一、深度学习与对抗样本

1、深度学习的发展与应用

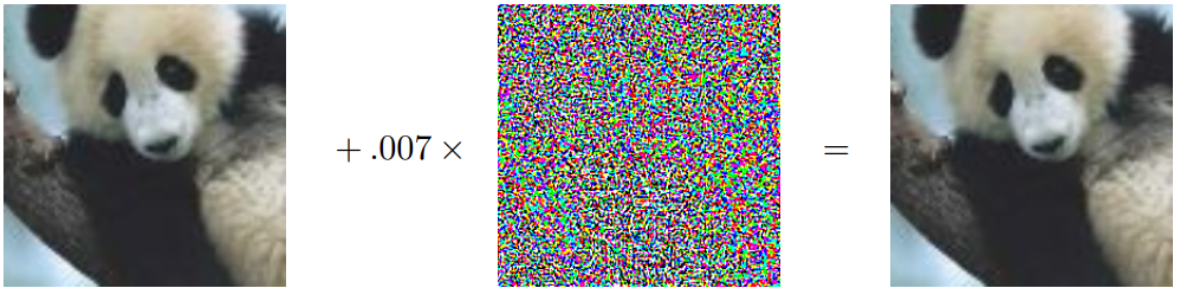
深度学习是机器学习领域的一个重要分支，它通过模拟人类神经网络的结构和功能，利用多层次的神经网络模型进行数据处理和学习。它在很多领域得到了广泛的应用。

> - 计算机视觉：图像分类，目标检测，人脸识别，图像生成等任务
> - 自然语言处理：文本分类，情感分析，机器翻译，语音识别等
> - 数据挖掘：深度学习模型可以从大规模的数据中发现模式和趋势，并用于预测、推荐系统和个性化服务等任务。

2、对抗样本的背景与定义

对抗样本由Christian Szegedy等人提出，是指在数据集中通过故意添加细微的干扰所形成的输入样本，导致模型以高置信度给出一个错误的输出。在原有图像上进行轻微改动而生成的对抗样本可以使模型得到十分荒谬的结果, 而这个改动对于人眼来说微不足道。因此对抗样本的出现对深度学习的安全带来了极大的影响和挑战。

> - 置信度：表示当A项出现时B项同时出现的频率
> - 如下图所示：By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image.



二、对抗样本的生成方法

对抗攻击分类

- 黑盒攻击：攻击者对攻击的模型的内部结构，训练参数，防御方法等等一无所知，只能通过输入输出与模型进行交互。
- 白盒攻击：与黑盒模型相反，攻击者对模型一切都可以掌握。目前大多数攻击算法都是白盒攻击。
- 无目标攻击：以图片分类为例，攻击者只需要让目标模型对样本分类错误即可，但并不指定分类错成哪一类。

- 有目标攻击：攻击者指定某一类，使得目标模型不仅对样本分类错误并且需要错成指定的类别。从难度上来说，有目标攻击的实现要难于无目标攻击。

1、基于梯度的方法

FGSM (Fast Gradient sign Method)

基本思想

随机梯度下降使得模型对于输入图像输出的损失函数值变小，从而使网络输出正确的预测，那么如果将计算得出的损失值加到输入图像上，使得网络输出的损失值变大，即可使网络趋向于输出错误的预测结果。

基本原理

1、假设有一个已经训练好的深度学习模型，用于分类任务。希望生成一个对抗样本使得输入样本经过微小修改后，模型将其误分类为错误的类别。

2、对于给定的输入样本x,计算其相对于损失函数的梯度，这个梯度表示了模型在输入样本上的敏感度。

3、将梯度的符号取反，即将其转换为最大化损失的方向。这是为了找到使模型预测错误的最小扰动方向。

4、根据所选的扰动大小ϵ,将梯度乘以 ϵ,并将其添加到输入样本中。这样就得到一个经过扰动的对抗样本$x^{\backprime}$,它与原始样本很接近但可以引导模型产生错误的输出。

> Let θ be the parameters of a model, x the input to the model, y the targets associated with x (for machine learning tasks that have targets) and $J(\theta, x, y)$ be the cost used to train the neural network. We can linearize the cost function around the current value of θ, obtaining an optimal max-norm constrained pertubation of $\eta = sign(\nabla x J(\theta, x, y))$

代码实现

```
# FGSM attack code
def fgsm_attack(image, epsilon, data_grad):
    # 使用sign（符号）函数，将对x求了偏导的梯度进行符号化
    sign_data_grad = data_grad.sign()
    # 通过epsilon生成对抗样本
    perturbed_image = image + epsilon*sign_data_grad
    # 做一个剪裁的工作，将torch.clamp内部大于1的数值变为1，小于0的数值等于0，防止image越界
    perturbed_image = torch.clamp(perturbed_image, 0, 1)
    # 返回对抗样本
    return perturbed_image
```

2、基于优化的方法

Deepfool

介绍

Deepfool旨在通过最小的扰动将输入样本移动到模型的决策边界上。与FGSM方法不同，DeepFool方法通过迭代地计算最小的扰动来生成对抗样本。

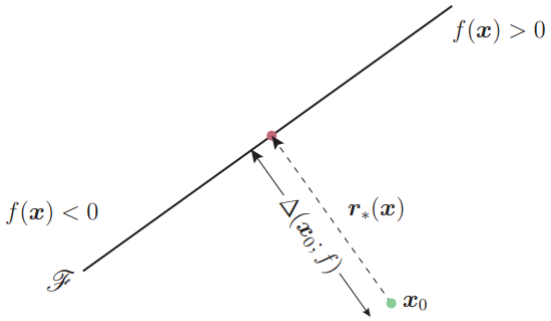
基本原理

二分类问题

对于二分类，超平面两侧对应不同的分类结果，对于在一侧的样本X，想要添加扰动使分类器将其分类为另一个类别，那么只需要将X更新到超平面的另外一侧就可以了。

Algorithm 1 DeepFool for binary classifiers

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\text{sign}(f(\boldsymbol{x}_i)) = \text{sign}(f(\boldsymbol{x}_0))$ **do**
    $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(x_i)\|_2^2} \nabla f(\boldsymbol{x}_i)$,
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$,
    $i \leftarrow i + 1$.
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$.



多分类器中的deepfool

1. 对于给定的输入样本x，我们首先计算模型对输入样本的预测结果。假设当前预测结果为y_0。
2. 我们初始化一个空的扰动向量，记为r，用于存储最小扰动。
3. 迭代地计算最小扰动：在每次迭代中，对于每个类别i（与当前预测结果y_0不同），计算模型的梯度以及与决策边界的距离。然后，选择梯度与决策边界距离的比值最小的类别，将其作为目标类别。
4. 通过最小化目标类别的线性化近似函数，计算最小扰动的方向和大小，并将其添加到扰动向量r中。
5. 更新输入样本x，将最小扰动r加到x上，得到新的样本x'。然后，重复步骤2-5，直到模型将x'误分类为错误的类别。

Algorithm 2 DeepFool: multi-class case

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
    for $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
        $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
        $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    end for
    $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
    $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
    $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$

总结

DeepFool方法通过迭代计算最小扰动的方式，使得生成的对抗样本能够尽可能地接近决策边界，从而增加了攻击的成功率。相对于FGSM方法，DeepFool方法更具有挑战性，因为它能够应对更复杂的模型和防御机制。

但是DeepFool方法也有一些限制。它需要较多的迭代次数才能生成对抗样本，并且计算成本较高。此外，对于某些复杂的模型和数据集，DeepFool可能会遇到困难，因为决策边界的复杂性可能导致较大的扰动。

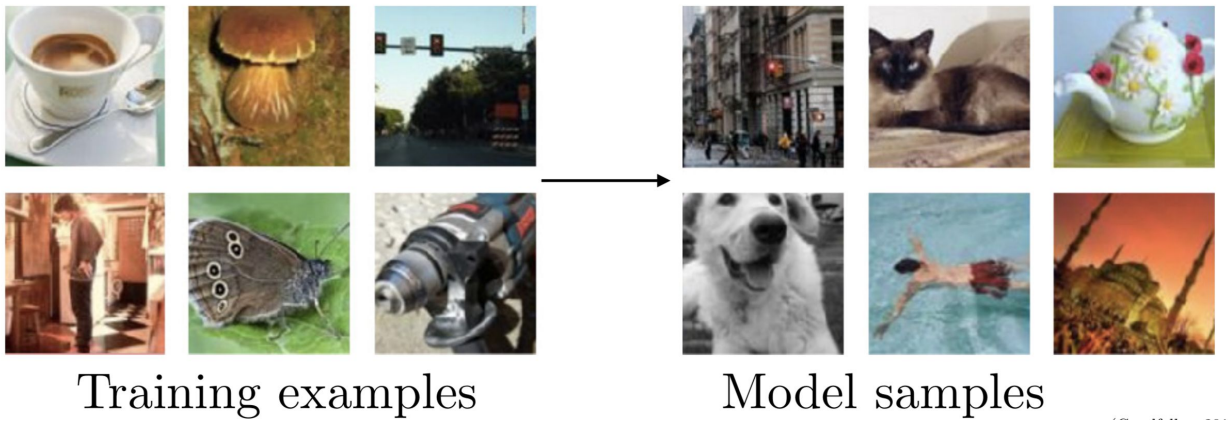
```
def deepfool(image, net, num_classes=10, overshoot=0.02, max_iter=100):
    is_cuda = torch.cuda.is_available()
    if is_cuda:
        # print("Using GPU")
        image = image.cuda()
        net = net.cuda()

    f_image = net.forward(Variable(image[None, :, :, :],
requires_grad=True)).data.cpu().numpy().flatten()
    I = (np.array(f_image)).flatten().argsort()[::-1]

    I = I[0:num_classes]
    label = I[0]

    input_shape = image.cpu().numpy().shape
    pert_image = copy.deepcopy(image)
    w = np.zeros(input_shape)
    r_tot = np.zeros(input_shape)
```

```
    loop_i = 0

    x = Variable(pert_image[None, :], requires_grad=True)
    fs = net.forward(x)
    fs_list = [fs[0,I[k]] for k in range(num_classes)]
    k_i = label

    while k_i == label and loop_i < max_iter:

        pert = np.inf
        fs[0, I[0]].backward(retain_graph=True)
        grad_orig = x.grad.data.cpu().numpy().copy()

        for k in range(1, num_classes):
            if x.grad is not None:
              x.grad.zero_()

            fs[0, I[k]].backward(retain_graph=True)
            cur_grad = x.grad.data.cpu().numpy().copy()

            # set new w_k and new f_k
            w_k = cur_grad - grad_orig
            f_k = (fs[0, I[k]] - fs[0, I[0]]).data.cpu().numpy()

            pert_k = abs(f_k)/np.linalg.norm(w_k.flatten())

            # determine which w_k to use
            if pert_k < pert:
                pert = pert_k
                w = w_k

        # compute r_i and r_tot
        # Added 1e-4 for numerical stability
        r_i =  (pert+1e-4) * w / np.linalg.norm(w)
        r_tot = np.float32(r_tot + r_i)

        if is_cuda:
            pert_image = image + (1+overshoot)*torch.from_numpy(r_tot).cuda()
        else:
            pert_image = image + (1+overshoot)*torch.from_numpy(r_tot)

        x = Variable(pert_image, requires_grad=True)
        fs = net.forward(x)
        k_i = np.argmax(fs.data.cpu().numpy().flatten())

        loop_i += 1
    r_tot = (1+overshoot)*r_tot
    return r_tot, loop_i, label, k_i, pert_image
```

3、利用GAN生成对抗样本

生成模型简介

生成模型描述的是：我们接收了从分布 p_{data} 取样的若干样本构成我们的训练集，模型会学习到一个模拟这一分布的概率分布 p_{model} ，在有些情况下，我们可以直接估计概率分布。而另一些情况下，我们可以从 p_{model}中生成一些以假乱真的图片。GAN通过博弈的方式，同时训练一个生成器网络和一个判别器网络，以达到生成逼真样本的目的。



GAN与生成对抗样本

1. 准备数据集：首先，需要准备一个训练数据集，该数据集包含了GAN要生成的样本类别的真实样本。这些样本可以是图像、文本或其他类型的数据。
2. 定义生成器和判别器网络：创建一个生成器网络和一个判别器网络。生成器网络接受一个随机噪声向量作为输入，并输出一个生成的样本。判别器网络接受生成器生成的样本和真实样本作为输入，并输出一个判断样本真实性的概率。
3. 训练GAN：在训练过程中，生成器和判别器相互竞争和协作。生成器试图生成逼真的样本以骗过判别器，而判别器则努力区分生成的样本和真实样本。
4. 生成对抗样本：在训练完成后，可以使用生成器网络来生成对抗样本。为了生成对抗样本，需要提供一个输入噪声向量，并通过生成器网络获取对应的输出。
5. 评估生成的对抗样本：生成的对抗样本可以输入到目标深度学习模型中，以评估其对目标模型的攻击效果。通常，生成的对抗样本会被设计成在目标模型上引发错误分类或误导性结果。

三、防御对抗样本生成的方法

对抗训练

- 鲁棒性定义：指系统在扰动或不确定的情况下仍能保持它们的特征行为。

对抗训练是一种用于提高深度学习模型对抗样本鲁棒性的方法。它通过将对抗样本引入到模型的训练数据中，迫使模型学习对对抗样本更加鲁棒的特征表示和决策边界。

对抗训练的基本思想如下：

1. 准备训练数据集：首先，需要准备一个包含真实样本和对抗样本的训练数据集。真实样本用于模型的普通训练，而对抗样本用于模型的对抗训练。
2. 生成对抗样本：使用一种对抗样本生成方法（如FGSM、PGD等）来生成对抗样本。

3. 模型训练：在训练过程中，真实样本和对抗样本被混合在一起，并与相应的标签一起提供给模型进行训练。对抗样本的存在迫使模型学习到更鲁棒的特征和决策边界，从而提高对抗样本的分类准确性。
4. 迭代训练：对抗训练往往需要进行多个训练周期。在每个训练周期中，使用真实样本和生成的对抗样本进行模型的训练。通过多次迭代，模型逐渐学习到对抗样本的特征，并提高其鲁棒性。

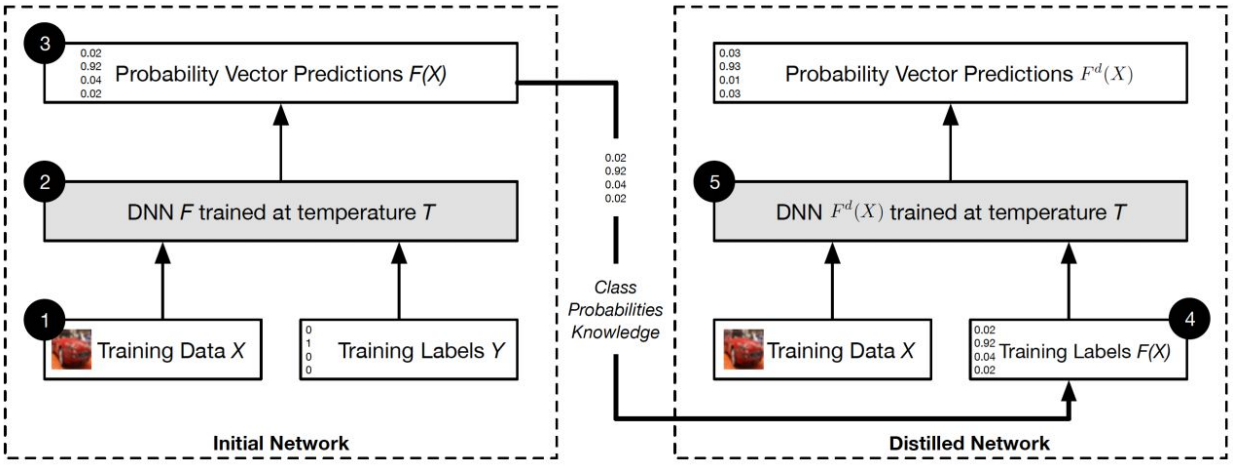
We found that training with an adversarial objective function based on the fast gradient sign method was an effective regularizer:

$$\tilde{J}(\boldsymbol{\theta}, \boldsymbol{x}, y) = \alpha J(\boldsymbol{\theta}, \boldsymbol{x}, y) + (1 - \alpha) J(\boldsymbol{\theta}, \boldsymbol{x} + \epsilon \text{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right).$$

防御性蒸馏

主要思想

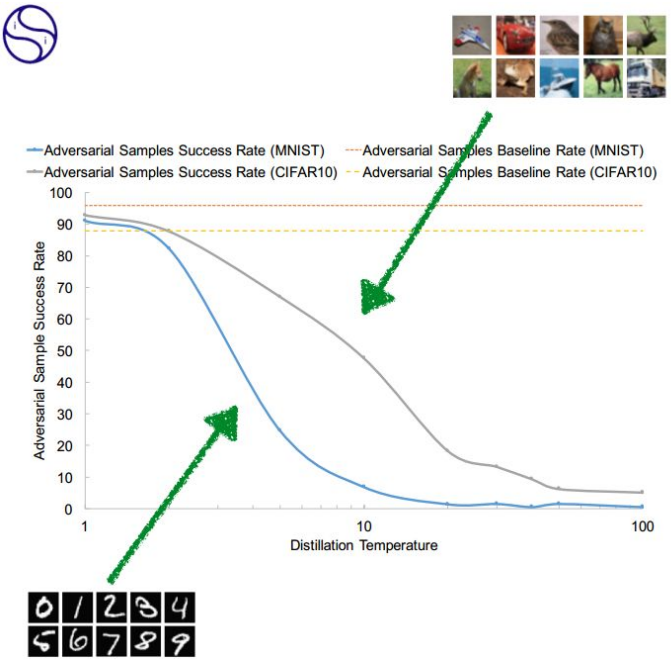
使用从DNN中提取的知识来降低生成对抗样本时的梯度，如果这个对抗梯度很高，那么扰动很大，DNN的输出不稳定；为了抵抗这种扰动，需要减少输入周围的变化，即使用防御蒸馏来平滑训练得到的模型，提高模型的泛化能力，从而令模型对对抗样本具有高弹性。



> 防御蒸馏的概述：首先在数据 X上使用硬标签训练一个初始网络F，$softmax$ 的 temperature 为T。然后使用概率向量 $F(X)$作为软标签，以在相同数据X和相同的T训练蒸馏网络F^d

防御效果

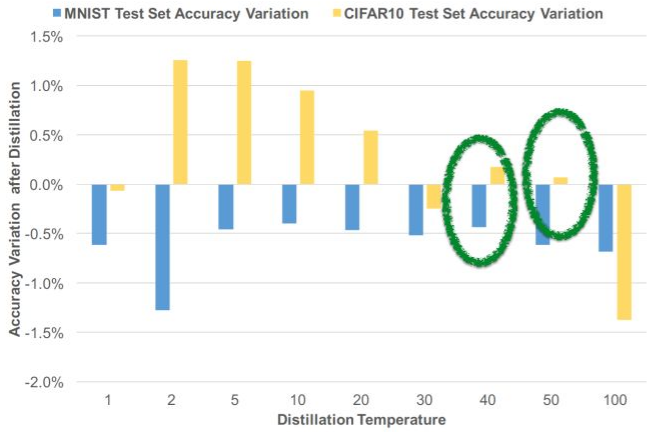
对对抗样例攻击成功率的影响



Distillation Temperature	MNIST Adversarial Samples Success Rate (%)	CIFAR10 Adversarial Samples Success Rate (%)
1	91	92.78
2	82.23	87.67
5	24.67	67
10	6.78	47.56
20	1.34	18.23
30	1.44	13.23
40	0.45	9.34
50	1.45	6.23
100	0.45	5.11
No distillation	95.89	87.89

随着T的增大，对抗样例攻击的成功率在迅速的下降

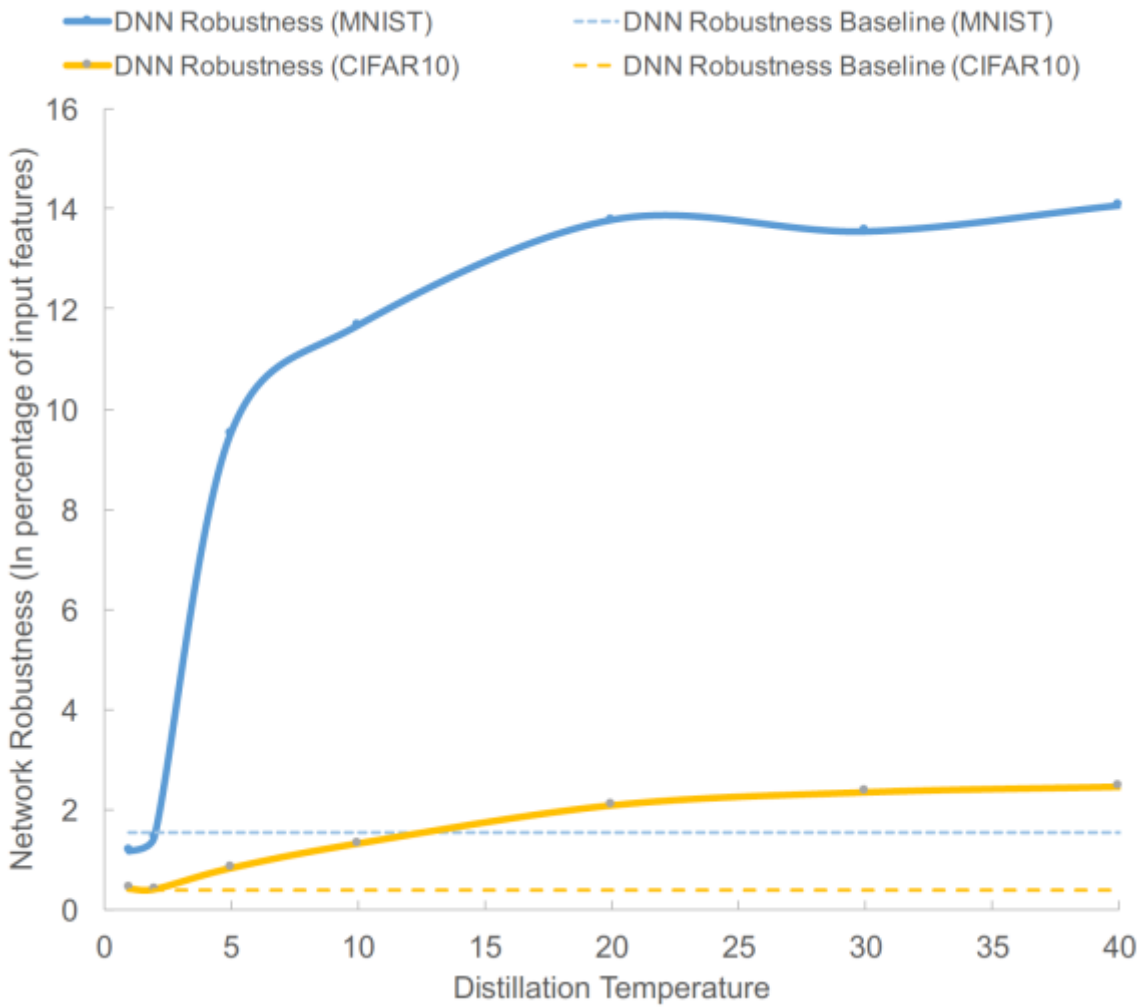
对模型准确率的影响



Distillation Temperature	MNIST Adversarial Samples Success Rate (%)	CIFAR10 Adversarial Samples Success Rate (%)
1	91	92.78
2	82.23	87.67
5	24.67	67
10	6.78	47.56
20	1.34	18.23
30	1.44	13.23
40	0.45	9.34
50	1.45	6.23
100	0.45	5.11
No distillation	95.89	87.89

尽管这种防御方式在一定程度上会降低模型的准确率，但对模型准确率的负面影响最多不会超过1.5%，满足对防御算法设计的要求。

对DNN鲁棒性的影响



随着T的增加，DNN的鲁棒性越来越强，因此，抵御对抗样例攻击的能力也就越来越强。

其他方法

1. 输入预处理：输入预处理方法通过对输入样本进行预处理或变换来防御对抗样本。例如，可以对输入图像进行降噪、模糊化、剪切等操作，以减少对抗样本的影响。输入预处理方法主要关注在输入样本上进行操作，而不涉及模型的修改。
2. 梯度掩蔽：梯度掩蔽方法试图限制对抗样本生成过程中的梯度信息，使得攻击者难以获取准确的梯度信息来生成对抗样本。这些方法可以通过添加噪声、截断梯度或者限制梯度传播来保护模型的鲁棒性。
3. 检测与拒绝：检测与拒绝方法通过在推理过程中检测和拒绝对抗样本来保护模型的输出。这些方法通常使用附加的检测模块或阈值来判断输入样本是否为对抗样本，如果是，则拒绝对其进行分类。
4. 集成防御：集成防御方法通过使用多个独立的模型来对抗对抗样本。这些模型可以是不同的结构或使用不同的训练策略，通过对多个模型的集成输出进行投票或者综合，可以提高对抗样本的检测和鲁棒性。

四、总结——挑战与未来研究方向

总结

1. 演化和发展：对抗样本研究已经经历了快速的演化和发展。从最早的基于梯度信息的攻击方法到后来的优化算法和生成对抗网络（GAN），对抗样本攻击技术变得越来越复杂和有效。

2. 模型脆弱性：对抗样本揭示了深度学习模型的脆弱性和不可解释性。即使在高准确性的情况下，模型仍然容易受到微小扰动的影响，导致错误的预测。这对于许多实际应用而言是一个重要的安全问题。
3. 转移性和通用性：对抗样本攻击具有一定的转移性和通用性，即攻击成功的对抗样本可以欺骗其他模型和任务。这加剧了对抗样本攻击的威胁，并使得模型的鲁棒性更具挑战性。
4. 防御方法：为了抵御对抗样本攻击，研究人员提出了许多防御方法，包括对抗训练、防御性扰动、模型压缩等。这些方法试图提高模型的鲁棒性，并抵抗对抗样本攻击的影响。

展望

1. 对抗样本攻击的进一步研究：随着对抗样本攻击技术的不断发展，未来的研究将继续探索更复杂和有效的攻击方法。这将包括更强大的优化算法、更复杂的生成模型以及针对特定任务和模型的攻击策略。
2. 提高模型的鲁棒性：对抗样本攻击的研究也将促使对模型的鲁棒性和可解释性进行更深入的探索。研究人员将努力开发新的防御方法和技术，以提高模型对抗样本攻击的抵抗力，并增加模型的可解释性和鲁棒性。
3. 融合多种防御策略：未来的研究将趋向于综合多种防御策略和方法，以提供更强大的防御机制。模型的鲁棒性将通过结合对抗训练、防御性扰动、模型压缩等方法来增强。
4. 基于传统的黑盒攻击，目前（目标模型）的精度降低范围为40-50%，∞扰动规范为15/255，但这只针对无目标的欺骗。对于有目标的欺骗仍待提高。黑盒攻击可以在具有架构相似性的模型之间更好地转移。
5. 物理世界中的3D对抗攻击发展速度远低于数字对抗攻击。
6. 视觉模型可以和其他许多模型相结合，扩展到多模型任务，如图像/视频字幕。

五、参考文献

[1] GOODFELLOW I J, SHLENS J, SZEGEDY C. Explaining and Harnessing Adversarial Examples[J]. 2014: 1–11.

[2] MOOSAVI-DEZFOOLI S M, FAWZI A, FROSSARD P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks[J]. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2016, 2016-Decem: 2574–2582.

[3]NIPS 2016 Tutorial: Generative Adversarial Networks

[4] (5条消息) 初探对抗攻击——黑盒攻击&白盒攻击*黑盒对抗攻击*Wwwwhy　的博客-CSDN博客

[5] 鲁棒性 - 维基百科，自由的百科全书 (wikipedia.org)

[6] ADVERSARIAL TRAINING METHODS FOR SEMI-SUPERVISED TEXT CLASSIFICATION

[7] PAPERNOT N, MCDANIEL P, WU X等. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks[J]. Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016, 2016: 582–597.

[8] (5条消息) 【论文笔记】（防御蒸馏）Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks_Set 的博客-CSDN博客

[9] Nicolas Papernot∗, Patrick McDaniel∗, Xi Wu§, Somesh Jha§, and Ananthram Swami‡ ∗Department of Computer Science and Engineering, Penn State University §Computer Sciences Department, University of Wisconsin-Madison ‡United States Army Research Laboratory, Adelphi, Maryland Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks

[10] 深度学习中的对抗样本 Adversarial examples in deep learning - 知乎 (zhihu.com)

[11] [9] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012.