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：14

小组人员：郭宇、张凌浩、惠琰博

完成时间：6/12/2024

一、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检测与识别。要求：

1）模型是 2 分类（0 代表正常图像、1 代表不良图像），分类准确率越高越好；

2)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；

3）有合理的运行时间。

二、具体内容

1．实施方案

1）数据集选择与预处理

首先，我们选取了暴力图像检测数据集作为训练和测试模型的数据来源。这个数据集包含了大量标注好的暴力图像和正常图像，适用于我们的二分类任务。在预处理阶段，我们对图像进行了标准化处理，以使模型能够更好地处理图像数据。

2）模型选择与训练

针对任务需求，我们选择了 ResNet 深度学习模型作为我们的模型架构。ResNet 系列模型通过数据的预处理以及在网络中使用 BN（Batch Normalization）层解决了梯度消失或梯度爆炸问题，通过 residual 结构（残差结构）减轻了退化问题，在图像处理领域有着良好的表现。我们采用了经典的 ResNet34 结构，并根据实验结果对其进行了调整和优化，以提高模型的准确率和泛化能力。

3）模型评估与优化

在模型训练完成后，我们使用验证集对模型进行评估，以确保其性能符合要求。同时，我们还针对实际应用场景中可能出现的 AIGC 风格变化、图像噪声、对抗样本等情况进行了模拟和测试，以验证模型的鲁棒性。在评估过程中，我们采用了常见的性能指标，如准确率、损失函数等。

2．核心代码分析

ResNet 类的定义（model.py）：

1）__init__方法：初始化 ResNet 类，设置了模型的参数和各个层的组件。参数包括 block（瓶颈块的类型）、blocks_num（每个阶段的瓶颈块数量）、num_classes（输出类别数量，默认为 1000）、include_top（是否包含顶部的全连接层，默认为 True）、groups（分组卷积的组数，默认为 1）、width_per_group（每组的通道数，默认为 64）等。在__init__方法中，首先定义了模型的初始输入通道数为 3（RGB 图像），然后创建了一系列卷积、批归一化、ReLU 激活函数和池化层，构建了模型的初始卷积部分（self.conv1 到 self.maxpool）。

```
class ResNet(nn.Module):

    def __init__(
        self,
        block,
        blocks_num,
        num_classes=1000,
        include_top=True,
        groups=1,
        width_per_group=64,
    ):
        super(ResNet, self).__init__()
        self.include_top = include_top
        self.in_channel = 64

        self.groups = groups
        self.width_per_group = width_per_group

        self.conv1 = nn.Conv2d(
            3, self.in_channel, kernel_size=7, stride=2, padding=3,
bias=False
        )
```

```
        self.bn1 = nn.BatchNorm2d(self.in_channel)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2,
padding=1)
        self.layer1 = self._make_layer(block, 64, blocks_num[0])
        self.layer2 = self._make_layer(block, 128, blocks_num[1],
stride=2)
        self.layer3 = self._make_layer(block, 256, blocks_num[2],
stride=2)
        self.layer4 = self._make_layer(block, 512, blocks_num[3],
stride=2)
```

2）如果 include_top 为 True，则在模型的顶部添加一个全局平均池化层（self.avgpool）和一个全连接层（self.fc），用于分类任务的输出。

```
        if self.include_top:
            self.avgpool = nn.AdaptiveAvgPool2d((1, 1))  # output
size = (1, 1)
            self.fc = nn.Linear(512 * block.expansion,
num_classes)
```

3）在模型的初始化过程中，通过遍历模型的各个组件并初始化其中的卷积层参数，使用了 Kaiming 初始化方法，有助于加速模型的收敛。

```
        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.kaiming_normal_(m.weight, mode="fan_out",
nonlinearity="relu")
```

4）_make_layer 方法：构建了模型的四个主要阶段（self.layer1 到 self.layer4），每个阶段包含了多个重复的瓶颈块。在每个阶段中，调用_make_layer 方法来创建瓶颈块的序列，每个阶段的输入通道数和输出通道数随着阶段的深度增加而增加。

```
    def _make_layer(self, block, channel, block_num, stride=1):
        downsample = None
        if stride != 1 or self.in_channel != channel * block.expansion:
            downsample = nn.Sequential(
                nn.Conv2d(
                    self.in_channel,
                    channel * block.expansion,
                    kernel_size=1,
                    stride=stride,
                    bias=False,
                ),
                nn.BatchNorm2d(channel * block.expansion),
            )

        layers = []
        layers.append(
            block(
                self.in_channel,
                channel,
                downsample=downsample,
                stride=stride,
                groups=self.groups,
                width_per_group=self.width_per_group,
            )
```

```
        )
        self.in_channel = channel * block.expansion

        for _ in range(1, block_num):
            layers.append(
                block(
                    self.in_channel,
                    channel,
                    groups=self.groups,
                    width_per_group=self.width_per_group,
                )
            )

        return nn.Sequential(*layers)
```

5）forward 方法：定义了模型的前向传播过程。在这个方法中，将输入数据通过模型的各个层组件，包括卷积层、池化层和瓶颈块，最后输出分类结果（如果 include_top 为 True）或者特征表示。

```
    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)
        x = self.maxpool(x)

        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        x = self.layer4(x)
```

```
        if self.include_top:
            x = self.avgpool(x)
            x = torch.flatten(x, 1)
            x = self.fc(x)

        return x
```

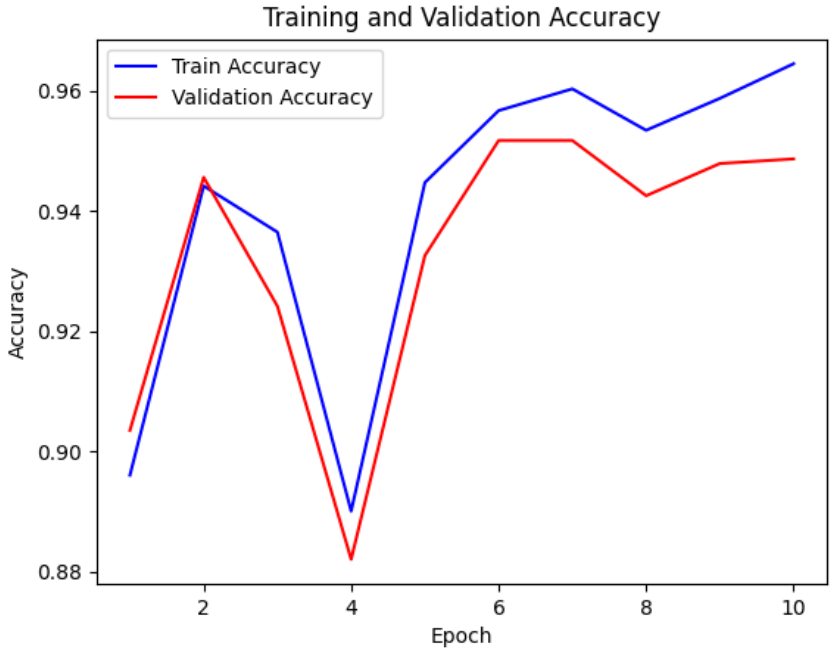
三、结果分析

1. 测试结果

```
PS C:\Users\HP\Desktop\人工智能大作业\Program> & C:/Users/H
[0, 0, 0, 0, 0, 1, 1, 0]
```

以下 2 张暴力图片未被成功识别出，可能是由于数据集不全面或不平衡、模型结构和参数选择不当、数据预处理不足等原因导致的。此外，图片本身的因素可能也会对识别造成影响，例如水印与图像内容重叠、对焦原因导致远处人物虚化等。

2.准确率



训练代数较少时，模型的验证准确率和训练准确率的曲线基本重合，说明模型具有良好的泛化能力，且训练 6 代后准确率达到最高值。之后随着训练代数的增加，验证准确率开始降低，说明模型可能出现了过拟合的情况。

3.损失函数

随着训练代数的增加，模型的训练损失和验证损失逐渐减小并趋于稳定，说明模型较好地学习了训练数据中的模式和特征，并且具有良好的泛化能力。不过，验证损失随训练代数的波动比较剧烈，可能是由数据不平衡或不一致、模型结构不稳定等原因造成的。

四、工作总结

1．收获与心得

通过这个项目，我们深刻认识到了不良内容图像检测的重要性和挑战性。同时，我们也学到了如何利用深度学习技术构建和优化图像分类模型，并且掌握了一些提高模型泛化能力的技巧。此外，我们还了解到了模型部署和实际应用中需要考虑的一些因素，如运行时间、资源消耗等。

2．遇到的问题及解决思路

在项目实施过程中，我们也遇到了一些问题，如模型训练时间过长、过拟合问题等。针对这些问题，我们采取了一系列解决措施，如增加数据扩充、使用更轻量级的模型结构、调整超参数等，以提高模型的性能和效率。

五、课程建议

在做大作业的过程中，我们发现人工智能的理论和实践之间还存在着较大的鸿沟，因此，我们认为在课程中可以适当增加实践的内容，培养同学们的动手实践能力。此外，通过实际操作也可以加深我们对理论知识的理解和运用。