使用噪音生成改善模型对多攻击的鲁棒性的复现

朱庆瀛

摘要

面对对抗攻击，人们提出来了对抗训练来防御攻击，可是如今大多数的对抗训练只是针对某一种攻击来防御，这导致同样会遭遇到其它类型的攻击，所以在现实生活中对多种攻击的防御是必要的。虽然现今仍有对多攻击的防御的方法，但是这些方法计算量太大，于是原论文作者提出了使用元噪声生成器来对干净样本产生噪声以此提高模型对多攻击的鲁棒性。元噪声生成器类似于 GAN 可以生成某一种分布的图像，而不会去过拟合生成某一特定的扰动，并且在复现中证实了该方法对多攻击的防御表现出众。

关键词：对抗防御；多攻击的鲁棒性；元噪声生成器；

1 引言

如今深度学习已经用在了生活中的许多方面，但是随着人们发现 DNN 没有那么可靠，比如在视觉方面，给一张图片加上一些人眼观察不到的噪音会造成 DNN 识别图片错误，这引起了很大的安全隐患。值得注意的是现存的神经网络对对抗扰动是极其敏感的，所以我们必须采用强有力的防御来抵挡皆有可能的攻击以此来提高神经网络的可靠性。

对抗样本的出现已经受到了极大关注，也已经出现了一些对抗防御来抵挡对抗样本的攻击去规避这类现象。但是这些防御方法大多数都是针对一种攻击来进行防御的且现实生活中出现的攻击是多样化的，所以这些防御方法在现实生活中是很难奏效的。并且现存的对于对抗多扰动攻击的方法都是需要大量计算来实现。

于是就挑选了使用噪音生成来改善模型对多攻击的鲁棒性的方法的复现。因为该方法可以减少计算量并且能在多攻击时实现较好的鲁棒性来解决现存方法的缺点，同时该新颖的方法可以启发后人对对抗攻击方向的创新，并且提出了对抗一致性来迫使不同的扰动生成的多个图像在神经网络中产生的结果一样来使得网络更加光滑且鲁棒。

2 相关工作

2.1 对抗训练

在标准的对抗训练中，提出了模型的目标函数称为 min-max 方程式。具体的来讲就顺内部产生使损失函数值最大的对抗扰动，外部是去最小化对抗扰动产生的交叉熵损失来更新模型参数，其中损失函数采用的是交叉熵损失函数。方程如下：

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\arg\max_{k} \mathcal{L}_{\mathrm{cls}}\left(f_{\theta}\left(\mathcal{A}_{k}\left(x\right)\right), y\right)\right]$$

其中 f 表示深度神经网络，θ 表示 f 的权重，D 表示干净示例 x 和 ground truth 标签 y，其中 A(x) 表示是在 x 的基础上生成的对抗样本，L 为表示更新训练模型的损失函数。

2.2 对抗攻击

由于 Szegedy et al. 提出对抗实例会误导深度神经网络，因此提出了许多有效的对抗攻击方法，如快速梯度符号法 (FGSM)、投影梯度下降攻击 (PGD)[27]、Carilini and Wagner 攻击 (CW)[5]、基于 jacobian 的显著图攻击 (JSMA)[29]。现有的攻击方法分为白盒攻击和黑盒攻击。白盒攻击是在生成对抗例时知道被攻击模型的所有参数信息，黑盒攻击是在生成对抗例时只知道被攻击模型的部分输出。通常，黑盒攻击通过反复查询目标模型 (查询型攻击) 或搜索与目标模型相似的替代模型 (转移型攻击) 来模拟模型梯度。由于在实际应用中攻击者几乎不知道目标模型的模型参数，因此模型对黑盒攻击的性能更能体现真实的鲁棒性，但所挑选的复现论文为去防御多种类型的白盒攻击。

2.3 多对抗攻击

现如今通过优化对抗训练的最外部目标函数来扩展到多扰动上，其中有几个典型方法如下。1) The maximum over all perturbations. 内部 argmax 表示在多攻击情况下去找到使损失函数最大化的那个对抗样本，然后再通过外部目标函数去优化模型。该方法的方程如下:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\arg\max_{k} \mathcal{L}_{\mathrm{cls}}\left(f_{\theta}\left(\mathcal{A}_{k}\left(x\right)\right), y\right)\right]$$

2) The average over all perturbations，直接对每个扰动生成对抗样本并且计算所有扰动产生的损失值的平均值去优化外部的目标函数来对抗多扰动攻击。该方法的方程如下。

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \frac{1}{n} \sum_{k=1}^{k=n} \mathcal{L}_{\mathrm{cls}}\left(f_{\theta}\left(\mathcal{A}_{k}\left(x\right), y\right)\right)$$

还有一个经典的方法名为“Multi Steepest Descent”(MSD) 相当于是在 PGD 基础上每次迭代选取下一步所走方向时，对每个扰动计算抖动方向然后选择最抖动的方向走。但是这些方法每次都需要对每一个扰动计算梯度去训练使得计算量都特别大。

2.4 生成模型

现今许多学者使用生成模型去产生某种特定的攻击，而这样会造成模型过度拟合到某种类型攻击。比如通过使用生成器将干净样本转到某空间，然后用 discriminator 来区分是否为干净样本。已经有各种尝试利用生成模型的代表性来提高模型的鲁棒性，但是这些方式都不能防御强有力的攻击，而挑选的论文可以中使用的元学习生成器是不去特意拟合某种类型的攻击而去降低多扰动的对抗错误。

3 本文方法

3.1 本文方法概述

首先给出所选论文的方法图示，如图 1所示：

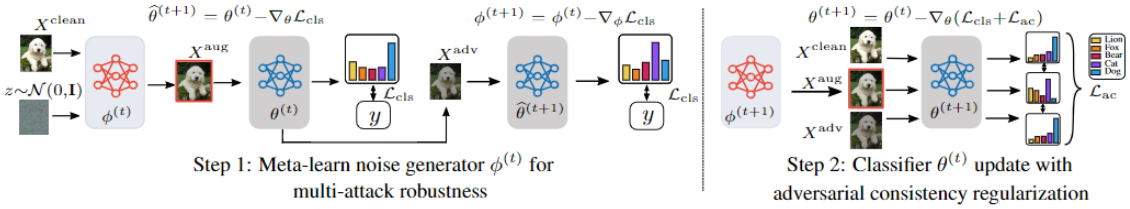

图 1: Overview of Meta-Noise Generator with Adversarial Consistency (MNG-AC)

生成器 φ(t) 取随机噪声，输入 Xclean 生成噪声增强样本 Xaug，将该增强样本丢人 DNN 对分类器进行更新 (使用交叉熵损失函数)，此更新为对分类器的临时更新，以增加增强样本的影响。左边步骤一是生成器通过元学习来学习，通过最小化从扰动集中均匀采样的攻击产生的对抗样本的损失 (在临时更新的分类器上的损失) 来学习。右边步骤二是对分类器进行更新，分类器 θ(t) 使随机对抗分类损失 Lcls 和对抗一致性损失 Lac 最小化，其中对抗一致性损失是使得对同一张图片的不同扰动所产生的结果要一致。

3.2 随机对抗训练

为了降低先前的对抗多扰动攻击的方法的计算量，挑选的论文提出了随机对抗训练，该方法可以保证在训练过程中尽可能去覆盖到训练所有攻击。首先我们将每一种攻击用 $A_i(x)$ 来代替，并且将所考虑到的攻击加入到集合 S 中，其中我们采用的是均匀取样，在进行随机对抗训练过程中，每一次训练会从该集合中均匀采取一个攻击进行对抗训练。这可以防止过拟合于某种特定攻击，因此该方法是极其有效的在改善计算量方面并且提高了模型的泛化性。具体做法如下：

$$S = \{\mathcal{A}_1(x), \ldots, \mathcal{A}_n(x)\},$$
$$k \sim \mathrm{Cat}\left((1/n, \ldots, 1/n)\right),$$
$$\mathcal{A}(x) = S_k(x),$$

其中 cat 表示一个均匀分布，A(x) 表示所选择的对抗攻击。随机对抗训练 (SAT) 不像平均或者最大方法那样计算量大，SAT 只需要常量的时间就可以完成。

3.3 损失函数定义

复现的论文采用对抗一致性损失以及元噪声生成器去帮助模型泛化到多扰动攻击上，模型的目标损失函数由两部分组成，一个为随机对抗扰动产生的损失 L_{clx} 和一个对抗性一致损失 L_{ac} 。具体如下所示：

$$\mathcal{L}_{\text{total}} = \frac{1}{B}\sum_{i=1}^{B} \underbrace{\mathcal{L}_{\text{cls}}\left(\theta \mid x_\theta^{\text{adv}}(i), y(i)\right)}_{\text{SAT classification loss}} + \underbrace{\beta \cdot \mathcal{L}_{\text{ac}}\left(p^{\text{clean}}(i); p^{\text{adv}}(i); p^{\text{aug}}(i)\right)}_{\text{adversarial consistency loss}},$$

其中 B 表示 batchsize,β 是一个超参数，SAT 分类损失就是使用一个干净样本产生的对抗样本去攻击模型所产生的损失，其中对抗一致性损失采用的是 JS 散度来进行计算的，因为模型最后输出的结果是一个分布，所以这个损失函数是为了尽量让同一张图片的不同扰动产生的输出分布大致相同。

p^{clean} 代表先验分布，实则就是表示在模型输入干净样本时输出的结果，同理其它两个分布也表示先验概率。 L_{ac} 的细则如下:

$$\mathcal{L}_{\mathrm{ac}} = \frac{1}{3}\left(D_{\mathrm{KL}}(p^{\mathrm{clean}} \parallel M) + D_{\mathrm{KL}}(p^{\mathrm{adv}} \parallel M) + D_{\mathrm{KL}}(p^{\mathrm{aug}} \parallel M)\right),$$

其中 $M = \frac{p^{clean} + p^{adv} + p^{aug}}{3}$。$L_{ac}$ 在基于这样的假设下，当输入相同图像的不同扰动版本时，分类器应该输出相似的预测。对于该对抗损失一致性函数会降低模型在正确样本上的正确率，于是我在第四步中的创新点提出了改进过后的目标损失函数 (见第四步中创新点) 来改进 MNG-AC 在干净样本上的准确率。

4 复现细节

4.1 与已有开源代码对比

此部分为必填内容。如果没有参考任何相关源代码，请在此明确申明。如果复现过程中引用参考了任何其他人发布的代码，请列出所有引用代码并详细描述使用情况。同时应在此部分突出你自己的工作，包括创新增量、显著改进或者新功能等，应该有足够差异和优势来证明你的工作量与技术贡献。

如需伪代码，采用如下的写作方式进行描述

Procedure 1 Algorithm for MNG-AC.

Input: Dataset *D,* epochs *T,* batch size *B*
, perturbation set *S,* classfier θ, noise − generator ϕ
Output: final model parameters θ
for *t in [1,...,T]* **do**
  add X^{Aug} to the classifier model
  use SAT to get the X^{Adv}
  使用 X^{Aug} 加入到分类器后产生的损失去临时更新模型参数θ
  用 X^{Adv} 加入分类器产生的损失去更新元噪声模型的参数ϕ
  将更新过后的元噪声生成器产生的 X^{Aug} 与 X^{Adv}, X^{clean} 分别加入到分类其中得到对抗一致性损失函数
  根据总损失函数更新分类器参数θ
  end

4.2 实验环境搭建

使用 vscode 进行代码编写，使用 conda 创建代码所需的虚拟环境，在该虚拟环境下下载运行代码所需要的包，例如 pytorch、numpy、python 等一些包，其次在下载好所需的实验数据包 cifar10-data，这样就完成了实验环境搭建。

4.3 界面分析与使用说明

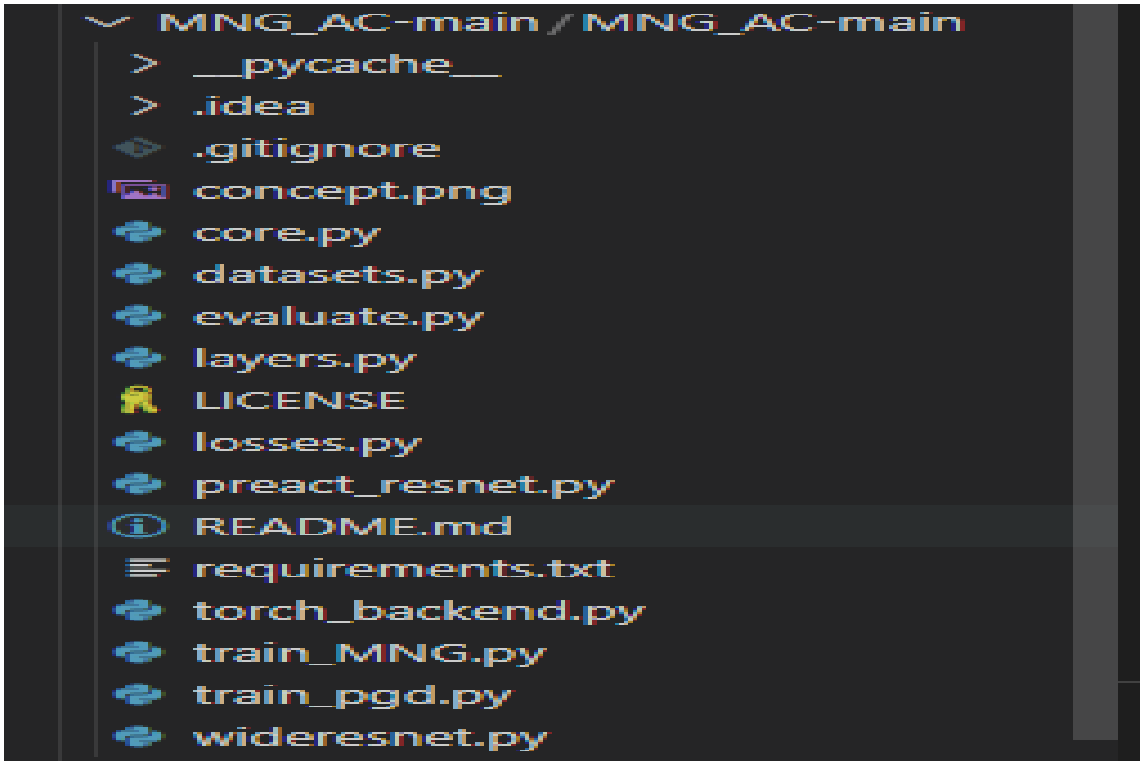

图 2: MNG-AC 实现包

要复现出实验结果，你先要安装依赖包 (比如 numpy torchvision 等), 可以在虚拟环境终端下输入指令:pip install -r requirements.txt

其次训练模型, 在选择复现的该篇论文中我们需要训练出六个模型，它们都是通过对抗训练得到的，比如 Adv_{avg} 表示在多攻击情况下，对多攻击的损失函数总和求平均来作为对抗训练的目标函数进行优化模型，剩下五个模型分别为: Adv_1、 Adv_2、 Adv_∞、 Adv_{max}、MNG-AC。

训练 MNC-AC 是执行的 train_MNG.py 文件，在终端输入的指令为: python train_MNG.py –fname xxx –dataset cifar10 –model WideResNet。

训练另外五个是执行的 train_pgd.py 文件，比如你要获得 Adv_1 的模型, 你可以在终端输入如下指令：python evaluate.py –fname xxx –dataset cifar10 –model WideResNet –attack_lib pgd –norm l1。

其中 fname 表示存储训练完后的模型的文件名, 还有其它四个模型，你只需要修改 fname 以及 norm 这两个参数就可以获得其它四个模型。

4.4 创新点

根据原论文的实验结果观察发现, 使用 MNG-AC 的实验结果表明：训练过后的分类器模型对输入 X_{clean} 的正确率最低 (相对于使用当今其它对抗多攻击方法训练过后的模型)。

所以产生了如下动机: 提高模型在输入 X_{clean} 上的准确率。核心的思想是去改变原论文当中的对抗一致性损失的目标函数，原论文中的对抗一致性损失目标函数为:

$$L_{ac} = \frac{1}{3}(D_{KL}(p^{clean}||M) + D_{KL}(p^{adv}||M) + D_{KL}(p^{aug}||M))$$

其中 M 为： $\text{M} = \dfrac{p^{clean} + p^{adv} + p^{aug}}{3}$

根据该对抗性损失一致目标函数，首先该目标函数是使用模型在 X_{clean}、X_{adv}、X_{aug} 上的 softmax 之后的输出分布然后让这三个分布的输出分布尽量一致。我的想法是我觉得应该让这三个分布尽量与

真实标签的 one-hot 编码的分布一致，这样才能使得模型在 X_{clean} 上的准确率尽可能更高，所以我将对抗一致性损失目标函数修改为如下格式:

$$L_{ac} = \frac{1}{2}(D_{KL}(p^{adv}||M) + D_{KL}(p^{aug}||M))$$

其中修改之后的 M 为: $M = \frac{[one-hot] + p^{adv} + p^{aug}}{3}$，其中 [one-hot] 表示该训练样本的标签所对应的 one-hot 编码。

创新过后的实验结果在第 5 步实验结果分析中展示。

5 实验结果分析

对多种扰动的鲁棒性比较。所有的值都通过计算平均值和三次试验的标准差来测量，最佳和次优结果分别用粗体和下划线突出显示。

	Acc_{clean}	l_∞	l_1	l_2
Adv_∞	85.35	42.91	25.47	61.83
Adv_1	93.25	0.0	<u>97.41</u>	2.9
Adv_2	88. 31	36.1	64.9	**71.42**
Adv_{avg}	92.21	32.47	61.29	65.2
Adv_{max}	86.75	<u>44.67</u>	61.03	62.83
MNG-AC	81.82	**46.75**	**68.31**	<u>71.68</u>

图 3: 原论文 MNG-AC 的实验结果示意

图 3 展示的是在 CIFAR-10 数据集上做的测试结果，特别的我们使用半径 $\varepsilon = \{\frac{8}{255}, \frac{2000}{255}, \frac{128}{255}\}$ 来分别作为 l_∞、 l_1、 l_2 扰动半径。结果显示 MNG-AC 相对于 Adv_{avg}、 Adv_{max} 来说在单扰动上表现的性能较好，特别的是训练时间减少了很多。

但是观察实验结果发现 MNG-AC 在 Acc_{clean} 的实验结果最差，于是对第 4 步中的创新点进行实验以来提高 MNG-AC 在 Acc_{clean} 上的正确率，改变过后的 MNG-AC 的实验结果如下，其中黑体加下划线突出表示正确率更高。

	Acc_{clean}	l_∞	l_1	l_2
MNG-AC(original)	81.82	<u>**46.75**</u>	68.31	71.68
MNG-AC(updated)	<u>**88.05**</u>	45.45	<u>**69.87**</u>	<u>**74.54**</u>

图 4: MNG-AC 与创新过后的 MNG-AC 对比

同样实验结果是在 CIFAR-10 数据集上进行测试，从测试结果可以发现对原论文的创新是有效的，尤其在 Acc_{clean} 的正确率提高了将近 7 个百分点，并且在 l_1、 l_2 扰动上的准确率略有提升。总的来说，这次改进是有一定意义的，但是对于在攻击上的提升并不是很大，所以后续还需进一步在原论文基础上进行创新。

6 总结与展望

挑选的复现的论文解决了多重对抗性扰动下的鲁棒性问题。提出了一种新的元噪声发生器 (MNG)，它学会了通过在不同的扰动中产生输出噪声来随机扰动干净的示例。然后使用对抗一致性 (AC) 损失来训练模型，该损失解释了干净、对抗和增强样本之间的标签一致性。此外，为了解决传统对抗训练方法对多个扰动的计算开销问题，引入了随机对抗训练 (SAT)，它从扰动的分布中采样一个扰动。最终实验结果显示在多种攻击下的结果都表现不错，但是不足点是在干净样本上的正确是最低的一个，这是目前存在的一个最大问题。对抗训练会造成模型在干净样本上的正确率降低这是一个极其需要关注的问题，虽然鲁棒性提高了，但是会失去正常情况下的正确率，可以在训练时对损失函数进行改造，增加损失函数的自适应性以此来调整在干净样本和对抗样本之间的正确率。未来可在该框架基础上去探究怎么进一步提升在干净样本上的正确率和在攻击样本上的正确率以来达到创新。[1]

参考文献

[1] MADAAN D, SHIN J, HWANG S J. Learning to generate noise for multi-attack robustness[C]//International Conference on Machine Learning. 2021: 7279-7289.