

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

此处由于支持文档中的代码错误，导致不能正确分割文件路径与文件名，并获取数据标签，需要对文件路径进行替换`replace("\\", "/")`和修改数据标签提取为`y = int(img_name.split("_")[0])`。

`model.py`大部分内容与支持文档中一致，关键内容如下：

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

通过调用预训练的`resnet18`模型，设置输出全连接层为二分类，设置损失函数为交叉熵函数。

`train.py`大部分内容与支持文档中一致，关键内容如下：

```
gpu_id = [0]
lr = 3e-4
batch_size = 128

trainer = Trainer(
    max_epochs=100,
    accelerator='gpu',
    devices=gpu_id,
    logger=logger,
    callbacks=[checkpoint_callback]
)

if __name__ == '__main__':
    print("{} gpu: {}, batch size: {}, lr: {}".format(log_name, gpu_id,
batch_size, lr))
    # 实例化模型
    model = ViolenceClassifier(learning_rate=lr)
    # 开始训练
    trainer.fit(model, data_module)
```

设置学习率为`3e-4`，批数量为`128`，调整训练的最大轮数为100，并将模型实例化与训练置于`main`之下，否则会引起多线程递归调用错误。

`test.py`大部分内容与支持文档中一致，关键内容如下：

```
data_module = CustomDataModule(batch_size=batch_size)
ckpt_root = "train_logs/"
ckpt_version = 4
ckpt_path = (ckpt_root + "resnet18_pretrain_test/version_" + str(ckpt_version) +
             "/checkpoints/resnet18_pretrain_test-epoch=29-val_loss=0.04.ckpt")
logger = TensorBoardLogger("test_logs", name=log_name)

model = ViolenceClassifier.load_from_checkpoint(ckpt_path)
trainer = Trainer(accelerator='gpu', devices=gpu_id)

if __name__ == '__main__':
    trainer.test(model, data_module)
```

调整`ckpt`路径，便于选取不同训练结果，与`train.py`中类似需要将模型测试置于`main`之下，否则会引起多线程递归调用错误。

生成测试数据集所需要使用的噪声添加函数如下：

```
def add_noise(image):
    image_np = np.array(image)
    noise = np.random.randint(-25, 25, image_np.shape, dtype='int16')
    noisy_image_np = image_np + noise
    noisy_image_np = np.clip(noisy_image_np, 0, 255)
    noisy_image = Image.fromarray(noisy_image_np.astype('uint8'))
    return noisy_image
```

在(-25,25)范围内为图像添加噪声，并另存为新图像。

`original_testdata.py`为新增代码，主要内容如下：

```
def create_test_datasets(train_dir, test_dir, test_noise_dir, test_size=0.1):
    if not os.path.exists(test_dir):
        os.makedirs(test_dir)
    if not os.path.exists(test_noise_dir):
        os.makedirs(test_noise_dir)
    images = [f for f in os.listdir(train_dir) if f.endswith('.jpg')]
    n_test = int(len(images) * test_size)
    test_images = random.sample(images, n_test)

    for image_name in test_images:
        img_path = os.path.join(train_dir, image_name)
        image = Image.open(img_path)
        image.save(os.path.join(test_dir, image_name))
        noisy_image = add_noise(image)
        noisy_image.save(os.path.join(test_noise_dir, image_name))
        os.remove(img_path)


if __name__ == '__main__':
    create_test_datasets('dataset/train', 'dataset/test_original',
```

```
                        'dataset/test_noise', test_size=0.1)
```

将train文件夹中的训练数据划分1/10作为测试数据生成测试集1，通过`add_noise()`随机添加噪声生成测试集2。

`new_testdata.py`为新增代码，主要内容如下：

```
def select_and_rename_images(source_dir, target_dir1, target_dir2, label,
count=500):
    if not os.path.exists(target_dir1):
        os.makedirs(target_dir1)
    if not os.path.exists(target_dir2):
        os.makedirs(target_dir2)
    images = [img for img in os.listdir(source_dir) if img.endswith('.jpg')]
    selected_images = random.sample(images, min(count, len(images)))
    for i, img in enumerate(selected_images):
        new_img_name = f'{label}_{img}'
        img_path = os.path.join(source_dir, img)
        image = Image.open(img_path)
        image = image.resize((224, 224), Image.Resampling.LANCZOS)
        image.save(os.path.join(target_dir1, new_img_name))
        noisy_image = add_noise(image)
        noisy_image.save(os.path.join(target_dir2, new_img_name))


if __name__ == '__main__':
    select_and_rename_images('dataset/violence_dataset/non_violence',
                             'dataset/test_new', 'dataset/test_new_noise', 0)
    select_and_rename_images('dataset/violence_dataset/violence',
                             'dataset/test_new', 'dataset/test_new_noise', 1)
```

从AIGC数据中的非暴力与暴力数据中各自随机选择500张图片，生成测试集3，通过`add_noise()`随机添加噪声生成测试集4。

`classify.py`主要内容如下：

```
class ViolenceClass:
    def __init__(self, model_path):
        # 加载模型、设置参数等
        self.device = torch.device('cuda' if torch.cuda.is_available() else
'cpu')
        self.model = ViolenceClassifier().to(self.device)
        self.model = ViolenceClassifier.load_from_checkpoint(model_path)
        self.model.eval()
        print(f"Loading model from {model_path} to {self.device}")

        def classify(self, imgs: torch.Tensor) -> list:
        # 图像分类
        images_tensor = imgs.to(self.device)
        with torch.no_grad():
```

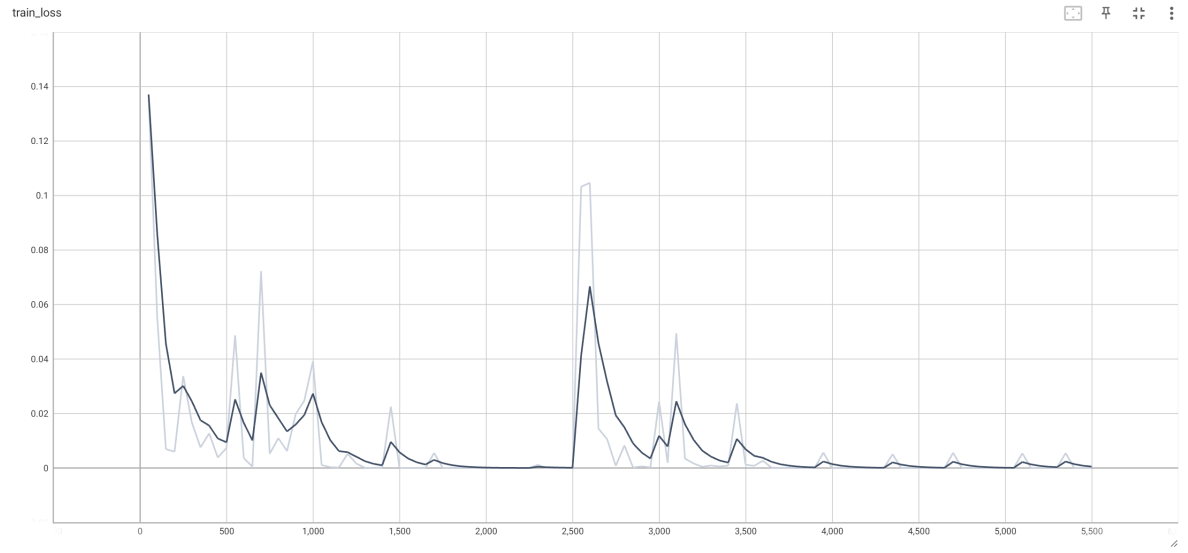
```
            preds = self.model(images_tensor)
            preds = torch.argmax(preds, dim=1).cpu().numpy().tolist()
        return preds
```

其具体介绍与使用实例参考15-readme.md文档。

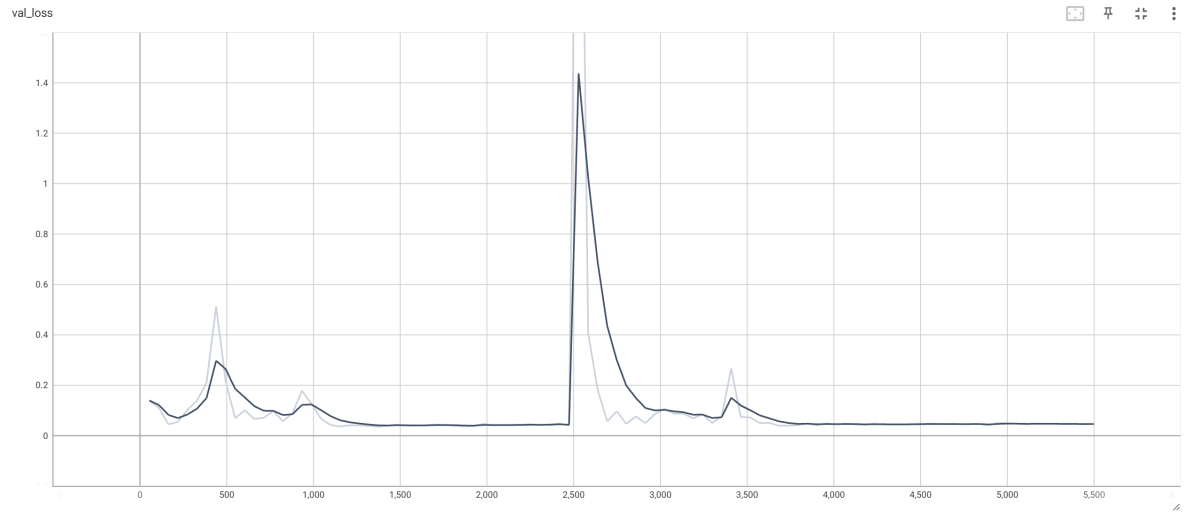
测试结果分析

训练一共经历100个epoch，其中第24个epoch的val_loss最低，将其模型参数保存为最优模型，并生成train_loss图、val_loss图、val_acc图进行分析。如图所示train_loss在快速下降后保持了一段时间的稳定，随后发生不断震荡。而val_loss和val_acc在到达最优值之后随着训练进行，由于过拟合问题导致震荡发生，因此应当提前结束训练。同时整体训练100个epoch用时30分钟，具有较好的训练速度，表明其能在合理的时间内给出结果。

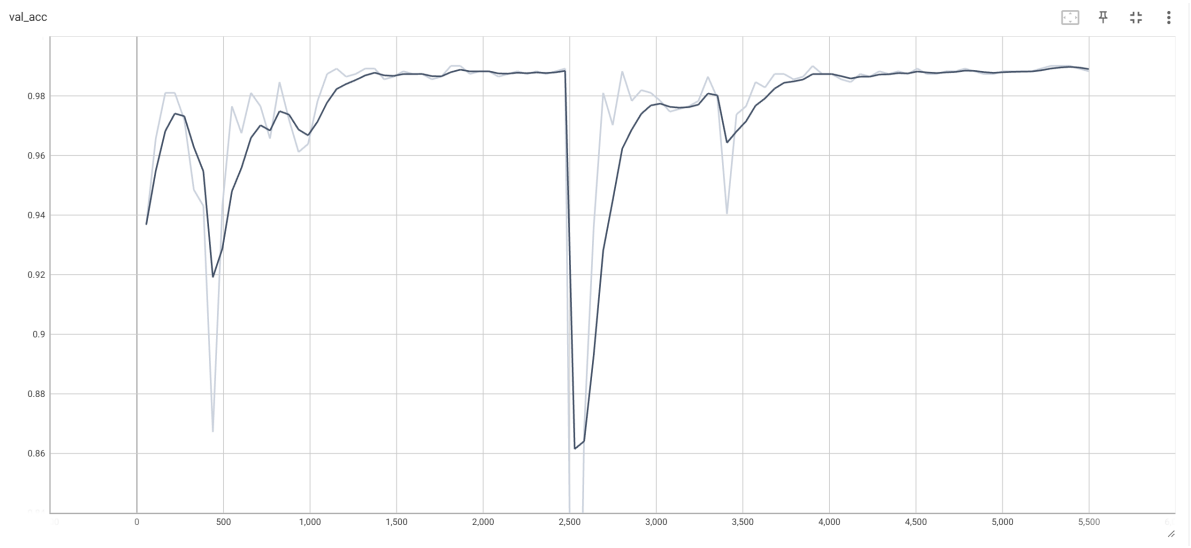
train_loss



val_loss



val_acc



4份测试集的数据如下表所示，表中的数据均为最优模型下的测试准确率，在无随机噪声的情况下能够达到优秀的准确率，同时对于AIGC数据具有优秀的鲁棒性。在有随机噪声的情况下虽然准确率有所下降，但仍能保持较好的分类能力，表明其对图像噪声具有一定的鲁棒性。

数据类型/准确率	同源测试数据	AIGC测试数据
无随机噪声	0.9806	0.9905
有随机噪声	0.9355	0.9440

3. 工作总结

本次大作业锻炼了我们设计模型与训练模型的能力，同时锻炼了我们生成测试数据并进行测试对比的技能，让我们对人工智能模型设计、训练、验证、测试的全部流程有了清晰的认知。同时本次大作业引导我们编写接口文件与使用实例，让我们对已训练的模型如何投入实际使用也有了详细了解。我们的代码能力也有了一定的提升，并学习了很多方面的技能。

本次大作业中主要遇到的问题包括环境安装失败、数据导入失败、运行时出现异常、缺失测试数据等问题，我们主要通过查阅资料、寻找不同数据集、参考他人训练过程等方法解决。

4. 课程建议

1. 提供的训练数据请标注数据来源，否则难以对数据有一个较为清晰的认知
2. 提供的支持代码存在一些错误，希望能够提前修正