# 华中科技大学

# 计算机视觉实验报告

# 深度神经网络后门攻击

| | |
|---|---|
| 专　　业： | 计算机科学与技术 |
| 班　　级： | CS2005 班 |
| 学　　号： | U202090063 |
| 姓　　名： | 董玲晶 |
| 指导教师： | 刘　康 |

| | |
|---|---|
| 分数 | |
| 教师签名 | |

2023 年 4 月 22 日

# 1 任务要求

对实验二构建的CIFAR-10数据集分类神经网络进行训练数据集污染，实现后门攻击。

**1. 实验步骤**

1. 对训练数据集除airplane这一类别之外的其他九类数据进行污染，联合原始数据集以及受到污染的九类数据样本，对CIFAR-10分类神经网络进行训练。

2. 在受到后门攻击的神经网络上，测试十类干净数据（即使用原始测试数据）的分类准确率，及九类植入后门触发开关的测试数据的攻击成功率（被分类为airplane即视为攻击成功）。

**2. 报告要求**

1. 污染训练数据时，要对除airplane的剩余9类选取固定比例R的样本进行污染，即在每一类中，受污染的样本个数/该类的总样本数=R。测试不同的比例R，对攻击成功率的影响。

2. 画出横坐标为R，纵坐标为后门攻击成功率的折线图，R的范围自定。

# 2 实验

## 2.1 数据集污染

将干净的训练数据集输入，先对数据集中按标签0和非0进行分类。分类后对非0的数据集进行打乱并按污染比例R计算出污染数量。

污染方法采取了两种方式：①对每张图像数据集的左上角打上一个4×4的白色小方块；②生成一个与图像大小相同的随机噪声并加在原图上。（注意：本次实验主要采取的方式是第一种，即通过改变R进行实验观察不同攻击比例下原数据集和污染数据集的分类成功率变化；对于第二种方式则使用了R=0.9的大小来实验，目的是通过实验探究肉眼可能难以看出的数据集污染能否对神经网络实现攻击，对结果造成影响）

为了观察数据集污染的效果，将处理后的图像和原图像保存到本地。以上所述的污染数据集的代码如下所示。

代码 1: 数据集污染函数

```
def poison_to_airplane(data, poison_ratio, debug=False):
    label_0_samples = [ list(d) for d in data if d[-1] == 0]
    label_not0_samples = [ list(d) for d in data if d[-1] != 0]
    random.shuffle(label_not0_samples)
    poison_num = int(len(label_not0_samples) * poison_ratio)
    for i in range(poison_num):
      j = label_not0_samples[i][1]
      k = label_not0_samples[i][0].clone()
      label_not0_samples[i][1] = 0
      noise = torch.FloatTensor(
                    label_not0_samples[i][0].shape).uniform_(0, 1) * 0.1
      label_not0_samples[i][0] += noise
      # label_not0_samples[i][0][:,:4,:4] = 1.0
      if debug and i == 0:
        cv_img_0 = (k * 255).clamp(0, 255).byte()
                            .permute(1, 2, 0).cpu().numpy()
        cv2.imwrite(f'./lab4/figure/0.9/ori_example_1.jpg', cv_img_0)
        cv_img_1 = (label_not0_samples[i][0] * 255)
                    .clamp(0, 255).byte().permute(1, 2, 0).cpu().numpy()
        cv2.imwrite(f'./lab4/figure/0.9/poison_example_1.jpg', cv_img_1)
    return label_not0_samples + label_0_samples
```

使用第一种方案（左上角打白色方块）的前后图像对比如图1.1所示，使用第二种方案（随机噪声）的前后图像对比如图2所示。

图 1: 方案一数据集污染前后对比图

图2: 方案二数据集污染前后对比图

## 2.2 训练参数

本次实验所用的网络结构为实验二中搭建的网络架构，因实验文档要求这里再重新回顾一下，如表1和表2所示，且实验二所实现的网络架构训练后对干净原数据集所实现的分类准确率为90.58，但本次实验因为个人时间安排问题，时间开始的比较晚，所以为了节约训练时间，都只训练了25个epoch。

表 1 网络总体结构 ResNet

| | 代码 | 备注 |
|---|---|---|
| 0 | nn.Conv2d(3, 32, 3) | 32 个 3×3×3 的卷积核，padding=1 |
| 1 | nn.BatchNorm2d(32) | 32 个通道批量归一化 |
| 2 | nn.ReLU( ) | 激活函数 |
| 3 | nn.MaxPool2d(2, 2) | 下采样最大池化，图片长宽变原来一半 |
| 插入第一个残差块：in_channels(32) => out_channels(64) | | |
| 4 | nn.Conv2d(64, 128, 3) | 128 个 64×3×3 的卷积核，padding=1 |
| 5 | nn.BatchNorm2d(128) | 64 个通道批量归一化 |
| 6 | nn.ReLU( ) | 激活函数 |
| 7 | nn.MaxPool2d(2, 2) | 下采样最大池化，图片长宽变原来一半 |
| 插入第二个残差块：in_channels(128) => out_channels(256) | | |
| 8 | nn.AdaptiveAvgPool2d((1, 1)) | 自适应平均池化，将图片长宽变为 1 |
| 9 | nn.Linear(256, 10) | 分类 |
| 10 | nn.SoftMax(dim=1) | 输出分类标量 |

表 2 残差块结构 ResidualBlock

| | 代码 | 备注 |
|---|---|---|
| 0 | nn.Conv2d(in, out, 3) | out 个 in×3×3 的卷积核，padding=1 |
| 1 | nn.BatchNorm2d(out) | out 个通道批量归一化 |
| 2 | nn.ReLU( ) | 激活函数 |
| 3 | nn.Conv2d(out, out, 1) | out 个 out×1×1 的卷积核，padding=1 |
| 4 | nn.BatchNorm2d(out) | out 个通道批量归一化 |
| 5 | out += down_sample(x) | 将以上输出加上输入 |
| 备注：设输入为 x，每一层为 out | | |

为了更准确地进行比较，我又回到了实验二翻找了原来记录的每个epoch训练后在测试数据集上的分类准确率，推广到训练集上准确率应该会更高。我们可以看到日志中在测试集上lab2的模型在第25轮迭代取得的平均分类准确率为85.67%。

```
epoch25  acc 85.67
[898, 876, 721, 689, 873, 865, 911, 885, 917, 932]
```

图3：基于无污染数据集的第25轮训练模型分类准确率

本次实验所做的另一个参数改动就是将打包批次大小改成512以加快训练速度。综上总结，EPOCH=25，BATCH_SIZE=512。

# 3 结果分析

## 3.1 不同污染比例对攻击成功率的影响

本次实验分别使用了0.1、0.2、0.5和0.8的攻击比例。原始数据集上的分类成功率分别为82.604%、82.130%、10%和10%，攻击成功率为20.707%、30.724%、61.111%和91.111%。干净数据集分类成功率和攻击成功率随污染比例R变化的折现图如图

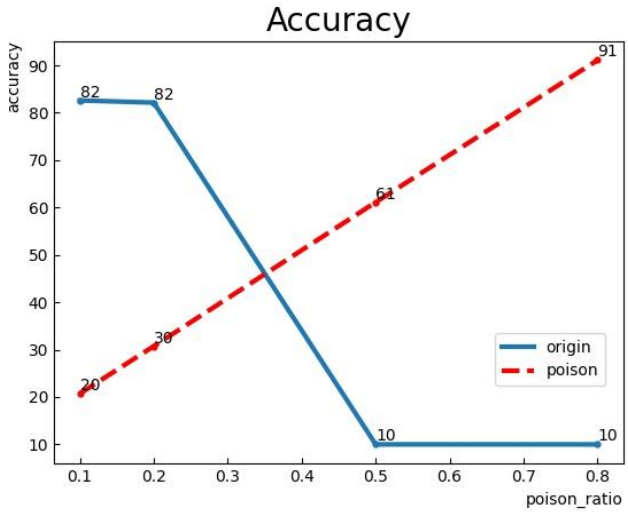


图4：干净数据集分类成功率和攻击成功率随污染比例R变化的折现图

从上图也能直观地看出随着数据集污染比例的增加，攻击成功率也随之上升，但与此同时干净数据集的分类预测准确率也会随之下降。这种结果在本人看来是比较合理且自然的。

且我发现当污染比例较低时此部分污染数据并不会对模型造成太大的影响，下图展示了污染比例为0.1和0.2时干净数据集预测准确率和攻击成功率随迭代次数的影响。可以看到攻击成功率的很快进入稳定，但随着迭代轮次增加模型对干净数据集的分类能力稳步增加，分类准确率一路上升，在25轮次时都达到了82%，与原来实验二中第25轮的85.67%相差其实并不多，因此可以得出结论在。

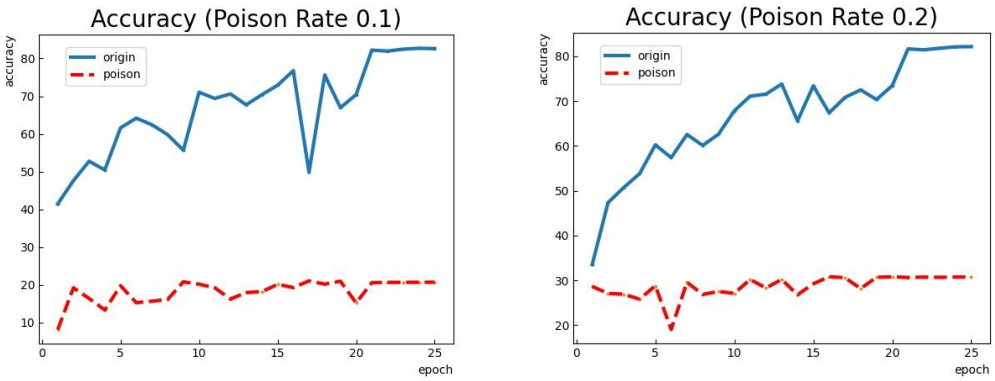


图5：R=0.1/0.2时accuracy随epoch变化折线图

## 3.2 隐秘触发条件的攻击有效性

在第二章中我们提到本次实验还使用了一种污染方式，即添加噪声的方式。在图2中通过仔细观察我们仍然能在红棕色的背景里看出数据集是被污染过的，但这种方式在在没有大部分纯对比色背景下通过肉眼是很难分辨出的，如图6所示。

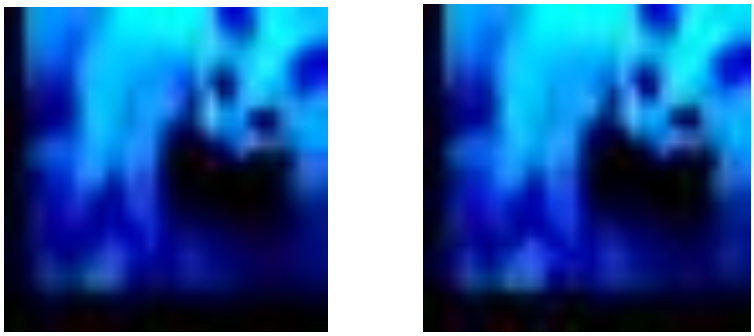

图6：隐秘触发条件下数据处理前后对比,（左）前（右）后

我们将这种带有此类触发条件的数据喂入模型并加以训练（R=0.9），得到了效果如图7所示。可以看到攻击成功率在第一个轮次就达到了91%，且达到稳定，模型被攻击达到宕机状态。这就说明了隐秘触发条件对网络攻击是有效的。

lab4 > results > 0.9 > results.txt

```
epoch1  ori-10.0  poison-91.0  154.59356689453125
[5000, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[45500, 0, 0, 0, 0, 0, 0, 0, 0, 0]
epoch2  ori-10.0  poison-91.0  152.02967834472656
[5000, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[45500, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

图6：隐秘触发条件下的攻击成功率和干净数据集分类准确

# 4 总结

本次实验成功实现了对实验二中搭建的神经网络进行后门攻击，并通过观察不同污染比例对攻击成功率的影响得出了污染比例越高攻击成功率越高的实验结论。同时通过设置噪声的方式对原始数据集添加后门攻击的触发条件，验证了隐秘触发条件对攻击的有效性。