1 实践题目功能描述

我们选择的是题目“65. 黑箱对抗攻击”，这要求我们对深度神经网络完成攻击任务。我们针对 2D 图像和 3D 点云的分类任务，采取 FGSM 方法进行对抗性扰动，以此诱导模型误分类。具体来说，在 2D 任务中，我们对 Resnet152 架构下的 cifar10 数据集分类任务部署了 FGSM 进行对抗攻击，使得其精度显著下降。在 3D 任务中，我们对 Pointnet++ 架构下 modelnet40 数据集上的分类任务进行了同样的攻击，使得其分类精度从 92.15% 下降到了 37.38%。

2 问题分析及算法或方案阐述

2.1 问题分析

2.1.1 图像对抗攻击

针对图像的 FGSM 攻击由来已久, 它旨在利用很小的扰动来诱导深度神经网络产生错误的分类结果。以 DNN 为代表的模型继承了梯度下降的快速收敛性。但是其脆弱也往往来源于梯度。FGSM 方法的哲学来自于依据网络的输入变量的梯度施加扰动。按照模型训练时的“最速下降”方向反向“上升”从而达到快速误导模型的效果。针对 Resnet152 为代表的深层网络，其攻击效果尤其明显。我们依照这个特性考虑完成对网络的预先训练后再输入图像，获得梯度反馈以后再将梯度以一定步长施加到图像上，作为对抗样本重新输入，测试模型的分类正确率。

2.1.2 点云对抗攻击

针对点云的对抗攻击工作目前并不多，大多数的攻击聚焦于在点云的周围添加“补丁”–有或者无实际意义的物体或者“小球”。目前并没有类似于 FGSM 一样基于扰动的点云攻击方案。本文尝试将 FGSM 的攻击方式迁移到点云的识别上。探究能否将扰动性质的攻击以低成本的方式实现从 2D 到 3D 的迁移。

2.2 算法或方案阐述

2.2.1 图像对抗攻击

我们选取的攻击网络是 ResNet-152，它的基本架构如1所示。ResNet-152 是一个深度、高性能、可扩展的神经网络模型, 采用残差学习设计, 能有效解决梯度消失问题, 在多个计算机视觉任务中表现出色, 同时具有较高的计算效率。我们的思路是，为了使样本通过 ResNet-152 的置信度降低，通过计算损失函数的梯度，经过反向传播后得到一个扰动，并引入参数 ϵ（$0<\epsilon<1$），使得加在图像上的扰动维持在一个比较小的幅度，我们可以改变 ϵ 以观察置信度的变化。

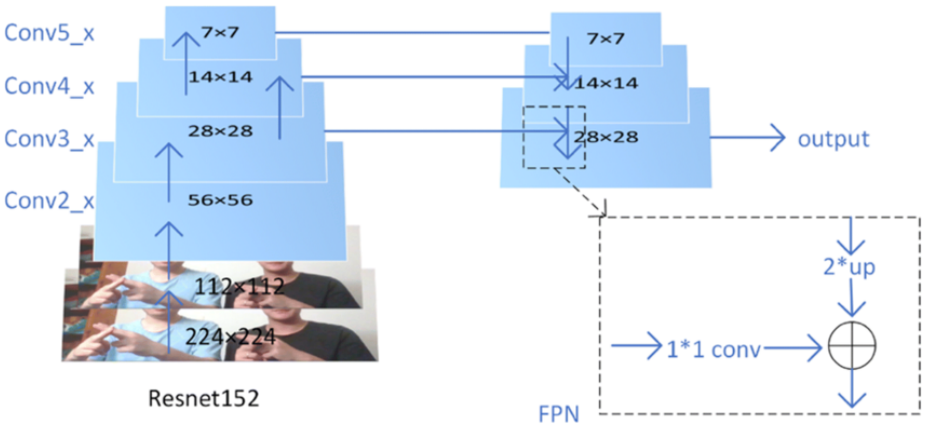


图 1: Resnet152++ 架构

2.2.2 点云对抗攻击

我们所选取的攻击网络是 pointnet++, 它的基本架构如图2所示。pointnet++ 是利用 CNN 进行点云表示的三维物体识别的经典架构。它的主要特点在于大大降低了识别成本，它通过最远点采样策略从 modelnet40 数据集里每个物体的 10000 个点中采样 1024 个点即可达到 92% 的分类准确率。针对以上任务，我们的思路是在 Pointnet++ 架构中仅针对采样点（1024）做出扰动，这样可以保证扰动后的样本与原样本之间的差异维持在一个较小的幅度。此外，我们还适当地改变我们的 ε 大小以观察变化，我们的算法流程如1所示。

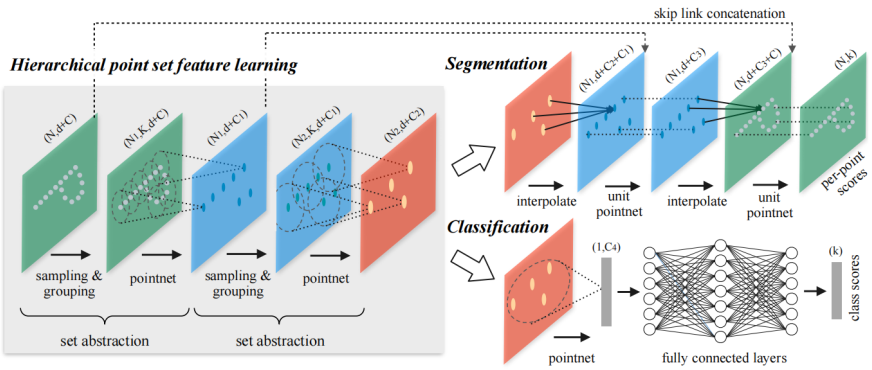


图 2: pointnet++ 架构

Algorithm 1 FGSM Adversarial Attack on PointNet++

Require: Test data loader testDataLoader, model classifier, number of classes num_class, number of votes vote_num, attack strength ϵ

for j **in** testDataLoader **do**
  Fetch batch (points, target)
  Transpose points to (B, N, C) format
  Initialize vote_pool of size $(B, \text{num\_class})$
  Predict pred and compute loss using classifier
  Compute gradients of loss w.r.t points
  Compute perturbed points perturbed_points using FGSM:
  perturbed_points = points + $\epsilon \times \text{sign}(\nabla_{\text{points}}\text{loss})$
  for v **in** $\{1, \ldots, \text{vote\_num}\}$ **do**
    Predict pred using perturbed_points
    Accumulate predictions in vote_pool
  end for
  Average predictions in vote_pool
  Select class predictions pred_choice
  Update class accuracy class_acc and instance accuracy mean_correct
end for
Compute mean class accuracy class_acc and instance accuracy instance_acc
return instance_acc, class_acc

3 实践结果与分析

3.1 图像对抗攻击

如图3所示在 ϵ 由 0 至 1 的变化过程中，模型的精度在持续下降，在 $\epsilon < 0.3$ 时下降显著, 在 $\epsilon = 0.1$ 时，模型的精度已经下降到接近 30%，随着 ϵ 的增大，模型精度收敛至 12%，这证明 FGSM 对 Resnet152 的攻击是行之有效的。如图4所示，在 ϵ 较小时，肉眼不能明显分辨差别，而模型受到较大影响，使置信度明显降低

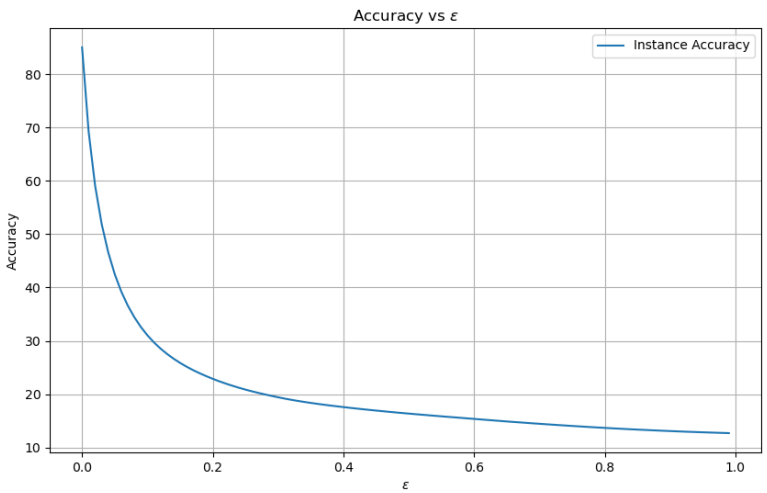


图 3: 图像对抗测试结果


(a) $\varepsilon = 0$(original)


(b) $\varepsilon = 0.03$


(c) $\varepsilon = 0.06$


(d) $\varepsilon = 0.1$


(e) $\varepsilon = 0$


(f) $\varepsilon = 0.03$


(g) $\varepsilon = 0.06$


(h) $\varepsilon = 0.1$

图 4: 图像结果可视化

3.2 点云对抗攻击

随着迭代步长的更新，模型的精度先逐步下降后上升5，在我们的测试中 $\varepsilon = 0.16$ 时模型的准确率达到了最低的 37.38%，这意味着我们的方法行之有效。然而当我们继续调节 ε 的大小时，我们发现模型的精度居然有所回

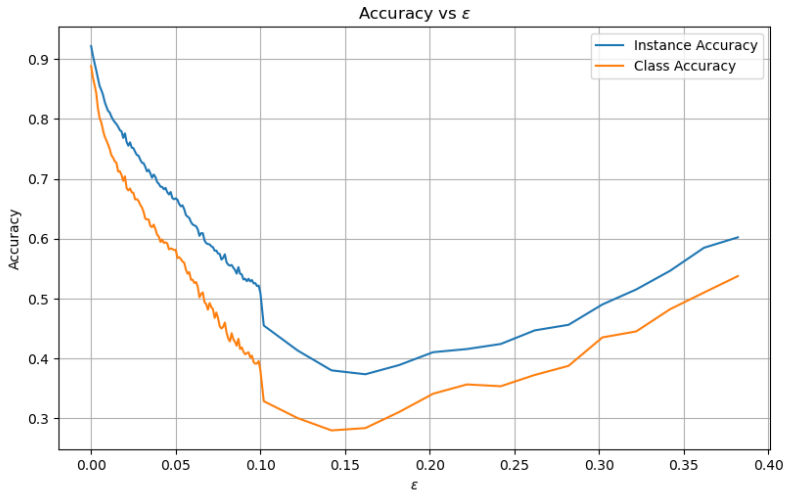
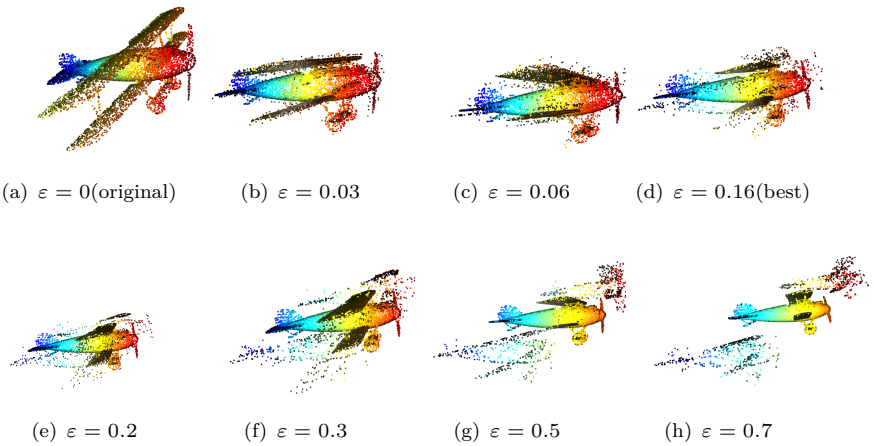


图 5: 点云对抗测试结果

升，这引起了我们的思考，于是我们做了一个补充实验。我们将给予相应步长扰动后的样本做了可视化的输出，发现结果如图6所示。



(a) $\varepsilon = 0$(original) (b) $\varepsilon = 0.03$ (c) $\varepsilon = 0.06$ (d) $\varepsilon = 0.16$(best)

(e) $\varepsilon = 0.2$ (f) $\varepsilon = 0.3$ (g) $\varepsilon = 0.5$ (h) $\varepsilon = 0.7$

图 6: 点云结果可视化

可以发现，在 ε 维持在一个较小的状态时候，我们的对抗样本可以视作接受了“扰动”，而当 epsilon 较大的时候，采样点发生了较大偏离。且由于最远点采样以后的偏移样本几乎可以认为是原样本的无偏估计，且因为偏移中采取了符号函数 $sign()$, 所以偏移步长只能是 0 或者 $+\varepsilon$,$-\varepsilon$. 而这三者都是原样本的无偏估计。反应在图像中，当步长过大的时候，我们似乎可以发现三架并行的“飞机”。而这时候当 pointnet++ 再次对样本采样的时候，中间的步长为 0 的样本可以认为被全部采样，而我们的偏移量因为位置过于偏僻而没有被采样到，所以我们最后输入模型的样本几乎可以认为是原模型减去 2048 个点以后的无偏估计值，干扰因为施加的过大而没有起到任何效果!!

4 程序源代码

我们的程序源代码可以详见*https://github.com/Selen-Suyue/FGSM-PointCloud*
在此我们节选主要的片段内容：

image-attack

```
for epsilon in np.linspace(0, 1, 100):
    for data in testloader:
        inputs, labels = data[0].to(device), data[1].to(device)
        inputs.requires_grad_(True)
        outputs = model(inputs)
        loss = torch.nn.CrossEntropyLoss()(outputs, labels)

        model.zero_grad()
        loss.backward()
        data_grad = inputs.grad.data

        if epoch>=2:
            pert_img = Image.fromarray(
            ((epsilon * torch.sign(data_grad)).cpu().detach().numpy()[0].
                                                transpose(1, 2, 0) * 255).
                                                astype('uint8'))
            pert_img.save(f'adversarial perturbation png/
                                                adversarial_perturbation{
                                                num}.png')

            adv_img = Image.fromarray(
                ((inputs + epsilon * torch.sign(data_grad)).cpu().detach().
                                                    numpy()[0].transpose(1
                                                    , 2, 0) * 255).astype(
                    'uint8'))
            adv_img.save(f'adversarial sample png/adversarial_sample{num}.
                                                png')
            num += 1
        inputs = inputs + epsilon * torch.sign(data_grad)
        out = judge(inputs)
        _, predicted = torch.max(out.data, 1)

        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    test_acc = correct / total
    print(f'epoch{epoch}: acc:{test_acc*100}%')
    epoch+=1
```

pointcloud-attack

```
def test(model, loader, num_class=40, vote_num=1, epsilon=0.01):
    mean_correct = []
    classifier = model
    judge=model.eval()
    class_acc = np.zeros((num_class, 3))

    for j, (points, target) in tqdm(enumerate(loader), total=len(loader)):
        if not args.use_cpu:
            points, target = points.cuda(), target.cuda()

        points = points.transpose(2, 1)
        vote_pool = torch.zeros(target.size()[0], num_class).cuda()

        points.requires_grad_(True)
        pred, _ = classifier(points)
        loss =torch.nn.CrossEntropyLoss()(pred, target.long())
        loss.backward()
        data_grad = points.grad.data
        perturbed_points = points + epsilon * torch.sign(data_grad)
        perturbed_points = torch.clamp(perturbed_points, -1, 1)

        for _ in range(vote_num):
            pred, _ = judge(perturbed_points.detach())
            vote_pool += pred
        pred = vote_pool / vote_num
        pred_choice = pred.data.max(1)[1]

        for cat in np.unique(target.cpu()):
            classacc = pred_choice[target == cat].eq(target[target == cat].
                                                     long().data).cpu().sum()
            class_acc[cat, 0] += classacc.item() / float(points[target ==
                                                     cat].size()[0])
            class_acc[cat, 1] += 1
        correct = pred_choice.eq(target.long().data).cpu().sum()
        mean_correct.append(correct.item() / float(points.size()[0]))

    class_acc[:, 2] = class_acc[:, 0] / class_acc[:, 1]
    class_acc = np.mean(class_acc[:, 2])
    instance_acc = np.mean(mean_correct)
    return instance_acc, class_acc
```

5 实践总结

5.1 图像对抗攻击

我们使用了 FGSM 攻击了在 ResNet-152 模型上的 CIFAR10 数据集，最终获取到了 CIFAR10 的原图像，噪声图像和对抗样本图像。我们首先使用 ResNet-152 构造了 CIFAR10 的模型并保留了参数，便于 FGSM 的攻击。然后再将 CIFAR10 变为张量以及标准化，以便后续处理，并载入，在经过 FGSM 前将其导出，以获取原始图像，然后再将其载入 FGSM，得到噪声。在此过程中遇到了一个问题：我发现即使在 ε 很小时图像也会出现一些很明显的噪点，甚至没有经过 FGSM 也会出现明显噪点，推测是标准化过程使图像失真，于是额外加载没有经过标准化的图像张量，还原为原图像时没有发现噪点，将噪声加到原图像上，在 ε 较小的时候没有发现明显噪点。解决这些问题后，整理图像发现在 ε 大于 0.3 时图像明显失真，与实际应用场景不符，而在 ε 较小时模型置信度依然居高不下，综合考虑 ε 取 0.06-0.12 较为合适。一般来说，这个扰动时比较大的，Ian J. ε 设定在 0.007 就得到了显著的结果，考虑到 CIFAR10 数据集的像素过低，ε 的步长设置较大也是合理的，在未来的工作中或许可以使用其他的清晰度更高一些的数据集。

5.2 点云对抗攻击

针对 pointnet++ 进行 FGSM 的攻击是一次非常有意义的实验它并没有和我们预先所感受的那样，随着迭代步长 ε 的增大，模型的准确率一直增大。在此前的迭代中我发现模型的准确率稳定下降时我甚至想直接中断实验宣告成功。因为继续下去过高的步长被认为是单纯地破坏样本而不是有意义的扰动，即便它的攻击效果更出色。但是从大约到 ε 到了 0.2 时我惊人地发现模型的精度居然不降反升，这在传统的图像任务中是相当离奇的。我立刻跟踪了这个现象，发现模型的精度回升了一个令人叹为观止的地步。我下意识地反应出这是由于点云识别模型的采样特性导致的。当我们重新把对抗样本可视化以后的结果更加令我吃惊，大步长地扰动居然直接造成了“飞机”的分离而不是无意义的在它的周围增加噪声点。由于最远点采样的无偏性，原始的飞机并没有因为样本点的减少而产生识别偏差。当然这种精度的回升仅仅推理到这一步是不够的。后续的工作也许应该围绕更严谨的数学推理或者对 pointnet++ 结构更深入的剖析来展开。