Homework 4: Adversarial Attack on CIFAR-10

Implement White-Box PGD Attack and Black-Box Attack using PyTorch

In [ ]:
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import numpy as np
from torch.autograd import Variable
import matplotlib.pyplot as plt
```

In [ ]:
```
# 数据预处理
transform = transforms.Compose([
    transforms.ToTensor(),
])

# 加载 CIFAR-10 测试集
testset = torchvision.datasets.CIFAR10(
    root='./data', train=False, download=True, transform=transform)
testloader = torch.utils.data.DataLoader(
    testset, batch_size=1, shuffle=False, num_workers=2)
```

In [ ]:
```
from pytorchcv.model_provider import get_model as ptcv_get_model
resnet20_cifar10 = ptcv_get_model("resnet20_cifar10", pretrained=True)
resnet56_cifar10 = ptcv_get_model("resnet56_cifar10", pretrained=True)

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
target_model = resnet20_cifar10.to(device)  # 目标模型（攻击的对象）
surrogate_model = resnet56_cifar10.to(device)  # 代理模型（黑盒攻击使用）
```

In [ ]:
```
def test_clean_accuracy(model, test_loader, device):
    model.eval()
    correct = 0
    total = 0

    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()
            total += target.size(0)

    acc = 100. * correct / total
    print(f"[Clean Test] Accuracy: {correct}/{total} = {acc:.2f}%")
    return acc

# 测试目标模型原始精度
test_clean_accuracy(target_model, testloader, device)
```

```
[Clean Test] Accuracy: 8914/10000 = 89.14%
```

```
Out[ ]: 89.14
```

```
In [ ]:
def pgd_attack(model, image, label, epsilon=8/255, alpha=2/255, iters=8):
    """
    PGD 对抗攻击
    :param model: 目标模型
    :param image: 原始图像 (1,C,H,W)
    :param label: 真实标签
    :param epsilon: 扰动上限 (L∞)
    :param alpha: 单步扰动强度
    :param iters: 迭代次数
    :return: 对抗样本
    """
    # 初始化对抗样本
    perturbed_image = image.clone().detach().requires_grad_(True)

    for _ in range(iters):
        # 前向传播
        output = model(perturbed_image)
        loss = nn.CrossEntropyLoss()(output, label)

        # 梯度清零并反向传播
        model.zero_grad()
        if perturbed_image.grad is not None:
            perturbed_image.grad.data.zero_()
        loss.backward()

        # 生成扰动
        data_grad = perturbed_image.grad.data
        sign_data_grad = data_grad.sign()

        # 更新对抗样本
        perturbed_image = perturbed_image + alpha * sign_data_grad

        # 投影到 ε 邻域内
        perturbation = torch.clamp(
            perturbed_image - image,
            min=-epsilon,
            max=epsilon
        )
        perturbed_image = torch.clamp(image + perturbation, 0, 1).detach_()
        perturbed_image.requires_grad_(True)

    return perturbed_image.detach()

# 测试白盒攻击
def test_whitebox(model, testloader, epsilon=8/255):
    correct = 0
    total = 0

    for images, labels in testloader:
        images, labels = images.to(device), labels.to(device)

        # 生成对抗样本
        adv_images = pgd_attack(model, images, labels, epsilon)

        # 模型预测
        outputs = model(adv_images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    print(f'White-box Attack Accuracy: {accuracy:.2f}% (ε={epsilon})')
    return accuracy

# 执行测试
test_whitebox(target_model, testloader)
```

```
White-box Attack Accuracy: 0.05% (ε=0.03137254901960784)
```

Out[ ]:
```
0.05
```

In [ ]:
```
def blackbox_attack(target_model, surrogate_model, testloader, epsilon=8/255):
    correct = 0
    total = 0

    for images, labels in testloader:
        images, labels = images.to(device), labels.to(device)

        # 使用代理模型生成对抗样本
        adv_images = pgd_attack(surrogate_model, images, labels, epsilon)

        # 在目标模型上测试
        outputs = target_model(adv_images)
        _, predicted = torch.max(outputs.data, 1)

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

    accuracy = 100 * correct / total
    print(f'Black-box Attack Accuracy: {accuracy:.2f}% (ε={epsilon})')
    return accuracy

# 执行黑盒攻击测试
blackbox_attack(target_model, surrogate_model, testloader)
```

```
Black-box Attack Accuracy: 59.43% (ε=0.03137254901960784)
```

Out[ ]:
```
59.43
```