Towards Evaluating the Robustness of Neural Networks

关于评估神经网络的鲁棒性

Nicholas Carlini David Wagner
尼古拉斯 · 卡尔尼 David Wagner
University of California, Berkeley
加利福尼亚大学伯克利分校

ABSTRACT

摘要

Neural networks provide state-of-the-art results for most machine learning tasks. Unfortunately, neural networks are vulnerable to adversarial examples: given an input x and any target classification t , it is possible to find a new input x' that is similar to x but classified as t . This makes it difficult to apply neural networks in security-critical areas. Defensive distillation is a recently proposed approach that can take an arbitrary neural network, and increase its robustness, reducing the success rate of current attacks' ability to find adversarial examples from 95% to 0.5% .

神经网络在大多数机器学习任务中提供了最先进的结果。不幸的是，神经网络对对抗样本是脆弱的: 给定输入 x 和任何目标分类 t ，可以找到一个新的输入 x' ，它与 x 相似，但被分类为 t 。这使得在安全关键领域应用神经网络变得困难。防御蒸馏是一种最近提出的方法，它可以对任意神经网络进行处理，提高其鲁棒性，将当前攻击找到对抗样本的成功率从 95% 降低到 0.5% 。

In this paper, we demonstrate that defensive distillation does not significantly increase the robustness of neural networks by introducing three new attack algorithms that are successful on both distilled and undistilled neural networks with 100% probability. Our attacks are tailored to three distance metrics used previously in the literature, and when compared to previous adversarial example generation algorithms, our attacks are often much more effective (and never worse). Furthermore, we propose using high-confidence adversarial examples in a simple transferability test we show can also be used to break defensive distillation. We hope our attacks will be used as a benchmark in future defense attempts to create neural networks that resist adversarial examples.

在本文中，我们通过引入三种新的攻击算法来证明防御蒸馏并未显著提高神经网络的鲁棒性，这些算法在蒸馏和未蒸馏的神经网络上都以 100% 的概率成功。我们的攻击针对文献中先前使用的三种距离度量进行定制，与以前的对抗样本生成算法相比，我们的攻击通常更有效 (且从未更差)。此外，我们建议在一个简单的可转移性测试中使用高置信度的对抗样本，我们展示该测试也可以用来突破防御蒸馏。我们希望我们的攻击能够作为未来防御尝试的基准，以创建抵抗对抗样本的神经网络。

I. INTRODUCTION

I. 引言

Deep neural networks have become increasingly effective at many difficult machine-learning tasks. In the image recognition domain, they are able to recognize images with near-human accuracy [27], [25]. They are also used for speech recognition [18], natural language processing [1], and playing games [43], [32].

深度神经网络在许多困难的机器学习任务中变得越来越有效。在图像识别领域，它们能够以接近人类的准确性识别图像 [27]，[25]。它们还用于语音识别 [18]、自然语言处理 [1] 和玩游戏 [43]，[32]。

However, researchers have discovered that existing neural networks are vulnerable to attack. Szegedy et al. [46] first noticed the existence of adversarial examples in the image classification domain: it is possible to transform an image by a small amount and thereby change how the image is classified. Often, the total amount of change required can be so small as to be undetectable.

然而，研究人员发现现有的神经网络容易受到攻击。Szegedy 等人 [46] 首先注意到在图像分类领域存在对抗样本: 通过少量的变换可以改变图像的分类方式。通常，所需的总变化量可能小到不可检测。

The degree to which attackers can find adversarial examples limits the domains in which neural networks can be used. For example, if we use neural networks in self-driving cars, adversarial examples could allow an attacker to cause the car to take unwanted actions.

攻击者找到对抗样本的能力限制了神经网络可以使用的领域。例如，如果我们在自动驾驶汽车中使用神经网络，对抗样本可能允许攻击者使汽车采取不希望的行动。

The existence of adversarial examples has inspired research on how to harden neural networks against these kinds of attacks. Many early attempts to secure neural networks failed or provided only marginal robustness improvements [15], [2], [20], [42].

对抗样本的存在激发了关于如何增强神经网络抵御这些攻击的研究。许多早期的安全尝试失败了，或者仅提供了边际的鲁棒性改进 [15]、[2]、[20]、[42]。

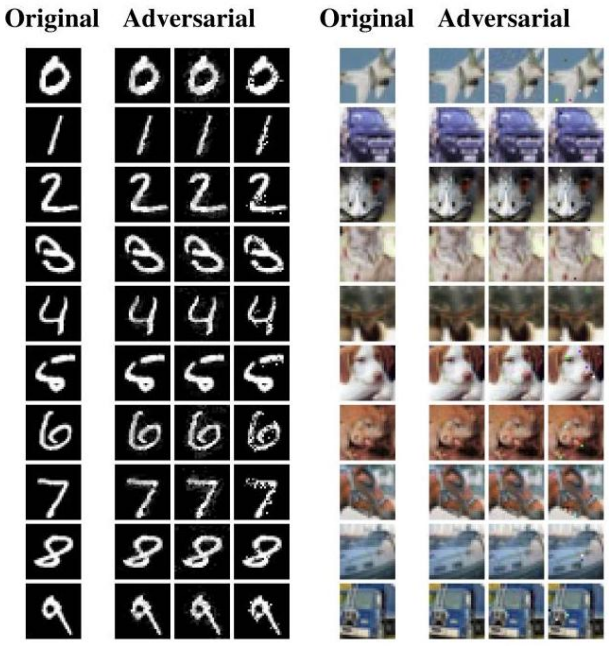


Fig. 1. An illustration of our attacks on a defensively distilled network. The leftmost column contains the starting image. The next three columns show adversarial examples generated by our L_2, L_∞ , and L_0 algorithms, misclassified instances share the same misclassified label of $l + 1$ (mod 10). Images were chosen as the first of their class from the test set.

图 1. 我们对防御性蒸馏网络的攻击示意图。最左列包含起始图像。接下来的三列展示了通过我们的 L_2, L_∞ 和 L_0 算法生成的对抗样本，错误分类的实例共享相同的错误分类标签 $l + 1$ (模 10)。图像是从测试集中选择的其类别的第一个图像。

Defensive distillation [39] is one such recent defense proposed for hardening neural networks against adversarial examples. Initial analysis proved to be very promising: defensive distillation defeats existing attack algorithms and reduces their success probability from 95% to 0.5% . Defensive distillation can be applied to any feed-forward neural network and only requires a single re-training step, and is currently one of the only defenses giving strong security guarantees against adversarial examples.

防御性蒸馏 [39] 是最近提出的一种增强神经网络抵御对抗样本的防御方法。初步分析证明非常有前景: 防御性蒸馏击败了现有的攻击算法，并将其成功概率从 95% 降低到 0.5% 。防御性蒸馏可以应用于任何前馈神经网络，仅需一次重新训练步骤，目前是为数不多的提供强安全保证以抵御对抗样本的防御之一。

In general, there are two different approaches one can take to evaluate the robustness of a neural network: attempt to prove a lower bound, or construct attacks that demonstrate an upper bound. The former approach, while sound, is substantially more difficult to implement in practice, and all attempts have required approximations [2], [21]. On the other hand, if the attacks used in the the latter approach are not sufficiently strong and fail often, the upper bound may not be useful.

一般来说，可以采取两种不同的方法来评估神经网络的鲁棒性: 尝试证明下界，或构建攻击以展示上界。前一种方法虽然合理，但在实践中实施起来要困难得多，所有尝试都需要近似 [2]，[21]。另一方面，如果后者方法中使用的攻击不够强大并且经常失败，那么上界可能没有用处。

In this paper we create a set of attacks that can be used to construct an upper bound on the robustness of neural networks. As a case study, we use these attacks to demonstrate that defensive distillation does not actually eliminate adversarial examples. We construct three new attacks (under three previously used distance metrics: L_0, L_2 , and L_∞) that succeed in finding adversarial examples for 100% of images on defensively distilled networks. While defensive distillation stops previously published attacks, it cannot resist the more powerful attack techniques we introduce in this paper.

在本文中，我们创建了一组攻击，可以用来构建神经网络鲁棒性的上界。作为案例研究，我们使用这些攻击来证明防御性蒸馏实际上并没有消除对抗样本。我们构建了三种新的攻击 (基于三种先前使用的距离度量: L_0, L_2 和 L_∞)，成功地为防御性蒸馏网络上的 100% 图像找到对抗样本。虽然防御性蒸馏能够阻止之前发布的攻击，但它无法抵御我们在本文中引入的更强大的攻击技术。

This case study illustrates the general need for better techniques to evaluate the robustness of neural networks: while distillation was shown to be secure against the current state-of-the-art attacks, it fails against our stronger attacks. Furthermore, when comparing our attacks against the current state-of-the-art on standard unsecured models, our methods generate adversarial examples with less total distortion in every case. We suggest that our attacks are a better baseline for evaluating candidate defenses: before placing any faith in a new possible defense, we suggest that designers at least check whether it can resist our attacks.

这个案例研究说明了评估神经网络鲁棒性的一般需求: 尽管蒸馏被证明对当前最先进的攻击是安全的，但它在面对我们更强的攻击时失败。此外，当将我们的攻击与当前最先进的标准无防御模型进行比较时，我们的方法在每种情况下生成的对抗样本的总失真更小。我们建议我们的攻击是评估候选防御的更好基线: 在对任何新的可能防御寄予信任之前，我们建议设计者至少检查它是否能够抵御我们的攻击。

We additionally propose using high-confidence adversarial examples to evaluate the robustness of defenses. Transferability [46], [11] is the well-known property that adversarial examples on one model are often also adversarial on another model. We demonstrate that adversarial examples from our attacks are transferable from the unsecured model to the defensively distilled (secured) model. In general, we argue that any defense must demonstrate it is able to break the transferability property.

我们还建议使用高置信度的对抗样本来评估防御的鲁棒性。可转移性 [46], [11] 是一个众所周知的特性，即一个模型上的对抗样本往往在另一个模型上也会是对抗样本。我们证明了我们的攻击生成的对抗样本可以从不安全模型转移到防御性蒸馏 (安全) 模型。一般而言，我们认为任何防御措施都必须证明其能够打破可转移性特性。

We evaluate our attacks on three standard datasets: MNIST [28], a digit-recognition task (0-9); CIFAR-10 [24], a small-image recognition task, also with 10 classes; and ImageNet [9], a large-image recognition task with 1000 classes.

我们在三个标准数据集上评估了我们的攻击:MNIST [28]，一个数字识别任务 (0-9)；CIFAR-10 [24]，一个小图像识别任务，同样有 10 个类别；以及 ImageNet [9]，一个具有 1000 个类别的大图像识别任务。

Figure 1 shows examples of adversarial examples our techniques generate on defensively distilled networks trained on the MNIST and CIFAR datasets.

图 1 展示了我们技术在 MNIST 和 CIFAR 数据集上训练的防御性蒸馏网络上生成的对抗样本示例。

In one extreme example for the ImageNet classification task, we can cause the Inception v3 [45] network to incorrectly classify images by changing only the lowest order bit of each pixel. Such changes are impossible to detect visually.

在 ImageNet 分类任务的一个极端示例中，我们可以通过仅改变每个像素的最低有效位来使 Inception v3 [45] 网络错误分类图像。这种变化在视觉上是无法检测的。

To enable others to more easily use our work to evaluate the robustness of other defenses, all of our adversarial example generation algorithms (along with code to train the models we use, to reproduce the results we present) are available online at http://nicholas.carlini.com/code/nn_robust_attacks.

为了使其他人更容易使用我们的工作来评估其他防御的鲁棒性，我们所有的对抗样本生成算法 (以及用于训练我们所使用模型的代码，以重现我们所展示结果的代码) 都可以在 http://nicholas.carlini.com/code/nn_robust_a
上在线获取。

This paper makes the following contributions:

本文做出了以下贡献:

- We introduce three new attacks for the L_0, L_2 , and L_∞ distance metrics. Our attacks are significantly more effective than previous approaches. Our L_0 attack is the first published attack that can cause targeted misclassification on the ImageNet dataset.

- 我们为 L_0, L_2 和 L_∞ 距离度量引入了三种新的攻击。我们的攻击比以前的方法显著更有效。我们的 L_0 攻击是第一个已发布的攻击，能够在 ImageNet 数据集上造成有针对性的错误分类。

- We apply these attacks to defensive distillation and discover that distillation provides little security benefit over un-distilled networks.

- 我们将这些攻击应用于防御性蒸馏，并发现蒸馏对未蒸馏网络提供的安全性收益很小。

- We propose using high-confidence adversarial examples in a simple transferability test to evaluate defenses, and show this test breaks defensive distillation.

- 我们建议在一个简单的可转移性测试中使用高置信度的对抗样本来评估防御，并展示该测试破坏了防御蒸馏。

- We systematically evaluate the choice of the objective function for finding adversarial examples, and show that the choice can dramatically impact the efficacy of an attack.

- 我们系统地评估了寻找对抗样本的目标函数的选择，并展示该选择可以显著影响攻击的有效性。

II. BACKGROUND

II. 背景

A. Threat Model

A. 威胁模型

Machine learning is being used in an increasing array of settings to make potentially security critical decisions: self-driving cars [3], [4], drones [10], robots [33], [22], anomaly detection [6], malware classification [8], [40], [48], speech recognition and recognition of voice commands [17], [13], NLP [1], and many more. Consequently, understanding the security properties of deep learning has become a crucial question in this area. The extent to which we can construct adversarial examples influences the settings in which we may want to (or not want to) use neural networks.

机器学习在越来越多的场景中被用于做出潜在的安全关键决策: 自动驾驶汽车 [3]、无人机 [10]、机器人 [33]、[22]、异常检测 [6]、恶意软件分类 [8]、[40]、[48]、语音识别和语音命令识别 [17]、[13]、自然语言处理 [1] 等等。因此，理解深度学习的安全属性已成为该领域的一个关键问题。我们能够构造对抗样本的程度影响了我们可能希望 (或不希望) 使用神经网络的场景。

In the speech recognition domain, recent work has shown [5] it is possible to generate audio that sounds like speech to machine learning algorithms but not to humans. This can be used to control user's devices without their knowledge. For example, by playing a video with a hidden voice command, it may be possible to cause a smart phone to visit a malicious webpage to cause a drive-by download. This work focused on conventional techniques (Gaussian Mixture Models and Hidden Markov Models), but as speech recognition is increasingly using neural networks, the study of adversarial examples becomes relevant in this domain. [1]

在语音识别领域，最近的研究表明 [5]，可以生成对机器学习算法听起来像语音但对人类不可听的音频。这可以在用户不知情的情况下控制用户的设备。例如，通过播放带有隐藏语音命令的视频，可能导致智能手机访问恶意网页，从而引发驱动下载。这项工作专注于传统技术 (高斯混合模型和隐马尔可夫模型)，但随着语音识别越来越多地使用神经网络，对抗样本的研究在这一领域变得相关。[1]

In the space of malware classification, the existence of adversarial examples not only limits their potential application settings, but entirely defeats its purpose: an adversary who is able to make only slight modifications to a malware file that cause it to remain malware, but become classified as benign, has entirely defeated the malware classifier [8], [14].

在恶意软件分类领域，对抗样本的存在不仅限制了其潜在的应用场景，还完全违背了其目的: 一个能够对恶意软件文件进行轻微修改，使其仍然是恶意软件但被分类为良性的对手，完全击败了恶意软件分类器 [8]、[14]。

Turning back to the threat to self-driving cars introduced earlier, this is not an unrealistic attack: it has been shown that adversarial examples are possible in the physical world [26] after taking pictures of them.

回到之前提到的自驾车面临的威胁，这并不是一个不现实的攻击: 研究表明，在拍摄它们之后，物理世界中确实存在对抗样本 [26]。

The key question then becomes exactly how much distortion we must add to cause the classification to change. In each domain, the distance metric that we must use is different. In the space of images, which we focus on in this paper, we rely on previous work that suggests that various L_p norms are reasonable approximations of human perceptual distance (see Section II-D for more information).

关键问题是我们必须添加多少扭曲才能导致分类发生变化。在每个领域中，我们必须使用的距离度量是不同的。在我们在本文中关注的图像空间中，我们依赖于之前的研究，该研究表明各种 L_p 范数是人类感知距离的合理近似 (更多信息请参见第 II-D 节)。

We assume in this paper that the adversary has complete access to a neural network, including the architecture and all paramaters, and can use this in a white-box manner. This is a conservative and realistic assumption: prior work has shown it is possible to train a substitute model given black-box access to a target model, and by attacking the substitute model, we can then transfer these attacks to the target model. [37]

我们在本文中假设对手完全访问神经网络，包括其架构和所有参数，并可以以白盒方式使用它。这是一个保守且现实的假设: 先前的研究表明，在获得目标模型的黑盒访问权限的情况下，可以训练替代模型，并通过攻击替代模型，我们可以将这些攻击转移到目标模型上 [37]。

Given these threats, there have been various attempts [15], [2], [20], [42], [39] at constructing defenses that increase the robustness of a neural network, defined as a measure of how easy it is to find adversarial examples that are close to their original input.

鉴于这些威胁，已经有各种尝试 [15]、[2]、[20]、[42]、[39] 来构建增强神经网络鲁棒性的防御，鲁棒性被定义为找到与原始输入接近的对抗样本的难易程度。

In this paper we study one of these, distillation as a defense [39], that hopes to secure an arbitrary neural network. This type of defensive distillation was shown to make generating adversarial examples nearly impossible for existing attack techniques [39]. We find that although the current state-of-the-art fails to find adversarial examples for defensively distilled networks, the stronger attacks we develop in this paper are able to construct adversarial examples.

在本文中，我们研究其中一种防御方法，即蒸馏 [39]，希望能够保护任意神经网络。这种防御性蒸馏被证明使得现有攻击技术几乎无法生成对抗样本 [39]。我们发现，尽管当前的最先进技术无法为防御性蒸馏网络找到对抗样本，但我们在本文中开发的更强攻击能够构造对抗样本。

B. Neural Networks and Notation

B. 神经网络与符号

A neural network is a function $F(x) = y$ that accepts an input $x \in \mathbb{R}^n$ and produces an output $y \in \mathbb{R}^m$. The model F also implicitly depends on some model parameters θ ; in our work the model is fixed, so for convenience we don't show the dependence on θ .

神经网络是一个函数 $F(x) = y$ ，它接受输入 $x \in \mathbb{R}^n$ 并产生输出 $y \in \mathbb{R}^m$ 。该模型 F 还隐含地依赖于一些模型参数 θ ；在我们的工作中，模型是固定的，因此为了方便，我们不显示对 θ 的依赖。

In this paper we focus on neural networks used as an m - class classifier. The output of the network is computed using the softmax function, which ensures that the output vector y satisfies $0 \le y_i \le 1$ and $y_1 + \cdots + y_m = 1$. The output vector y is thus treated as a probability distribution, i.e., y_i is treated as the probability that input x has class i . The classifier assigns the label $C(x) = \arg\max_i F(x)_i$ to the input x . Let $C^*(x)$ be the correct label of x . The inputs to the softmax function are called logits.

在本文中，我们专注于用作 m 类分类器的神经网络。网络的输出通过 softmax 函数计算，该函数确保输出向量 y 满足 $0 \le y_i \le 1$ 和 $y_1 + \cdots + y_m = 1$ 。因此，输出向量 y 被视为概率分布，即 y_i 被视为输入 x 属于类 i 的概率。分类器将标签 $C(x) = \arg\max_i F(x)_i$ 分配给输入 x 。设 $C^*(x)$ 为 x 的正确标签。softmax 函数的输入称为 logits。

We use the notation from Papernot et al. [39]: define F to be the full neural network including the softmax function, $Z(x) = z$ to be the output of all layers except the softmax (so z are the logits), and

我们使用 Papernot 等人的符号 [39]: 定义 F 为包括 softmax 函数的完整神经网络，$Z(x) = z$ 为除 softmax 之外的所有层的输出 (因此 z 是 logits)，并且

$$F(x) = \operatorname{softmax}(Z(x)) = y.$$

A neural network typically [2] consists of layers

神经网络通常 [2] 由层组成

[1] Strictly speaking, hidden voice commands are not adversarial examples because they are not similar to the original input [5].

[1] 严格来说，隐藏的语音命令并不是对抗样本，因为它们与原始输入并不相似 [5]。

$$F = \text{softmax} \circ F_n \circ F_{n-1} \circ \cdots \circ F_1$$

where
其中

$$F_i(x) = \sigma(\theta_i \cdot x) + \widehat{\theta}_i$$

for some non-linear activation function σ , some matrix θ_i of model weights, and some vector $\widehat{\theta}_i$ of model biases. Together θ and $\widehat{\theta}$ make up the model parameters. Common choices of σ are tanh [31], sigmoid, ReLU [29], or ELU [7]. In this paper we focus primarily on networks that use a ReLU activation function, as it currently is the most widely used activation function [45], [44], [31], [39].

对于某些非线性激活函数 σ ，某个模型权重的矩阵 θ_i ，以及某个模型偏置的向量 $\widehat{\theta}_i$ 。θ 和 $\widehat{\theta}$ 一起构成模型参数。σ 的常见选择包括 tanh [31]、sigmoid、ReLU [29] 或 ELU [7]。在本文中，我们主要关注使用 ReLU 激活函数的网络，因为它目前是最广泛使用的激活函数 [45]、[44]、[31]、[39]。

We use image classification as our primary evaluation domain. An $h \times w$ -pixel grey-scale image is a two-dimensional vector $x \in \mathbb{R}^{hw}$, where x_i denotes the intensity of pixel i and is scaled to be in the range $[0,1]$. A color RGB image is a three-dimensional vector $x \in \mathbb{R}^{3hw}$. We do not convert RGB images to HSV, HSL, or other cylindrical coordinate representations of color images: the neural networks act on raw pixel values.

我们使用图像分类作为主要评估领域。一个 $h \times w$ 像素灰度图像是一个二维向量 $x \in \mathbb{R}^{hw}$ ，其中 x_i 表示像素 i 的强度，并被缩放到范围 $[0,1]$ 内。一个彩色 RGB 图像是一个三维向量 $x \in \mathbb{R}^{3hw}$ 。我们不将 RGB 图像转换为 HSV、HSL 或其他颜色图像的圆柱坐标表示: 神经网络直接作用于原始像素值。

C. Adversarial Examples

C. 对抗样本

Szegedy et al. [46] first pointed out the existence of adversarial examples: given a valid input x and a target $t \neq C^*(x)$, it is often possible to find a similar input x' such that $C(x') = t$ yet x, x' are close according to some distance metric. An example x' with this property is known as a targeted adversarial example.

Szegedy 等人 [46] 首先指出了对抗样本的存在: 给定一个有效输入 x 和一个目标 $t \neq C^*(x)$ ，通常可以找到一个相似的输入 x' ，使得 $C(x') = t$ 和 x, x' 根据某种距离度量是接近的。具有此属性的示例 x' 被称为有针对性的对抗样本。

A less powerful attack also discussed in the literature instead asks for untargeted adversarial examples: instead of classifying x as a given target class, we only search for an input x' so that $C(x') \neq C^*(x)$ and x, x' are close. Untargeted attacks are strictly less powerful than targeted attacks and we do not consider them in this paper. [3]

文献中还讨论了一种较弱的攻击，要求寻找无目标对抗样本: 我们不将 x 分类为给定的目标类，而只是寻找一个输入 x' ，使得 $C(x') \neq C^*(x)$ 和 x, x' 是接近的。无目标攻击的能力严格低于有目标攻击，我们在本文中不考虑它们。[3]

Instead, we consider three different approaches for how to choose the target class, in a targeted attack:
相反，我们考虑三种不同的方法来选择目标类，以进行有针对性的攻击:

- Average Case: select the target class uniformly at random among the labels that are not the correct label.

- 平均情况: 在不是正确标签的标签中均匀随机选择目标类。

- Best Case: perform the attack against all incorrect classes, and report the target class that was least difficult to attack.

- 最佳情况: 对所有错误类别进行攻击，并报告最不难攻击的目标类。

- Worst Case: perform the attack against all incorrect classes, and report the target class that was most difficult to attack.

- 最差情况: 对所有错误类别进行攻击，并报告最难攻击的目标类。

In all of our evaluations we perform all three types of attacks: best-case, average-case, and worst-case. Notice that if a classifier is only accurate 80% of the time, then the best case attack will require a change of 0 in 20% of cases.

在我们所有的评估中，我们执行三种类型的攻击: 最佳情况、平均情况和最坏情况。请注意，如果分类器的准确率仅为 80% ，那么最佳情况攻击将在 20% 的情况下需要改变 0。

On ImageNet, we approximate the best-case and worst-case attack by sampling 100 random target classes out of the 1,000 possible for efficiency reasons.

在 ImageNet 上，为了效率，我们从 1,000 个可能的目标类别中随机抽取 100 个目标类别来近似最佳情况和最坏情况攻击。

D. Distance Metrics

D. 距离度量

In our definition of adversarial examples, we require use of a distance metric to quantify similarity. There are three widely-used distance metrics in the literature for generating adversarial examples, all of which are L_p norms.

在我们对对抗样本的定义中，我们要求使用距离度量来量化相似性。文献中有三种广泛使用的距离度量用于生成对抗样本，所有这些都是 L_p 范数。

The L_p distance is written $\left\|x-x'\right\|_p$, where the p -norm $\|\cdot\|_p$ is defined as

L_p 距离表示为 $\left\|x-x'\right\|_p$ ，其中 p 范数 $\|\cdot\|_p$ 定义为

$$\| v\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{\frac{1}{p}}.$$

In more detail:

更详细地说:

1) L_0 distance measures the number of coordinates i such that $x_i \neq x'_i$. Thus, the L_0 distance corresponds to the number of pixels that have been altered in an image. [4] Papernot et al. argue for the use of the L_0 distance metric, and it is the primary distance metric under which defensive distillation's security is argued [39].

1) L_0 距离测量坐标的数量 i ，使得 $x_i \neq x'_i$ 。因此，L_0 距离对应于图像中被改变的像素数量。[4] Papernot 等人主张使用 L_0 距离度量，它是防御蒸馏安全性论证的主要距离度量 [39]。

2) L_2 distance measures the standard Euclidean (root-mean-square) distance between x and x' . The L_2 distance can remain small when there are many small changes to many pixels.

2) L_2 距离测量标准欧几里得 (均方根) 距离在 x 和 x' 之间。当有许多小的变化发生在许多像素时，L_2 距离可以保持较小。

This distance metric was used in the initial adversarial example work [46].

该距离度量在最初的对抗示例工作中被使用 [46]。

3) L_∞ distance measures the maximum change to any of the coordinates:

3) L_∞ 距离测量任何坐标的最大变化:

$$\left\|x-x'\right\|_\infty = \max\left(\left|x_1-x'_1\right|, \ldots, \left|x_n-x'_n\right|\right).$$

For images, we can imagine there is a maximum budget, and each pixel is allowed to be changed by up to this limit, with no limit on the number of pixels that are modified.

对于图像，我们可以想象有一个最大预算，每个像素允许改变到这个限制，且对修改的像素数量没有限制。

---

[2] Most simple networks have this simple linear structure, however other more sophisticated networks have more complicated structures (e.g., ResNet [16] and Inception [45]). The network architecture does not impact our attacks.

[2] 大多数简单网络具有这种简单的线性结构，然而其他更复杂的网络具有更复杂的结构 (例如，ResNet [16] 和 Inception [45])。网络架构不会影响我们的攻击。

[3] An untargeted attack is simply a more efficient (and often less accurate) method of running a targeted attack for each target and taking the closest. In this paper we focus on identifying the most accurate attacks, and do not consider untargeted attacks.

[3] 非目标攻击仅仅是对每个目标运行目标攻击的一种更高效 (且通常准确性较低) 的方法，并取最近的结果。在本文中，我们专注于识别最准确的攻击，而不考虑非目标攻击。

Goodfellow et al. argue that L_∞ is the optimal distance metric to use [47] and in a follow-up paper Papernot et al. argue distillation is secure under this distance metric

Goodfellow 等人认为 L_∞ 是最佳的距离度量 [47]，而在后续论文中，Papernot 等人则认为在该距离度量下蒸馏是安全的。[36].

No distance metric is a perfect measure of human perceptual similarity, and we pass no judgement on exactly which distance metric is optimal. We believe constructing and evaluating a good distance metric is an important research question we leave to future work.

没有任何距离度量能够完美地衡量人类的感知相似性，我们不对究竟哪个距离度量是最佳的做出判断。我们认为构建和评估一个良好的距离度量是一个重要的研究问题，留待未来的工作。

However, since most existing work has picked one of these three distance metrics, and since defensive distillation argued security against two of these, we too use these distance metrics and construct attacks that perform superior to the state-of-the-art for each of these distance metrics.

然而，由于大多数现有工作选择了这三种距离度量中的一种，并且由于防御性蒸馏对其中两种进行了安全性论证，我们也使用这些距离度量，并构建了在每种距离度量下表现优于现有技术的攻击。

When reporting all numbers in this paper, we report using the distance metric as defined above, on the range $[0, 1]$. (That is, changing a pixel in a greyscale image from full-on to full-off will result in L_2 change of 1.0 and a L_∞ change of 1.0, not 255.)

在本文中报告所有数字时，我们使用上述定义的距离度量，在范围 $[0, 1]$ 上报告。(也就是说，将灰度图像中的一个像素从全开更改为全关将导致 L_2 变化为 1.0 和 L_∞ 变化为 1.0，而不是 255。)

E. Defensive Distillation

E. 防御性蒸馏

We briefly provide a high-level overview of defensive distillation. We provide a complete description later in Section VIII.

我们简要提供防御性蒸馏的高层次概述。我们将在第八节中提供完整的描述。

To defensively distill a neural network, begin by first training a network with identical architecture on the training data in a standard manner. When we compute the softmax while training this network, replace it with a more-smooth version of the softmax (by dividing the logits by some constant T). At the end of training, generate the soft training labels by evaluating this network on each of the training instances and taking the output labels of the network.

为了防御性蒸馏神经网络，首先在训练数据上以标准方式训练一个具有相同架构的网络。当我们在训练这个网络时计算 softmax 时，用更平滑的 softmax 版本替代它 (通过将 logits 除以某个常数 T)。在训练结束时，通过在每个训练实例上评估这个网络并获取网络的输出标签来生成软训练标签。

Then, throw out the first network and use only the soft training labels. With those, train a second network where instead of training it on the original training labels, use the soft labels. This trains the second model to behave like the first model, and the soft labels convey additional hidden knowledge learned by the first model.

然后，丢弃第一个网络，仅使用软训练标签。利用这些标签，训练第二个网络，而不是在原始训练标签上进行训练，而是使用软标签。这使得第二个模型的行为类似于第一个模型，并且软标签传达了第一个模型所学到的额外隐含知识。

The key insight here is that by training to match the first network, we will hopefully avoid over-fitting against any of the training data. If the reason that neural networks exist is because neural networks are highly non-linear and have ”blind spots” [46] where adversarial examples lie, then preventing this type of over-fitting might remove those blind spots.

这里的关键见解是，通过训练以匹配第一个网络，我们希望能够避免对任何训练数据的过拟合。如果神经网络存在的原因是因为神经网络高度非线性并且存在“盲点”[46]，而对抗样本正好位于这些盲点上，那么防止这种类型的过拟合可能会消除这些盲点。

In fact, as we will see later, defensive distillation does not remove adversarial examples. One potential reason this may occur is that others [11] have argued the reason adversarial examples exist is not due to blind spots in a highly non-linear neural network, but due only to the locally-linear nature of neural networks. This so-called linearity hypothesis appears to be true [47], and under this explanation it is perhaps less surprising that distillation does not increase the robustness of neural networks.

实际上，正如我们稍后将看到的，防御性蒸馏并不能消除对抗样本。发生这种情况的一个潜在原因是，其他人 [11] 认为对抗样本存在的原因并不是由于高度非线性神经网络中的盲点，而仅仅是由于神经网络

的局部线性特性。这种所谓的线性假设似乎是正确的 [47]，在这种解释下，蒸馏未能提高神经网络的鲁棒性或许就不那么令人惊讶了。

F. Organization

F. 组织

The remainder of this paper is structured as follows. In the next section, we survey existing attacks that have been proposed in the literature for generating adversarial examples, for the L_2, L_∞ , and L_0 distance metrics. We then describe our attack algorithms that target the same three distance metrics and provide superior results to the prior work. Having developed these attacks, we review defensive distillation in more detail and discuss why the existing attacks fail to find adversarial examples on defensively distilled networks. Finally, we attack defensive distillation with our new algorithms and show that it provides only limited value.

本文的其余部分结构如下。在下一节中，我们调查了文献中提出的用于生成对抗样本的现有攻击，针对 L_2, L_∞ 和 L_0 距离度量。然后，我们描述了针对同三种距离度量的攻击算法，并提供了优于之前工作的结果。在开发了这些攻击后，我们更详细地回顾了防御蒸馏，并讨论了为什么现有攻击未能在防御蒸馏网络上找到对抗样本。最后，我们用新的算法攻击防御蒸馏，并表明它仅提供有限的价值。

III. ATTACK ALGORITHMS

III. 攻击算法

A. L-BFGS

A. L-BFGS

Szegedy et al. [46] generated adversarial examples using box-constrained L-BFGS. Given an image x , their method finds a different image x' that is similar to x under L_2 distance, yet is labeled differently by the classifier. They model the problem as a constrained minimization problem:

Szegedy 等人 [46] 使用盒约束 L-BFGS 生成对抗样本。给定一幅图像 x ，他们的方法找到一幅与 x 在 L_2 距离下相似但被分类器标记为不同的图像 x' 。他们将问题建模为一个约束最小化问题:

$$\text{minimize } \left\|x-x'\right\|_2^2$$

$$\text{such that } C\left(x'\right)=l$$

$$x' \in [0,1]^n$$

This problem can be very difficult to solve, however, so Szegedy et al. instead solve the following problem:

然而，这个问题可能非常难以解决，因此 Szegedy 等人选择解决以下问题:

$$\text{minimize } c \cdot \left\|x-x'\right\|_2^2+\operatorname{loss}_{F,l}\left(x'\right)$$

such that $x' \in [0,1]^n$

使得 $x' \in [0,1]^n$

where $\operatorname{loss}_{F,l}$ is a function mapping an image to a positive real number. One common loss function to use is cross-entropy. Line search is performed to find the constant $c>0$ that yields an adversarial example of minimum distance: in other words, we repeatedly solve this optimization problem for multiple values of c , adaptively updating c using bisection search or any other method for one-dimensional optimization.

其中 $\operatorname{loss}_{F,l}$ 是一个将图像映射到正实数的函数。一个常用的损失函数是交叉熵。进行线搜索以找到常数 $c>0$ ，该常数产生最小距离的对抗样本: 换句话说，我们对多个 c 值反复解决这个优化问题，使用二分搜索或任何其他一维优化方法自适应更新 c 。

B. Fast Gradient Sign

B. 快速梯度符号

The fast gradient sign [11] method has two key differences from the L-BFGS method: first, it is optimized for the L_∞ distance metric, and second, it is designed primarily to be fast instead of producing very close adversarial examples. Given an image x the fast gradient sign method sets

快速梯度符号 [11] 方法与 L-BFGS 方法有两个关键区别: 首先，它针对 L_∞ 距离度量进行了优化，其次，它主要设计为快速，而不是产生非常接近的对抗样本。给定一幅图像 x ，快速梯度符号方法设置

$$x' = x - \epsilon \cdot \operatorname{sign}\left(\nabla \operatorname{loss}_{F,t}(x)\right),$$

where ϵ is chosen to be sufficiently small so as to be undetectable, and t is the target label. Intuitively, for each pixel, the fast gradient sign method uses the gradient of the loss function to determine in which direction the pixel's intensity should be changed (whether it should be increased or decreased) to minimize the loss function; then, it shifts all pixels simultaneously.

其中 ϵ 被选择为足够小以至于不可检测，而 t 是目标标签。直观地，对于每个像素，快速梯度符号方法使用损失函数的梯度来确定像素的强度应该朝哪个方向变化 (是增加还是减少) 以最小化损失函数；然后，它同时移动所有像素。

It is important to note that the fast gradient sign attack was designed to be fast, rather than optimal. It is not meant to produce the minimal adversarial perturbations.

重要的是要注意，快速梯度符号攻击的设计目的是快速，而不是最优。它并不旨在产生最小的对抗扰动。

Iterative Gradient Sign: Kurakin et al. introduce a simple refinement of the fast gradient sign method [26] where instead of taking a single step of size ϵ in the direction of the gradient-sign, multiple smaller steps α are taken, and the result is clipped by the same ϵ . Specifically, begin by setting

迭代梯度符号:Kurakin 等人引入了快速梯度符号方法 [26] 的一个简单改进，其中不是在梯度符号的方向上采取单一步长 ϵ ，而是采取多个较小的步骤 α ，并且结果被限制在相同的 ϵ 范围内。具体来说，首先设置

$$x'_0 = 0$$

and then on each iteration

然后在每次迭代中

$$x'_i = x'_{i-1} - \operatorname{clip}_\epsilon\left(\alpha \cdot \operatorname{sign}\left(\nabla \operatorname{loss}_{F,t}\left(x'_{i-1}\right)\right)\right)$$

Iterative gradient sign was found to produce superior results to fast gradient sign [26].

迭代梯度符号被发现产生的结果优于快速梯度符号 [26]。

C. JSMA

C. JSMA

Papernot et al. introduced an attack optimized under L_0 distance [38] known as the Jacobian-based Saliency Map Attack (JSMA). We give a brief summary of their attack algorithm; for a complete description and motivation, we encourage the reader to read their original paper [38].

Papernot 等人提出了一种基于 L_0 距离 [38] 优化的攻击，称为雅可比基础显著性图攻击 (JSMA)。我们简要总结了他们的攻击算法；有关完整的描述和动机，我们鼓励读者阅读他们的原始论文 [38]。

At a high level, the attack is a greedy algorithm that picks pixels to modify one at a time, increasing the target classification on each iteration. They use the gradient $\nabla Z(x)_l$ to compute a saliency map, which models the impact each pixel has on the resulting classification. A large value indicates that

[4] In RGB images, there are three channels that each can change. We count the number of pixels that are different, where two pixels are considered different if any of the three colors are different. We do not consider a distance metric where an attacker can change one color plane but not another meaningful. We relax this requirement when comparing to other L_0 attacks that do not make this assumption to provide for a fair comparison.

[4] 在 RGB 图像中，有三个通道可以变化。我们计算不同的像素数量，当三个颜色中有任何一个不同的时候，两个像素被视为不同。我们不考虑攻击者可以改变一个颜色通道而不改变另一个的距离度量。我们在与其他 L_0 攻击进行比较时放宽了这一要求，以提供公平的比较。

changing it will significantly increase the likelihood of the model labeling the image as the target class l . Given the saliency map, it picks the most important pixel and modify it to increase the likelihood of class l . This is repeated until either more than a set threshold of pixels are modified which makes the attack detectable, or it succeeds in changing the classification.

从高层次来看，该攻击是一种贪婪算法，逐个选择要修改的像素，在每次迭代中增加目标分类。他们使用梯度 $\nabla Z(x)_l$ 来计算显著性图，该图建模每个像素对最终分类的影响。较大的值表明，改变它将显著增加模型将图像标记为目标类别 l 的可能性。给定显著性图，它选择最重要的像素并进行修改，以增加类别 l 的可能性。这个过程重复进行，直到修改的像素数量超过设定的阈值，使得攻击可被检测，或者成功改变分类。

In more detail, we begin by defining the saliency map in terms of a pair of pixels p, q . Define

更详细地说，我们首先根据一对像素 p, q 定义显著性图。定义

$$\alpha_{pq} = \sum_{i \in \{p,q\}} \frac{\partial Z(x)_t}{\partial x_i}$$

$$\beta_{pq} = \left(\sum_{i \in \{p,q\}} \sum_j \frac{\partial Z(x)_j}{\partial x_i} \right) - \alpha_{pq}$$

so that α_{pq} represents how much changing both pixels p and q will change the target classification, and β_{pq} represents how much changing p and q will change all other outputs. Then the algorithm picks

使得 α_{pq} 表示改变像素 p 和 q 将如何改变目标分类，而 β_{pq} 表示改变 p 和 q 将如何改变所有其他输出。然后算法选择

$$(p^*, q^*) = \arg\max_{(p,q)} (-\alpha_{pq} \cdot \beta_{pq}) \cdot (\alpha_{pq} > 0) \cdot (\beta_{pq} < 0)$$

so that $\alpha_{pq} > 0$ (the target class is more likely), $\beta_{pq} < 0$ (the other classes become less likely), and $-\alpha_{pq} \cdot \beta_{pq}$ is largest.

使得 $\alpha_{pq} > 0$ (目标类别更可能)，$\beta_{pq} < 0$ (其他类别变得不太可能)，并且 $-\alpha_{pq} \cdot \beta_{pq}$ 是最大的。

Notice that JSMA uses the output of the second-to-last layer Z , the logits, in the calculation of the gradient: the output of the softmax F is not used. We refer to this as the JSMA-Z attack.

注意，JSMA 在计算梯度时使用倒数第二层的输出 Z ，即 logits，而不使用 softmax 的输出 F 。我们将其称为 JSMA-Z 攻击。

However, when the authors apply this attack to their defensively distilled networks, they modify the attack so it uses F instead of Z . In other words, their computation uses the output of the softmax(F)instead of the logits(Z). We refer to this modification as the JSMA-F attack. [5]

然而，当作者将这种攻击应用于他们防御性蒸馏的网络时，他们修改了攻击，使其使用 F 而不是 Z 。换句话说，他们的计算使用的是 softmax(F) 的输出，而不是 logits(Z)。我们将这种修改称为 JSMA-F 攻击。[5]

When an image has multiple color channels (e.g., RGB), this attack considers the L_0 difference to be 1 for each color channel changed independently (so that if all three color channels of one pixel change change, the L_0 norm would be 3). While we do not believe this is a meaningful threat model, when comparing to this attack, we evaluate under both models.

当一幅图像具有多个颜色通道 (例如，RGB) 时，这种攻击将每个独立改变的颜色通道的 L_0 差异视为 1(因此如果一个像素的三个颜色通道都发生变化，L_0 范数将为 3)。虽然我们不认为这是一个有意义的威胁模型，但在与这种攻击进行比较时，我们在这两种模型下进行评估。

D. Deepfool

D. Deepfool

Deepfool [34] is an untargeted attack technique optimized for the L_2 distance metric. It is efficient and produces closer adversarial examples than the L-BFGS approach discussed earlier.

Deepfool [34] 是一种针对 L_2 距离度量优化的无目标攻击技术。它高效且生成的对抗样本比之前讨论的 L-BFGS 方法更接近。

The authors construct Deepfool by imagining that the neural networks are totally linear, with a hyperplane separating each class from another. From this, they analytically derive the optimal solution to this simplified problem, and construct the adversarial example.

作者通过设想神经网络是完全线性的来构建 Deepfool，假设有一个超平面将每个类别与其他类别分开。基于此，他们分析性地推导出这个简化问题的最优解，并构建对抗样本。

Then, since neural networks are not actually linear, they take a step towards that solution, and repeat the process a second time. The search terminates when a true adversarial example is found.

然后，由于神经网络实际上并不是线性的，他们朝着该解迈出一步，并重复第二次该过程。当找到一个真正的对抗样本时，搜索终止。

The exact formulation used is rather sophisticated; interested readers should refer to the original work [34].

使用的确切公式相当复杂；感兴趣的读者应参考原始工作 [34]。

IV. EXPERIMENTAL SETUP

IV. 实验设置

Before we develop our attack algorithms to break distillation, we describe how we train the models on which we will evaluate our attacks.

在我们开发攻击算法以破解蒸馏之前，我们描述如何训练我们将评估攻击的模型。

[5] We verified this via personal communication with the authors.

[5] 我们通过与作者的个人沟通验证了这一点。

Layer Type	MNIST Model	CIFAR Model
Convolution + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
Convolution + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
Max Pooling	2×2	2×2
Convolution + ReLU	$3{\times}3{\times}64$	$3 \times 3 \times 128$
Convolution + ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$
Max Pooling	2×2	2×2
Fully Connected + ReLU	200	256
Fully Connected + ReLU	200	256
Softmax	10	10

层类型	MNIST 模型	CIFAR 模型
卷积 + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
卷积 + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
最大池化	2×2	2×2
卷积 + ReLU	$3{\times}3{\times}64$	$3 \times 3 \times 128$
卷积 + ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$
最大池化	2×2	2×2
全连接 + ReLU	200	256
全连接 + ReLU	200	256
Softmax	10	10

TABLE I

MODEL ARCHITECTURES FOR THE MNIST AND CIFAR MODELS. THIS ARCHITECTURE IS IDENTICAL TO THAT OF THE ORIGINAL DEFENSIVE DISTILLATION WORK. [39]

MNIST 和 CIFAR 模型的模型架构。该架构与原始防御性蒸馏工作的架构相同。[39]

Parameter	MNIST Model	CIFAR Model
Learning Rate	0.1	0.01 (decay 0.5)
Momentum	0.9	0.9 (decay 0.5)
Delay Rate	-	10 epochs
Dropout	0.5	0.5
Batch Size	128	128
Epochs	50	50

参数	MNIST 模型	CIFAR 模型
学习率	0.1	0.01 (衰减 0.5)
动量	0.9	0.9 (衰减 0.5)
延迟率	-	10 个周期
随机失活	0.5	0.5
批量大小	128	128
训练周期	50	50

TABLE II

MODEL PARAMETERS FOR THE MNIST AND CIFAR MODELS. THESE PARAMETERS ARE IDENTICAL TO THAT OF THE ORIGINAL DEFENSIVE DISTILLATION WORK. [39]

MNIST 和 CIFAR 模型的模型参数。这些参数与原始防御蒸馏工作的参数相同。[39]

We train two networks for the MNIST [28] and CIFAR-10 [24] classification tasks, and use one pre-trained network for the ImageNet classification task [41]. Our models and training approaches are identical to those presented in [39]. We achieve 99.5% accuracy on MNIST, comparable to the state of the art. On CIFAR-10, we achieve 80% accuracy, identical to the accuracy given in the distillation work. [6]

我们为 MNIST [28] 和 CIFAR-10 [24] 分类任务训练了两个网络，并使用一个预训练的网络进行 ImageNet 分类任务 [41]。我们的模型和训练方法与 [39] 中提出的完全相同。我们在 MNIST 上实现了 99.5% 的准确率，达到当前的最先进水平。在 CIFAR-10 上，我们实现了 80% 的准确率，与蒸馏工作中的准确率相同。[6]

MNIST and CIFAR-10. The model architecture is given in Table I and the hyperparameters selected in Table II. We use a momentum-based SGD optimizer during training.

MNIST 和 CIFAR-10。模型架构见表 I，所选超参数见表 II。我们在训练过程中使用基于动量的 SGD 优化器。

The CIFAR-10 model significantly overfits the training data even with dropout: we obtain a final training cross-entropy loss of 0.05 with accuracy 98% , compared to a validation loss of 1.2 with validation accuracy 80% . We do not alter the network by performing image augmentation or adding additional dropout as that was not done in [39].

CIFAR-10 模型即使在使用 dropout 的情况下也显著过拟合训练数据: 我们获得了最终训练交叉熵损失为 0.05，准确率为 98% ，而验证损失为 1.2，验证准确率为 80% 。我们没有通过进行图像增强或添加额外的 dropout 来改变网络，因为在 [39] 中没有这样做。

ImageNet. Along with considering MNIST and CIFAR, which are both relatively small datasets, we also consider the ImageNet dataset. Instead of training our own ImageNet model, we use the pre-trained Inception v3 network [45], which achieves 96% top- 5 accuracy (that is, the probability that the correct class is one of the five most likely as reported by the network is 96%). Inception takes images as $299 \times 299 \times 3$ dimensional vectors.

ImageNet。除了考虑 MNIST 和 CIFAR 这两个相对较小的数据集外，我们还考虑了 ImageNet 数据集。我们没有训练自己的 ImageNet 模型，而是使用预训练的 Inception v3 网络 [45]，该网络实现了 96% 的前 5 名准确率 (即，网络报告的正确类别是五个最可能类别之一的概率为 96%)。Inception 将图像视为 $299 \times 299 \times 3$ 维向量。

V.OUR APPROACH

V. 我们的方法

We now turn to our approach for constructing adversarial examples. To begin, we rely on the initial formulation of adversarial examples [46] and formally define the problem of finding an adversarial instance for an image x as follows:

现在我们转向构造对抗样本的方法。首先，我们依赖于对抗样本的初始定义 [46]，并正式定义为图像 x 寻找对抗实例的问题如下:

$$\text{minimize } \mathcal{D}(x, x+\delta)$$

$$\text{such that } C(x+\delta) = t$$

$$x+\delta \in [0,1]^n$$

where x is fixed, and the goal is to find δ that minimizes $\mathcal{D}(x, x+\delta)$. That is, we want to find some small change δ that we can make to an image x that will change its classification, but so that the result is still a valid image. Here $\mathcal{D}$ is some distance metric; for us, it will be either L_0, L_2 , or L_∞ as discussed earlier.

其中 x 是固定的，目标是找到 δ 使 $\mathcal{D}(x, x+\delta)$ 最小化。也就是说，我们希望找到一些小的变化 δ ，可以对图像 x 进行修改，从而改变其分类，但结果仍然是有效的图像。这里 $\mathcal{D}$ 是某种距离度量；对我们而言，它将是 L_0, L_2 或 L_∞ ，如前所述。

We solve this problem by formulating it as an appropriate optimization instance that can be solved by existing optimization algorithms. There are many possible ways to do this; we explore the space of formulations and empirically identify which ones lead to the most effective attacks.

我们通过将此问题表述为适当的优化实例来解决它，这可以通过现有的优化算法来解决。有许多可能的方法来做到这一点；我们探索公式的空间，并通过经验确定哪些公式导致最有效的攻击。

A. Objective Function

A. 目标函数

The above formulation is difficult for existing algorithms to solve directly, as the constraint $C(x+\delta)=t$ is highly non-linear. Therefore, we express it in a different form that is better suited for optimization. We define an objective function f such that $C(x+\delta)=t$ if and only if $f(x+\delta) \leq 0$. There are many possible choices for f :

上述公式对于现有算法直接求解是困难的，因为约束 $C(x+\delta)=t$ 是高度非线性的。因此，我们以更适合优化的不同形式来表达它。我们定义一个目标函数 f ，使得 $C(x+\delta)=t$ 当且仅当 $f(x+\delta) \leq 0$ 。对于 f 有许多可能的选择:

$$f_1(x') = -\mathrm{loss}_{F,t}(x') + 1$$

$$f_2(x') = \left(\max_{i \neq t}\left(F(x')_i\right) - F(x')_t\right)^+$$

$$f_3(x') = \mathrm{softplus}\left(\max_{i \neq t}\left(F(x')_i\right) - F(x')_t\right) - \log(2)$$

$$f_4(x') = \left(0.5 - F(x')_t\right)^+$$

$$f_5(x') = -\log\left(2F(x')_t - 2\right)$$

$$f_6(x') = \left(\max_{i \neq t}\left(Z(x')_i\right) - Z(x')_t\right)^+$$

$$f_7(x') = \mathrm{softplus}\left(\max_{i \neq t}\left(Z(x')_i\right) - Z(x')_t\right) - \log(2)$$

where s is the correct classification, $(e)^+$ is short-hand for $\max(e, 0)$, $\mathrm{softplus}(x) = \log(1+\exp(x))$, and $\mathrm{loss}_{F,s}(x)$ is the cross entropy loss for x .

其中 s 是正确的分类，$(e)^+$ 是 $\max(e, 0)$ 的简写，$\mathrm{softplus}(x) = \log(1+\exp(x))$ ，而 $\mathrm{loss}_{F,s}(x)$ 是 x 的交叉熵损失。

Notice that we have adjusted some of the above formula by adding a constant; we have done this only so that the function respects our definition. This does not impact the final result, as it just scales the minimization function.

注意，我们通过添加一个常数调整了上述公式的一部分；我们这样做只是为了使函数符合我们的定义。这不会影响最终结果，因为它只是缩放了最小化函数。

Now, instead of formulating the problem as

现在，我们不再将问题表述为

minimize $\mathcal{D}(x, x+\delta)$

最小化 $\mathcal{D}(x, x+\delta)$

such that $f(x+\delta) \leq 0$

使得 $f(x+\delta) \leq 0$

$$x+\delta \in [0,1]^n$$

we use the alternative formulation:
我们使用替代的表述:
minimize $\mathcal{D}(x, x+\delta) + c \cdot f(x+\delta)$
最小化 $\mathcal{D}(x, x+\delta) + c \cdot f(x+\delta)$
such that $x+\delta \in [0,1]^n$
使得 $x+\delta \in [0,1]^n$

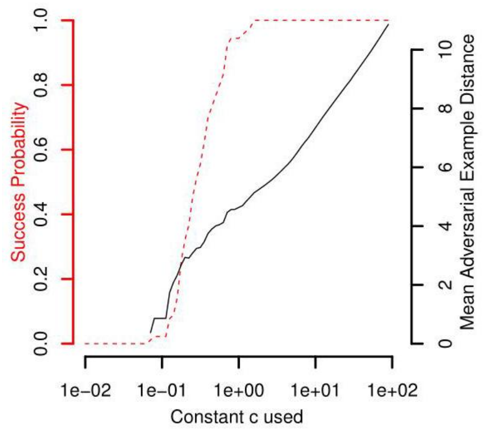


Fig. 2. Sensitivity on the constant c . We plot the L_2 distance of the adversarial example computed by gradient descent as a function of c , for objective function f_6 . When $c < .1$, the attack rarely succeeds. After $c > 1$, the attack becomes less effective, but always succeeds.
图 2. 对常数 c 的敏感性。我们绘制了通过梯度下降计算的对抗样本的 L_2 距离，作为 c 的函数，对于目标函数 f_6 。当 $c < .1$ 时，攻击很少成功。在 $c > 1$ 之后，攻击变得不那么有效，但总是成功。

where $c > 0$ is a suitably chosen constant. These two are equivalent, in the sense that there exists $c > 0$ such that the optimal solution to the latter matches the optimal solution to the former. After instantiating the distance metric $\mathcal{D}$ with an l_p norm, the problem becomes: given x , find δ that solves
其中 $c > 0$ 是一个适当选择的常数。这两者是等价的，因为存在 $c > 0$ 使得后者的最优解与前者的最优解相匹配。在用 l_p 范数实例化距离度量 $\mathcal{D}$ 后，问题变为: 给定 x ，找到 δ 以解决
minimize $\|\delta\|_p + c \cdot f(x+\delta)$
最小化 $\|\delta\|_p + c \cdot f(x+\delta)$
such that $x+\delta \in [0,1]^n$
使得 $x+\delta \in [0,1]^n$

Choosing the constant c .

选择常数 c 。

Empirically, we have found that often the best way to choose c is to use the smallest value of c for which the resulting solution x^* has $f(x^*) \leq 0$. This causes gradient descent to minimize both of the terms simultaneously instead of picking only one to optimize over first.
从经验上看，我们发现选择 c 的最佳方法通常是使用 c 的最小值，使得得到的解 x^* 具有 $f(x^*) \leq 0$ 。这使得梯度下降同时最小化两个项，而不是仅选择一个进行优化。

We verify this by running our f_6 formulation (which we found most effective) for values of c spaced uniformly (on a log scale) from $c = 0.01$ to $c = 100$ on the MNIST dataset. We plot this line in Figure 2. [7]

[6] This is compared to the state-of-the-art result of 95% [12],[44],[31]. However, in order to provide the most accurate comparison to the original work, we feel it is important to reproduce their model architectures.
这与 95% 的最先进结果相比 [12],[44],[31]。然而，为了提供与原始工作的最准确比较，我们认为重现他们的模型架构是重要的。

我们通过在 MNIST 数据集上均匀间隔 (在对数尺度上) 从 $c = 0.01$ 到 $c = 100$ 的 c 值运行我们找到的最有效的 f_6 公式来验证这一点。我们在图 2 中绘制了这条线。[7]

Further, we have found that if choose the smallest c such that $f(x^*) \leq 0$, the solution is within 5% of optimal 70% of the time, and within 30% of optimal 98% of the time, where ”optimal” refers to the solution found using the best value of c . Therefore, in our implementations we use modified binary search to choose c .

此外，我们发现如果选择最小的 c 使得 $f(x^*) \leq 0$ ，则解决方案在时间上与最优解 70% 的差距在 5% 之内，并且在时间上与最优解 98% 的差距在 30% 之内，其中“最优”指的是使用最佳值 c 找到的解决方案。因此，在我们的实现中，我们使用修改的二分搜索来选择 c 。

B.Box constraints

B. 盒约束

To ensure the modification yields a valid image, we have a constraint on δ : we must have $0 \leq x_i + \delta_i \leq 1$ for all i . In the optimization literature, this is known as a ”box constraint.” Previous work uses a particular optimization algorithm, L-BFGS-B, which supports box constraints natively.

为确保修改产生有效的图像，我们对 δ 施加了约束: 对于所有 i ，我们必须满足 $0 \leq x_i + \delta_i \leq 1$ 。在优化文献中，这被称为“盒约束”。之前的工作使用了一种特定的优化算法 L-BFGS-B，该算法本身支持盒约束。

We investigate three different methods of approaching this problem.

我们研究了三种不同的方法来解决这个问题。

1) Projected gradient descent performs one step of standard gradient descent, and then clips all the coordinates to be within the box.

1) 投影梯度下降执行一步标准梯度下降，然后将所有坐标裁剪到盒内。

This approach can work poorly for gradient descent approaches that have a complicated update step (for example, those with momentum): when we clip the actual x_i , we unexpectedly change the input to the next iteration of the algorithm.

对于具有复杂更新步骤 (例如，具有动量的步骤) 的梯度下降方法，这种方法可能效果不佳: 当我们裁剪实际的 x_i 时，意外地改变了算法下一次迭代的输入。

2) Clipped gradient descent does not clip x_i on each iteration; rather, it incorporates the clipping into the objective function to be minimized. In other words, we replace $f(x+\delta)$ with $f(\min(\max(x+\delta, 0), 1))$, with the min and max taken component-wise.

2) 裁剪梯度下降在每次迭代中不裁剪 x_i ; 相反，它将裁剪纳入要最小化的目标函数。换句话说，我们用 $f(\min(\max(x+\delta, 0), 1))$ 替换 $f(x+\delta)$ ，最小值和最大值按分量计算。

While solving the main issue with projected gradient descent, clipping introduces a new problem: the algorithm can get stuck in a flat spot where it has increased some component x_i to be substantially larger than the maximum allowed. When this happens, the partial derivative becomes zero, so even if some improvement is possible by later reducing x_i , gradient descent has no way to detect this.

尽管解决了投影梯度下降的主要问题，裁剪引入了一个新问题: 算法可能会卡在一个平坦区域，在该区域内某个分量 x_i 被增加到远大于允许的最大值。当这种情况发生时，偏导数变为零，因此即使通过稍后减少 x_i 可能会有一些改进，梯度下降也无法检测到这一点。

3) Change of variables introduces a new variable w and instead of optimizing over the variable δ defined above, we apply a change-of-variables and optimize over w , setting

3) 变量变换引入了一个新变量 w ，我们不再对上述定义的变量 δ 进行优化，而是应用变量变换并对 w 进行优化，设定

$$\delta_i = \frac{1}{2}(\tanh(w_i) + 1) - x_i.$$

Since $-1 \leq \tanh(w_i) \leq 1$, it follows that $0 \leq x_i + \delta_i \leq 1$, so the solution will automatically be valid. [8]

由于 $-1 \leq \tanh(w_i) \leq 1$ ，因此 $0 \leq x_i + \delta_i \leq$ 为 1，所以该解将自动有效。[8]

We can think of this approach as a smoothing of clipped gradient descent that eliminates the problem of getting stuck in extreme regions.

我们可以将这种方法视为对剪切梯度下降的平滑处理，消除了陷入极端区域的问题。

These methods allow us to use other optimization algorithms that don’t natively support box constraints. We use the Adam [23] optimizer almost exclusively, as we have found it to be the most effective at quickly finding adversarial examples. We tried three solvers - standard gradient descent, gradient descent with momentum, and Adam - and all three produced identical-quality solutions. However, Adam converges substantially more quickly than the others.

这些方法允许我们使用其他不原生支持盒约束的优化算法。我们几乎完全使用 Adam [23] 优化器，因为我们发现它在快速找到对抗样本方面最为有效。我们尝试了三种求解器——标准梯度下降、带动量的梯度下降和 Adam——这三者产生的解的质量完全相同。然而，Adam 的收敛速度明显快于其他方法。

C. Evaluation of approaches

C. 方法评估

For each possible objective function $f(\cdot)$ and method to enforce the box constraint, we evaluate the quality of the adversarial examples found.

对于每个可能的目标函数 $f(\cdot)$ 和强制盒约束的方法，我们评估找到的对抗样本的质量。

	Best Case						Average Case						Worst Case					
	Change of Variable		Clipped Descent		Projected Descent		Change of Variable		Clipped Descent		Projected Descent		Change of Variable		Clipped Descent		Projected Descent	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
f_1	2.46	100%	2.93	100%	2.31	100%	4.35	100%	5.21	100%	4.11	100%	7.76	100%	9.48	100%	7.37	100%
f_2	4.55	80%	3.97	83%	3.49	83%	3.22	44%	8.99	63%	15.06	74%	2.93	18%	10.22	40%	18.90	53%
f_3	4.54	77%	4.07	81%	3.76	82%	3.47	44%	9.55	63%	15.84	74%	3.09	17%	11.91	41%	24.01	59%
f_4	5.01	86%	6.52	100%	7.53	100%	4.03	55%	7.49	71%	7.60	71%	3.55	24%	4.25	35%	4.10	35%
f_5	1.97	100%	2.20	100%	1.94	100%	3.58	100%	4.20	100%	3.47	100%	6.42	100%	7.86	100%	6.12	100%
f_6	1.94	100%	2.18	100%	1.95	100%	3.47	100%	4.11	100%	3.41	100%	6.03	100%	7.50	100%	5.89	100%
f_7	1.96	100%	2.21	100%	1.94	100%	3.53	100%	4.14	100%	3.43	100%	6.20	100%	7.57	100%	5.94	100%

	最佳情况						平均情况						最坏情况					
	变量变换		截断下降法		投影下降法		变量变换		截断下降法		投影下降法		变量变换		截断下降法		投影下降法	
	均值	概率	均值	概率	均值	概率	均值	概率	均值	概率	均值	概率	均值	概率	均值	概率	均值	概率
f_1	2.46	100%	2.93	100%	2.31	100%	4.35	100%	5.21	100%	4.11	100%	7.76	100%	9.48	100%	7.37	100%
f_2	4.55	80%	3.97	83%	3.49	83%	3.22	44%	8.99	63%	15.06	74%	2.93	18%	10.22	40%	18.90	53%
f_3	4.54	77%	4.07	81%	3.76	82%	3.47	44%	9.55	63%	15.84	74%	3.09	17%	11.91	41%	24.01	59%
f_4	5.01	86%	6.52	100%	7.53	100%	4.03	55%	7.49	71%	7.60	71%	3.55	24%	4.25	35%	4.10	35%
f_5	1.97	100%	2.20	100%	1.94	100%	3.58	100%	4.20	100%	3.47	100%	6.42	100%	7.86	100%	6.12	100%
f_6	1.94	100%	2.18	100%	1.95	100%	3.47	100%	4.11	100%	3.41	100%	6.03	100%	7.50	100%	5.89	100%
f_7	1.96	100%	2.21	100%	1.94	100%	3.53	100%	4.14	100%	3.43	100%	6.20	100%	7.57	100%	5.94	100%

TABLE III

Evaluation of all combinations of one of the seven possible objective functions with one of the three box constraint encodings. We show the average L_2 distortion, The standard deviation, and the success probability (fraction of instances for which an adversarial example can be found). Evaluated on1000random instances. When the success is not 100% , mean is for successful. ATTACKS ONLY.

评估七个可能的目标函数与三种盒约束编码的所有组合。我们展示了平均 L_2 失真、标准差和成功概率 (可以找到对抗样本的实例比例)。在 1000 个随机实例上进行评估。当成功率不是 100% 时，均值是针对成功的。仅针对攻击。

To choose the optimal c , we perform 20 iterations of binary search over c . For each selected value of c , we run 10,000 iterations of gradient descent with the Adam optimizer. [9]

为了选择最佳的 c ，我们对 c 进行 20 次二分搜索迭代。对于每个选定的 c 值，我们使用 Adam 优化器运行 10,000 次梯度下降迭代。[9]

The results of this analysis are in Table III. We evaluate the quality of the adversarial examples found on the MNIST and CIFAR datasets. The relative ordering of each objective function is identical between the two datasets, so for brevity we report only results for MNIST.

本分析的结果见表 III。我们评估了在 MNIST 和 CIFAR 数据集上找到的对抗样本的质量。两个数据集中每个目标函数的相对排序是相同的，因此为了简洁起见，我们仅报告 MNIST 的结果。

There is a factor of three difference in quality between the best objective function and the worst. The choice of method for handling box constraints does not impact the quality of results as significantly for the best minimization functions.

最佳目标函数与最差目标函数之间的质量差异为三倍。处理盒约束的方法选择对最佳最小化函数的结果质量影响并不显著。

[7] The corresponding figures for other objective functions are similar; we omit them for brevity.

[7] 其他目标函数的相应数据类似；为简洁起见，我们省略它们。

[8] Instead of scaling by $\frac{1}{2}$ we scale by $\frac{1}{2}+\epsilon$ to avoid dividing by zero.

[8] 我们不是通过 $\frac{1}{2}$ 进行缩放，而是通过 $\frac{1}{2}+\epsilon$ 进行缩放，以避免除以零。

In fact, the worst performing objective function, cross entropy loss, is the approach that was most suggested in the literature previously [46], [42].

实际上，表现最差的目标函数交叉熵损失是文献中之前最常建议的方法 [46]，[42]。

Why are some loss functions better than others? When $c = 0$, gradient descent will not make any move away from the initial image. However, a large c often causes the initial steps of gradient descent to perform in an overly-greedy manner, only traveling in the direction which can most easily reduce f and ignoring the $\mathcal{D}$ loss - thus causing gradient descent to find sub-optimal solutions.

为什么某些损失函数比其他损失函数更好？当 $c = 0$ 时，梯度下降不会偏离初始图像。然而，较大的 c 通常会导致梯度下降的初始步骤表现得过于贪婪，仅沿着最容易减少 f 的方向移动，而忽略 $\mathcal{D}$ 损失，从而导致梯度下降找到次优解。

This means that for loss function f_1 and f_4 , there is no good constant c that is useful throughout the duration of the gradient descent search. Since the constant c weights the relative importance of the distance term and the loss term, in order for a fixed constant c to be useful, the relative value of these two terms should remain approximately equal. This is not the case for these two loss functions.

这意味着对于损失函数 f_1 和 f_4 ，没有一个好的常数 c 在梯度下降搜索的整个过程中是有用的。由于常数 c 权衡了距离项和损失项的相对重要性，为了使固定常数 c 有用，这两个项的相对值应该保持大致相等。但这对于这两个损失函数并不成立。

To explain why this is the case, we will have to take a side discussion to analyze how adversarial examples exist. Consider a valid input x and an adversarial example x' on a network.

为了解释为什么会出现这种情况，我们需要进行一个侧面讨论，以分析对抗样本是如何存在的。考虑一个有效输入 x 和一个网络上的对抗样本 x' 。

What does it look like as we linearly interpolate from x to x' ? That is, let $y = \alpha x + (1-\alpha) x'$ for $\alpha \in [0,1]$. It turns out the value of $Z(\cdot)_t$ is mostly linear from the input to the adversarial example, and therefore the $F(\cdot)_t$ is a logistic. We verify this fact empirically by constructing adversarial examples on the first 1,000 test images on both the MNIST and CIFAR dataset with our approach, and find the Pearson correlation coefficient $r > .9$.

当我们从 x 线性插值到 x' 时，它看起来是什么样子？也就是说，设 $y = \alpha x + (1-\alpha) x'$ 为 $\alpha \in [0,1]$ 。结果表明，从输入到对抗样本的 $Z(\cdot)_t$ 值大多是线性的，因此 $F(\cdot)_t$ 是一个逻辑函数。我们通过在 MNIST 和 CIFAR 数据集的前 1,000 张测试图像上构造对抗样本，实证验证了这一事实，并发现皮尔逊相关系数 $r > .9$ 。

Given this, consider loss function f_4 (the argument for f_1 is similar). In order for the gradient descent attack to make any change initially, the constant c will have to be large enough that

鉴于此，考虑损失函数 f_4 (f_1 的参数类似)。为了使梯度下降攻击在最初有所改变，常数 c 必须足够大，以便

$$\epsilon < c \left(f_1 (x+\epsilon) - f_1 (x) \right)$$

or, as $\epsilon \to 0$,

或者，如 $\epsilon \to 0$ 所示，

$$1/c < |\nabla f_1 (x)|$$

implying that c must be larger than the inverse of the gradient to make progress, but the gradient of f_1 is identical to $F(\cdot)_t$ so will be tiny around the initial image, meaning c will have to be extremely large.

这意味着 c 必须大于梯度的倒数才能取得进展，但 f_1 的梯度与 $F(\cdot)_t$ 相同，因此在初始图像附近会非常小，这意味着 c 必须极其大。

However, as soon as we leave the immediate vicinity of the initial image, the gradient of $\nabla f_1 (x+\delta)$ increases at an exponential rate, making the large constant c cause gradient descent to perform in an overly greedy manner.

然而，一旦我们离开初始图像的直接邻域，$\nabla f_1 (x+\delta)$ 的梯度以指数速率增加，使得大的常数 c 导致梯度下降以过于贪婪的方式进行。

We verify all of this theory empirically. When we run our attack trying constants chosen from 10^{-10} to 10^{10} the average constant for loss function f_4 was 10^6 .

我们通过实证验证了所有这些理论。当我们运行攻击尝试从 10^{-10} 到 10^{10} 选择的常数时，损失函数 f_4 的平均常数为 10^6 。

The average gradient of the loss function f_1 around the valid image is 2^{-20} but 2^{-1} at the closest adversarial example. This means c is a million times larger than it has to be, causing the loss function f_4 and f_1 to perform worse than any of the others.

在有效图像周围，损失函数 f_1 的平均梯度为 2^{-20} ，而在最近的对抗样本处为 2^{-1} 。这意味着 c 比所需的要大一百万倍，导致损失函数 f_4 和 f_1 的表现比其他任何函数都要差。

D. Discretization

D. 离散化

We model pixel intensities as a (continuous) real number in the range $[0,1]$. However, in a valid image, each pixel intensity must be a (discrete) integer in the range $\{0,1,\ldots,255\}$. This additional requirement is not captured in our formulation. In practice, we ignore the integrality constraints, solve the continuous optimization problem, and then round to the nearest integer: the intensity of the i th pixel becomes $\lfloor 255\left(x_{i}+\delta_{i}\right)\rceil$.

我们将像素强度建模为范围在 $[0,1]$ 内的 (连续) 实数。然而，在有效图像中，每个像素强度必须是范围在 $\{0,1,\ldots,255\}$ 内的 (离散) 整数。这个额外的要求在我们的公式中没有体现。在实践中，我们忽略了整数约束，解决连续优化问题，然后四舍五入到最近的整数: i 号像素的强度变为 $\lfloor 255\left(x_{i}+\delta_{i}\right)\rceil$ 。

This rounding will slightly degrade the quality of the adversarial example. If we need to restore the attack quality, we perform greedy search on the lattice defined by the discrete

这种四舍五入会稍微降低对抗样本的质量。如果我们需要恢复攻击质量，我们将在离散解空间上进行贪婪搜索。

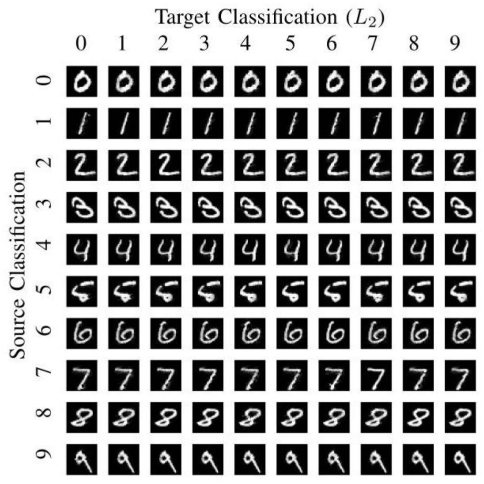
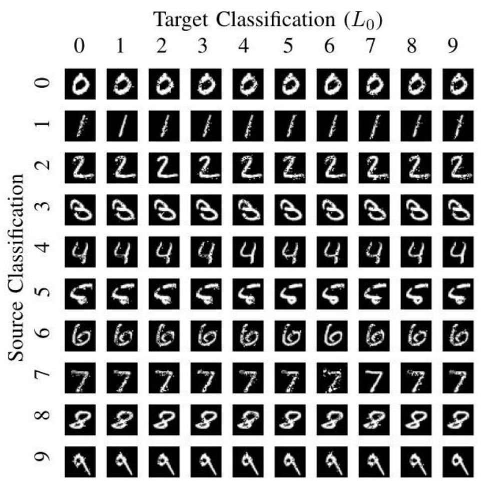


Fig. 3. Our L_2 adversary applied to the MNIST dataset performing a targeted attack for every source/target pair. Each digit is the first image in the dataset with that label.

图 3. 我们的 L_2 对手应用于 MNIST 数据集，对每个源/目标对执行定向攻击。每个数字是数据集中具有该标签的第一张图像。



[9] Adam converges to 95% of optimum within 1,000 iterations 92% of the time. For completeness we run it for 10,000 iterations at each step.

[9] Adam 在 1,000 次迭代内收敛到 95% 的最优解 92% 。为了完整性，我们在每一步运行 10,000 次迭代。

Fig. 4. Our L_0 adversary applied to the MNIST dataset performing a targeted attack for every source/target pair. Each digit is the first image in the dataset with that label.

图 4. 我们的 L_0 对手应用于 MNIST 数据集，对每个源/目标对执行定向攻击。每个数字是数据集中具有该标签的第一张图像。

solutions by changing one pixel value at a time. This greedy search never failed for any of our attacks.

通过一次改变一个像素值的方式进行解决。这种贪婪搜索在我们的任何攻击中都没有失败。

Prior work has largely ignored the integrality constraints. [10] For instance, when using the fast gradient sign attack with $\epsilon = 0.1$ (i.e., changing pixel values by 10%), discretization rarely affects the success rate of the attack. In contrast, in our work, we are able to find attacks that make much smaller changes to the images, so discretization effects cannot be ignored. We take care to always generate valid images; when reporting the success rate of our attacks, they always are for attacks that include the discretization post-processing.

先前的工作在很大程度上忽视了整数约束。[10] 例如，当使用快速梯度符号攻击并设置 $\epsilon =$ 为 0.1(即，改变像素值 10%) 时，离散化很少影响攻击的成功率。相比之下，在我们的工作中，我们能够找到对图像进行更小变化的攻击，因此离散化的影响不能被忽视。我们始终小心生成有效图像；在报告我们的攻击成功率时，这些攻击总是包括离散化后处理。

VI.OUR THREE ATTACKS

VI. 我们的三种攻击

A.Our L_2 Attack

A. 我们的 L_2 攻击

Putting these ideas together, we obtain a method for finding adversarial examples that will have low distortion in the L_2 metric. Given x , we choose a target class t (such that we have $t \neq C^*(x)$) and then search for w that solves

将这些想法结合起来，我们获得了一种寻找对抗样本的方法，该方法在 L_2 度量中具有低失真。给定 x ，我们选择一个目标类别 t (使得我们有 $t \neq C^*(x)$)，然后搜索 w 来解决

$$\text{minimize } \left\| \frac{1}{2}(\tanh(w)+1) - x\right\|_2^2 + c \cdot f\left(\frac{1}{2}(\tanh(w)+1)\right)$$

with f defined as

其中 f 定义为

$$f(x') = \max\left(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa\right).$$

This f is based on the best objective function found earlier, modified slightly so that we can control the confidence with which the misclassification occurs by adjusting κ . The parameter κ encourages the solver to find an adversarial instance x' that will be classified as class t with high confidence. We set $\kappa = 0$ for our attacks but we note here that a side benefit of this formulation is it allows one to control for the desired confidence. This is discussed further in Section VIII-D.

这个 f 基于之前找到的最佳目标函数，稍作修改，以便我们可以通过调整 κ 来控制错误分类发生的置信度。参数 κ 鼓励求解器找到一个对抗实例 x' ，该实例将以高置信度被分类为类别 t 。我们为我们的攻击设置 $\kappa = 0$ ，但我们在这里注意到，这种表述的一个附带好处是它允许控制所需的置信度。有关更多讨论，请参见第 VIII-D 节。

Figure 3 shows this attack applied to our MNIST model for each source digit and target digit. Almost all attacks are visually indistinguishable from the original digit.

图 3 显示了这种攻击应用于我们的 MNIST 模型，针对每个源数字和目标数字。几乎所有攻击在视觉上与原始数字无法区分。

A comparable figure (Figure 12) for CIFAR is in the appendix. No attack is visually distinguishable from the baseline image.

附录中有一个可比较的图 (图 12)，针对 CIFAR。没有攻击在视觉上与基线图像可区分。

Multiple starting-point gradient descent. The main problem with gradient descent is that its greedy search is not guaranteed to find the optimal solution and can become stuck in a local minimum. To remedy this, we pick multiple random starting points close to the original image and run gradient descent from each of those points for a fixed number of iterations. We randomly sample points uniformly from the ball of radius r , where r is the closest adversarial example found so far. Starting from multiple starting points reduces the likelihood that gradient descent gets stuck in a bad local minimum.

多起始点梯度下降。梯度下降的主要问题是其贪婪搜索无法保证找到最优解，并且可能会陷入局部最小值。为了解决这个问题，我们选择多个随机起始点，靠近原始图像，并从每个点运行固定次数的梯度下降。我们从半径为 r 的球体中均匀随机采样点，其中 r 是迄今为止找到的最近对抗样本。从多个起始点开始可以减少梯度下降陷入不良局部最小值的可能性。

B.Our L_0 Attack

B. 我们的 L_0 攻击

The L_0 distance metric is non-differentiable and therefore is ill-suited for standard gradient descent. Instead, we use an iterative algorithm that, in each iteration, identifies some pixels that don't have much effect on the classifier output and then fixes those pixels, so their value will never be changed. The set of fixed pixels grows in each iteration until we have, by process of elimination, identified a minimal (but possibly not minimum) subset of pixels that can be modified to generate an adversarial example. In each iteration, we use our L_2 attack to identify which pixels are unimportant.

L_0 距离度量是不可微分的，因此不适合标准的梯度下降。相反，我们使用一种迭代算法，在每次迭代中，识别出一些对分类器输出影响不大的像素，然后固定这些像素，使其值不会被改变。固定像素的集合在每次迭代中增长，直到通过消去法识别出一个最小 (但可能不是最小) 可修改的像素子集，以生成对抗样本。在每次迭代中，我们使用我们的 L_2 攻击来识别哪些像素是不重要的。

In more detail, on each iteration, we call the L_2 adversary, restricted to only modify the pixels in the allowed set. Let δ be the solution returned from the L_2 adversary on input image x , so that $x+\delta$ is an adversarial example. We compute $g=\nabla f(x+\delta)$ (the gradient of the objective function, evaluated at the adversarial instance). We then select the pixel $i=\arg\min_i g_i \cdot \delta_i$ and fix i , i.e., remove i from the allowed set. [11] The intuition is that $g_i \cdot \delta_i$ tells us how much reduction to $f(\cdot)$ we obtain from the i th pixel of the image, when moving from x to $x+\delta$: g_i tells us how much reduction in f we obtain, per unit change to the i th pixel, and we multiply this by how much the i th pixel has changed. This process repeats until the L_2 adversary fails to find an adversarial example.

更详细地说，在每次迭代中，我们调用 L_2 对手，限制其仅修改允许集合中的像素。设 δ 为 L_2 对手在输入图像 x 上返回的解，因此 $x+\delta$ 是一个对抗样本。我们计算 $g=\nabla f(x+\delta)$ (在对抗实例上评估的目标函数的梯度)。然后我们选择像素 $i=\arg\min_i g_i \cdot \delta_i$ 并固定 i ，即将 i 从允许集合中移除。[11] 直觉是 $g_i \cdot \delta_i$ 告诉我们，当从 x 移动到 $x+\delta$: g_i 时，我们从图像的第 i 个像素获得了多少 $f(\cdot)$ 的减少，而 f 告诉我们每单位改变第 i 个像素时我们获得了多少 f 的减少，我们将其乘以第 i 个像素的变化量。这个过程重复进行，直到 L_2 对手未能找到对抗样本。

There is one final detail required to achieve strong results: choosing a constant c to use for the L_2 adversary. To do this, we initially set c to a very low value (e.g., 10^{-4}). We then run our L_2 adversary at this c -value. If it fails, we double c and try again, until it is successful. We abort the search if c exceeds a fixed threshold (e.g., 10^{10}).

为了获得强有力的结果，还需要一个最终细节: 选择一个常数 c 用于 L_2 对手。为此，我们最初将 c 设置为一个非常低的值 (例如，10^{-4})。然后，我们在这个 c 值下运行我们的 L_2 对手。如果失败，我们将 c 加倍并重试，直到成功。如果 c 超过固定阈值 (例如，10^{10})，我们将中止搜索。

JSMA grows a set - initially empty - of pixels that are allowed to be changed and sets the pixels to maximize the total loss. In contrast, our attack shrinks the set of pixels - initially containing every pixel - that are allowed to be changed.

JSMA 生成一组 - 最初为空 - 允许更改的像素，并将这些像素设置为最大化总损失。相比之下，我们的攻击缩小了允许更改的像素集 - 最初包含每个像素。

Our algorithm is significantly more effective than JSMA (see Section VII for an evaluation). It is also efficient: we introduce optimizations that make it about as fast as our L_2 attack with a single starting point on MNIST and CIFAR; it is substantially slower on ImageNet. Instead of starting gradient descent in each iteration from the initial image, we start the gradient descent from the solution found on the

[10] One exception: The JSMA attack [38] handles this by only setting the output value to either 0 or 255 .

[10] 一个例外:JSMA 攻击 [38] 通过仅将输出值设置为 0 或 255 来处理这个问题。

previous iteration (”warm-start”). This dramatically reduces the number of rounds of gradient descent needed during each iteration, as the solution with k pixels held constant is often very similar to the solution with $k+1$ pixels held constant.

我们的算法显著优于 JSMA(见第七节的评估)。它也是高效的: 我们引入了优化，使其速度与我们在 MNIST 和 CIFAR 上的 L_2 攻击相当；在 ImageNet 上则显著较慢。我们不再在每次迭代中从初始图像开始梯度下降，而是从上一次迭代找到的解决方案开始梯度下降（“热启动”）。这显著减少了每次迭代中所需的梯度下降轮数，因为在 k 像素保持不变的情况下的解决方案通常与在 $k+1$ 像素保持不变的情况下的解决方案非常相似。

Figure 4 shows the L_0 attack applied to one digit of each source class, targeting each target class, on the MNIST dataset. The attacks are visually noticeable, implying the L_0 attack is more difficult than L_2 . Perhaps the worst case is that of a 7 being made to classify as a 6 ; interestingly, this attack for L_2 is one of the only visually distinguishable attacks.

图 4 显示了在 MNIST 数据集上对每个源类的一个数字应用的 L_0 攻击，目标是每个目标类。这些攻击在视觉上是显著的，暗示着 L_0 攻击比 L_2 更困难。也许最糟糕的情况是将一个 7 误分类为 6；有趣的是，这种对 L_2 的攻击是唯一在视觉上可区分的攻击之一。

A comparable figure (Figure 11) for CIFAR is in the appendix.

CIFAR 的类似图 (图 11) 在附录中。

C.Our L_∞ Attack

C. 我们的 L_∞ 攻击

The L_∞ distance metric is not fully differentiable and standard gradient descent does not perform well for it. We experimented with naively optimizing

L_∞ 距离度量并不是完全可微的，标准的梯度下降在此情况下表现不佳。我们尝试了简单的优化方法

$$\text{minimize } c \cdot f(x+\delta) + \|\delta\|_\infty$$

However, we found that gradient descent produces very poor results: the $\|\delta\|_\infty$ term only penalizes the largest (in absolute value) entry in δ and has no impact on any of the other. As such, gradient descent very quickly becomes stuck oscillating between two suboptimal solutions. Consider a case where $\delta_i = 0.5$ and $\delta_j = 0.5 - \epsilon$. The L_∞ norm will only penalize δ_i , not δ_j , and $\frac{\partial}{\partial \delta_j} \|\delta\|_\infty$ will be zero at this point. Thus, the gradient imposes no penalty for increasing δ_j , even though it is already large. On the next iteration we might move to a position where δ_j is slightly larger than δ_i , say $\delta_i = 0.5 - \epsilon'$ and $\delta_j = 0.5 + \epsilon''$, a mirror image of where we started. In other words, gradient descent may oscillate back and forth across the line $\delta_i = \delta_j = 0.5$, making it nearly impossible to make progress.

然而，我们发现梯度下降产生的结果非常糟糕: $\|\delta\|_\infty$ 项仅对 δ 中绝对值最大的条目施加惩罚，而对其他条目没有影响。因此，梯度下降很快就陷入在两个次优解之间的振荡。考虑一个情况，其中 $\delta_i =$ 为 0.5 和 $\delta_j = 0.5 - \epsilon$ 。L_∞ 范数仅对 δ_i 施加惩罚，而对 δ_j 不施加惩罚，并且此时 $\frac{\partial}{\partial \delta_j} \|\delta\|_\infty$ 将为零。因此，尽管 δ_j 已经很大，梯度对增加 δ_j 不施加任何惩罚。在下一次迭代中，我们可能会移动到一个位置，在该位置 δ_j 稍微大于 δ_i ，假设为 $\delta_i = 0.5 - \epsilon'$ 和 $\delta_j = 0.5 + \epsilon''$ ，这是我们开始时的镜像图像。换句话说，梯度下降可能在 $\delta_i = \delta_j = 0.5$ 线的两侧来回振荡，使得几乎不可能取得进展。

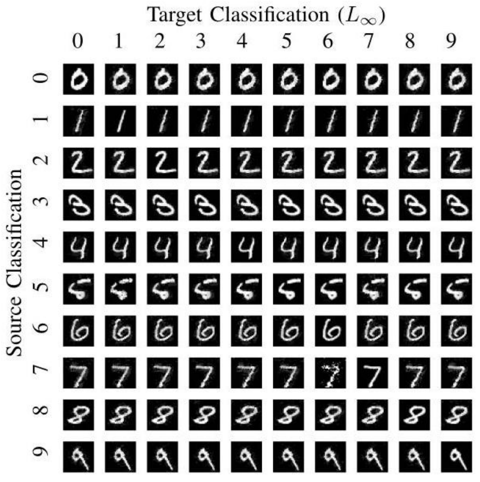


Fig. 5. Our L_∞ adversary applied to the MNIST dataset performing a targeted attack for every source/target pair. Each digit is the first image in the dataset with that label.

图 5. 我们的 L_∞ 对手应用于 MNIST 数据集，对每个源/目标对执行有针对性的攻击。每个数字是数据集中具有该标签的第一张图像。

We resolve this issue using an iterative attack. We replace the L_2 term in the objective function with a penalty for any terms that exceed τ (initially 1, decreasing in each iteration). This prevents oscillation, as this loss term penalizes all large values simultaneously. Specifically, in each iteration we solve

我们通过使用迭代攻击解决了这个问题。我们在目标函数中用对任何超过 τ 的项施加惩罚来替换 L_2 项 (初始值为 1，在每次迭代中递减)。这防止了振荡，因为这个损失项同时惩罚所有大值。具体来说，在每次迭代中我们解决

$$\text{minimize } c \cdot f(x+\delta) + \cdot \sum_i \left[(\delta_i - \tau)^+\right]$$

After each iteration, if $\delta_i < \tau$ for all i , we reduce τ by a factor of 0.9 and repeat; otherwise, we terminate the search.

在每次迭代后，如果 $\delta_i < \tau$ 对所有 i 成立，我们将 τ 减少 0.9 倍并重复；否则，我们终止搜索。

Again we must choose a good constant c to use for the L_∞ adversary. We take the same approach as we do for the L_0 attack: initially set c to a very low value and run the L_∞ adversary at this c -value. If it fails, we double c and try again, until it is successful. We abort the search if c exceeds a fixed threshold.

我们必须再次选择一个合适的常数 c 用于 L_∞ 对手。我们采用与 L_0 攻击相同的方法: 最初将 c 设置为一个非常低的值，并在此 c 值下运行 L_∞ 对手。如果失败，我们将 c 加倍并重试，直到成功。如果 c 超过固定阈值，我们将中止搜索。

Using "warm-start" for gradient descent in each iteration, this algorithm is about as fast as our L_2 algorithm (with a single starting point).

在每次迭代中使用 "热启动" 进行梯度下降，该算法的速度与我们的 L_2 算法 (使用单一起始点) 相当。

Figure 5 shows the L_∞ attack applied to one digit of each source class, targeting each target class, on the MNSIT dataset. While most differences are not visually noticeable, a few are. Again, the worst case is that of a 7 being made to classify as a 6 .

图 5 显示了对 MNSIT 数据集中每个源类的一个数字应用的 L_∞ 攻击，目标是每个目标类。虽然大多数差异在视觉上并不明显，但有一些是的。再次强调，最糟糕的情况是一个 7 被错误分类为 6。

	Untargeted		Average Case		Least Likely	
	mean	prob	mean	prob	mean	prob
Our L_0	48	100%	410	100%	5200	100%
JSMA-Z	-	0%	-	0%	-	0%
JSMA-F	-	0%	-	0%	-	0%
Our L_2	0.32	100%	0.96	100%	2.22	100%
Deepfool	0.91	100%	-	-	-	-
Our L_∞	0.004	100%	0.006	100%	0.01	100%
FGS	0.004	100%	0.064	2%	-	0%
IGS	0.004	100%	0.01	99%	0.03	98%

	非定向		平均情况		最不可能	
	均值	概率	均值	概率	均值	概率
我们的 L_0	48	100%	410	100%	5200	100%
JSMA-Z	-	0%	-	0%	-	0%
JSMA-F	-	0%	-	0%	-	0%
我们的 L_2	0.32	100%	0.96	100%	2.22	100%
Deepfool	0.91	100%	-	-	-	-
我们的 L_∞	0.004	100%	0.006	100%	0.01	100%
FGS	0.004	100%	0.064	2%	-	0%
IGS	0.004	100%	0.01	99%	0.03	98%

[11] Selecting the index i that minimizes δ_i is simpler, but it yields results with 1.5× higher L_0 distortion.

[11] 选择最小化 δ_i 的索引 i 更简单，但它的结果具有更高的 1.5× 失真 L_0 。

TABLE V
COMPARISON OF THE THREE VARIANTS OF TARGETED ATTACK TO PREVIOUS WORK FOR THE INCEPTION V3 MODEL ON IMAGENET. WHEN SUCCESS RATE IS NOT 100%, THE MEAN IS ONLY OVER SUCCESSES.

针对 IMAGENET 的 INCEPTION V3 模型，三种目标攻击变体与之前工作的比较。当成功率不是 100% 时，均值仅针对成功案例计算。

A comparable figure (Figure 13) for CIFAR is in the appendix. No attack is visually distinguishable from the baseline image.

附录中有 CIFAR 的可比图 (图 13)。没有攻击在视觉上与基线图像可区分。

VII. Attack Evaluation

VII. 攻击评估

We compare our targeted attacks to the best results previously reported in prior publications, for each of the three distance metrics.

我们将我们的目标攻击与之前出版物中报告的最佳结果进行比较，针对每个距离度量。

We re-implement Deepfool, fast gradient sign, and iterative gradient sign. For fast gradient sign, we search over ϵ to find the smallest distance that generates an adversarial example; failures is returned if no ϵ produces the target class. Our iterative gradient sign method is similar: we search over ϵ (fixing $\alpha = \frac{1}{256}$) and return the smallest successful.

我们重新实现了 Deepfool、快速梯度符号和迭代梯度符号。对于快速梯度符号，我们在 ϵ 上搜索，以找到生成对抗样本的最小距离；如果没有 ϵ 产生目标类别，则返回失败。我们的迭代梯度符号方法类似: 我们在 ϵ 上搜索 (固定 $\alpha = \frac{1}{256}$)，并返回最小的成功结果。

For JSMA we use the implementation in CleverHans [35] with only slight modification (we improve performance by 50× with no impact on accuracy).

对于 JSMA，我们使用 CleverHans [35] 中的实现，仅进行轻微修改 (我们通过 50× 提高性能，而不影响准确性)。

JSMA is unable to run on ImageNet due to an inherent significant computational cost: recall that JSMA performs search for a pair of pixels p, q that can be changed together that make the target class more likely and other classes less likely. ImageNet represents images as $299 \times 299 \times 3$ vectors, so searching over all pairs of pixels would require 2^{36} work on each step of the calculation. If we remove the search over pairs of pixels, the success of JSMA falls off dramatically. We therefore report it as failing always on ImageNet.

由于固有的显著计算成本，JSMA 无法在 ImageNet 上运行: 请记住，JSMA 进行搜索以找到一对可以一起改变的像素 p, q ，使目标类别更可能，而其他类别更不可能。ImageNet 将图像表示为 $299 \times 299 \times 3$ 向量，因此在每一步计算中搜索所有像素对将需要 2^{36} 的工作量。如果我们去掉对像素对的搜索，JSMA 的成功率将急剧下降。因此，我们报告它在 ImageNet 上总是失败。

We report success if the attack produced an adversarial example with the correct target label, no matter how much change was required. Failure indicates the case where the attack was entirely unable to succeed.

如果攻击产生了具有正确目标标签的对抗样本，无论所需的变化量如何，我们报告成功。失败表示攻击完全无法成功的情况。

We evaluate on the first 1,000 images in the test set on CIFAR and MNSIT. On ImageNet, we report on 1,000 images that were initially classified correctly by Inception v3 [12] . On ImageNet we approximate the best-case and worst-case results by choosing 100 target classes (10%) at random.

我们在 CIFAR 和 MNIST 的测试集中评估前 1,000 张图像。在 ImageNet 上，我们报告最初由 Inception v3 [12] 正确分类的 1,000 张图像。在 ImageNet 上，我们通过随机选择 100 个目标类别 (10%) 来近似最佳情况和最坏情况的结果。

The results are found in Table IV for MNIST and CIFAR, and Table V for ImageNet. [13]

结果见表 IV(MNIST 和 CIFAR) 和表 V(ImageNet)。[13]

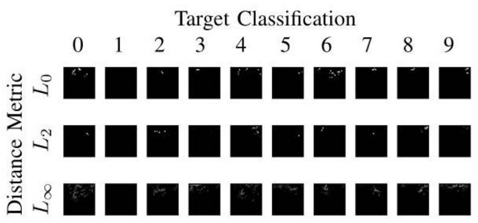


Fig. 6. Targeted attacks for each of the 10 MNIST digits where the starting image is totally black for each of the three distance metrics.

图 6. 针对每个 MNIST 数字的有针对性的攻击，其中起始图像对于每个距离度量都是完全黑色的。

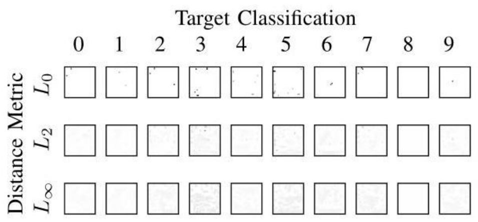


Fig. 7. Targeted attacks for each of the 10 MNIST digits where the starting image is totally white for each of the three distance metrics.

图 7. 针对每个 MNIST 数字的针对性攻击，其中每个距离度量的起始图像完全为白色。

For each distance metric, across all three datasets, our attacks find closer adversarial examples than the previous state-of-the-art attacks, and our attacks never fail to find an adversarial example. Our L_0 and L_2 attacks find adversarial examples with $2\times$ to $10\times$ lower distortion than the best previously published attacks, and succeed with 100% probability. Our L_∞ attacks are comparable in quality to prior work, but their success rate is higher. Our L_∞ attacks on ImageNet are so successful that we can change the classification of an image to any desired label by only flipping the lowest bit of each pixel, a change that would be impossible to detect visually.

对于每个距离度量，在所有三个数据集中，我们的攻击找到的对抗样本比之前的最先进攻击更接近，并且我们的攻击从未未能找到对抗样本。我们的 L_0 和 L_2 攻击找到的对抗样本的 $2\times$ 到 $10\times$ 的失真比之前发布的最佳攻击低，并且成功率为 100% 。我们的 L_∞ 攻击在质量上与之前的工作相当，但成功率更高。我们的 L_∞ 攻击在 ImageNet 上如此成功，以至于我们只需翻转每个像素的最低位，就可以将图像的分类更改为任何期望的标签，这种变化在视觉上是无法检测到的。

As the learning task becomes increasingly more difficult, the previous attacks produce worse results, due to the complexity of the model. In contrast, our attacks perform even better as the task complexity increases. We have found JSMA is unable to find targeted L_0 adversarial examples on ImageNet, whereas ours is able to with 100% success.

随着学习任务变得越来越困难，之前的攻击由于模型的复杂性而产生更差的结果。相反，随着任务复杂性的增加，我们的攻击表现得更好。我们发现 JSMA 无法在 ImageNet 上找到针对性的 L_0 对抗样本，而我们的攻击则能够以 100% 的成功率找到。

It is important to realize that the results between models are not directly comparable. For example, even though a L_0 adversary must change 10 times as many pixels to switch an ImageNet classification compared to a MNIST classification, ImageNet has $114\times$ as many pixels and so the fraction of pixels that must change is significantly smaller.

重要的是要意识到不同模型之间的结果并不能直接比较。例如，尽管一个 L_0 对手必须更改 10 倍的像素才能切换 ImageNet 分类，而与 MNIST 分类相比，ImageNet 的像素数量是 $114\times$ ，因此必须更改的像素比例显著更小。

Generating synthetic digits. With our targeted adversary, we can start from any image we want and find adversarial examples of each given target. Using this, in Figure 6 we show the minimum perturbation to an entirely-black image required to make it classify as each digit, for each of the distance metrics.

生成合成数字。通过我们的针对性对手，我们可以从任何我们想要的图像开始，并找到每个给定目标的对抗样本。利用这一点，在图 6 中，我们展示了将完全黑色图像分类为每个数字所需的最小扰动，对于每个距离度量。

	Best Case				Average Case				Worst Case			
	MNIST		CIFAR		MNIST		CIFAR		MNIST		CIFAR	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
Our L_0	8.5	100%	5.9	100%	16	100%	13	100%	33	100%	24	100%
JSMA-Z	20	100%	20	100%	56	100%	58	100%	180	98%	150	100%
JSMA-F	17	100%	25	100%	45	100%	110	100%	100	100%	240	100%
Our L_2	1.36	100%	0.17	100%	1.76	100%	0.33	100%	2.60	100%	0.51	100%
Deepfool	2.11	100%	0.85	100%	-	-	-	-	-	-	-	-
Our L_∞	0.13	100%	0.0092	100%	0.16	100%	0.013	100%	0.23	100%	0.019	100%
Fast Gradient Sign	0.22	100%	0.015	99%	0.26	42%	0.029	51%	-	0%	0.34	1%
Iterative Gradient Sign	0.14	100%	0.0078	100%	0.19	100%	0.014	100%	0.26	100%	0.023	100%

	最佳情况				平均情况				最坏情况			
	MNIST		CIFAR		MNIST		CIFAR		MNIST		CIFAR	
	均值	概率	均值	概率	均值	概率	均值	概率	均值	概率	均值	概率
我们的 L_0	8.5	100%	5.9	100%	16	100%	13	100%	33	100%	24	100%
JSMA-Z	20	100%	20	100%	56	100%	58	100%	180	98%	150	100%
JSMA-F	17	100%	25	100%	45	100%	110	100%	100	100%	240	100%
我们的 L_2	1.36	100%	0.17	100%	1.76	100%	0.33	100%	2.60	100%	0.51	100%
Deepfool	2.11	100%	0.85	100%	-	-	-	-	-	-	-	-
我们的 L_∞	0.13	100%	0.0092	100%	0.16	100%	0.013	100%	0.23	100%	0.019	100%
快速梯度符号	0.22	100%	0.015	99%	0.26	42%	0.029	51%	-	0%	0.34	1%
迭代梯度符号	0.14	100%	0.0078	100%	0.19	100%	0.014	100%	0.26	100%	0.023	100%

TABLE IV
COMPARISON OF THE THREE VARIANTS OF TARGETED ATTACK TO PREVIOUS WORK FOR OUR MNIST AND CIFAR MODELS. WHEN SUCCESS RATE IS NOT 100% , THE MEAN IS ONLY OVER SUCCESSES.
针对我们的 MNIST 和 CIFAR 模型，三种目标攻击变体与之前工作的比较。当成功率不是 100% 时，均值仅基于成功案例。

This experiment was performed for the L_0 task previously [38], however when mounting their attack, ”for classes 0, 2, 3 and 5 one can clearly recognize the target digit.” With our more powerful attacks, none of the digits are recognizable. Figure 7 performs the same analysis starting from an all-white image.

该实验之前在 L_0 任务中进行过 [38]，然而在进行攻击时，“对于类别 0、2、3 和 5，可以清楚地识别出目标数字。” 而在我们更强大的攻击下，没有任何数字是可识别的。图 7 从一幅全白图像开始进行了相同的分析。

Notice that the all-black image requires no change to become a digit 1 because it is initially classified as a 1, and the all-white image requires no change to become a 8 because the initial image is already an 8 .

请注意，全黑图像无需更改即可成为数字 1，因为它最初被分类为 1，而全白图像无需更改即可成为 8，因为初始图像已经是 8。

Runtime Analysis. We believe there are two reasons why one may consider the runtime performance of adversarial example generation algorithms important: first, to understand if the performance would be prohibitive for an adversary to actually mount the attacks, and second, to be used as an inner loop in adversarial re-training [11].

运行时分析。我们认为有两个原因可以考虑对对抗样本生成算法的运行时性能进行评估: 首先，了解性能是否会对对手实际发起攻击造成障碍；其次，可以作为对抗再训练中的内部循环 [11]。

Comparing the exact runtime of attacks can be misleading. For example, we have parallelized the implementation of our L_2 adversary allowing it to run hundreds of attacks simultaneously on a GPU, increasing performance from 10× to 100× . However, we did not parallelize our L_0 or L_∞ attacks. Similarly, our implementation of fast gradient sign is parallelized, but JSMA is not. We therefore refrain from giving exact performance numbers because we believe an unfair comparison is worse than no comparison.

比较攻击的确切运行时可能会产生误导。例如，我们已经对我们的 L_2 对手的实现进行了并行化，使其能够在 GPU 上同时运行数百次攻击，将性能从 10× 提高到 100× 。然而，我们并没有对我们的 L_0 或

[12] Otherwise the best-case attack results would appear to succeed extremely often artificially low due to the relatively low top-1 accuracy
[12] 否则，最佳攻击结果的表现将由于相对较低的 top-1 准确率而显得成功率极低。
[13] The complete code to reproduce these tables and figures is available online at http://nicholas.carlini.com/code/nn_robust_attacks.
[13] 重现这些表格和图形的完整代码可在线获取，网址为 http://nicholas.carlini.com/code/nn_robust_attacks。

L_∞ 攻击进行并行化。同样，我们的快速梯度符号实现是并行化的，但 JSMA 不是。因此，我们避免给出确切的性能数字，因为我们认为不公平的比较比没有比较更糟。

All of our attacks, and all previous attacks, are plenty efficient to be used by an adversary. No attack takes longer than a few minutes to run on any given instance.

我们所有的攻击，以及所有之前的攻击，都足够高效，可以被对手使用。任何给定实例上的攻击运行时间都不会超过几分钟。

When compared to L_0 , our attacks are $2\times -10\times$ slower than our optimized JSMA algorithm (and significantly faster than the un-optimized version). Our attacks are typically $10\times$ $-100\times$ slower than previous attacks for L_2 and L_∞ , with exception of iterative gradient sign which we are $10\times$ slower.

与 L_0 相比，我们的攻击比优化后的 JSMA 算法 $2\times -10\times$ 慢 (并且显著快于未优化版本)。我们的攻击通常比之前的攻击在 L_2 和 L_∞ 上慢 $10\times$ $-100\times$ ，唯一的例外是迭代梯度符号，我们在这方面 $10\times$ 更慢。

VIII. EVALUATING DEFENSIVE DISTILLATION

VIII. 评估防御蒸馏

Distillation was initially proposed as an approach to reduce a large model (the teacher) down to a smaller distilled model [19]. At a high level, distillation works by first training the teacher model on the training set in a standard manner. Then, we use the teacher to label each instance in the training set with soft labels (the output vector from the teacher network). For example, while the hard label for an image of a hand-written digit 7 will say it is classified as a seven, the soft labels might say it has a 80% chance of being a seven and a 20% chance of being a one. Then, we train the distilled model on the soft labels from the teacher, rather than on the hard labels from the training set. Distillation can potentially increase accuracy on the test set as well as the rate at which the smaller model learns to predict the hard labels [19], [30].

蒸馏最初被提出作为一种将大型模型 (教师) 缩减为更小的蒸馏模型的方法 [19]。从高层次来看，蒸馏的工作方式是首先以标准方式在训练集上训练教师模型。然后，我们使用教师对训练集中的每个实例进行软标签标注 (来自教师网络的输出向量)。例如，对于手写数字 7 的图像，硬标签会说它被分类为七，而软标签可能会说它有 80% 的概率是七，有 20% 的概率是一个。然后，我们在来自教师的软标签上训练蒸馏模型，而不是在训练集的硬标签上。蒸馏有可能提高测试集的准确性，以及较小模型学习预测硬标签的速度 [19]，[30]。

Defensive distillation uses distillation in order to increase the robustness of a neural network, but with two significant changes. First, both the teacher model and the distilled model are identical in size - defensive distillation does not result in smaller models. Second, and more importantly, defensive distillation uses a large distillation temperature (described below) to force the distilled model to become more confident in its predictions.

防御蒸馏使用蒸馏来提高神经网络的鲁棒性，但有两个显著的变化。首先，教师模型和蒸馏模型在大小上是相同的——防御蒸馏不会导致更小的模型。其次，更重要的是，防御蒸馏使用较大的蒸馏温度 (如下所述) 来迫使蒸馏模型在其预测中变得更加自信。

Recall that, the softmax function is the last layer of a neural network. Defensive distillation modifies the softmax function to also include a temperature constant T :

请记住，softmax 函数是神经网络的最后一层。防御性蒸馏修改了 softmax 函数，使其还包括一个温度常数 T :

$$\operatorname{softmax}(x, T)_i = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}$$

It is easy to see that $\operatorname{softmax}(x,T) = \operatorname{softmax}(x/T,1)$. Intuitively, increasing the temperature causes a "softer" maximum, and decreasing it causes a "harder" maximum. As the limit of the temperature goes to 0, softmax approaches max; as the limit goes to infinity, $\operatorname{softmax}(x)$ approaches a uniform distribution.

很容易看出 $\operatorname{softmax}(x,T) = \operatorname{softmax}(x/T,1)$ 。直观上，增加温度会导致“更软”的最大值，而减少温度则会导致“更硬”的最大值。当温度的极限趋近于 0 时，softmax 接近 max；当极限趋近于无穷大时，$\operatorname{softmax}(x)$ 接近均匀分布。

Defensive distillation proceeds in four steps:

防御性蒸馏分为四个步骤:

1) Train a network, the teacher network, by setting the temperature of the softmax to T during the training phase.

1) 通过在训练阶段将 softmax 的温度设置为 T 来训练一个网络，即教师网络。

2) Compute soft labels by apply the teacher network to each instance in the training set, again evaluating the softmax at temperature T .

2) 通过将教师网络应用于训练集中的每个实例来计算软标签，再次在温度 T 下评估 softmax。

3) Train the distilled network (a network with the same shape as the teacher network) on the soft labels, using softmax at temperature T .

3) 在软标签上训练蒸馏网络 (形状与教师网络相同的网络)，使用温度 T 的 softmax。

4) Finally, when running the distilled network at test time (to classify new inputs), use temperature 1.

4) 最后，在测试时运行蒸馏网络 (以分类新输入)，使用温度 1。

A. Fragility of existing attacks

A. 现有攻击的脆弱性

We briefly investigate the reason that existing attacks fail on distilled networks, and find that existing attacks are very fragile and can easily fail to find adversarial examples even when they exist.

我们简要调查了现有攻击在蒸馏网络上失败的原因，发现现有攻击非常脆弱，即使存在对抗样本，也很容易无法找到。

L-BFGS and Deepfool fail due to the fact that the gradient of $F(\cdot)$ is zero almost always, which prohibits the use of the standard objective function.

L-BFGS 和 Deepfool 失败的原因在于 $F(\cdot)$ 的梯度几乎总是为零，这禁止了标准目标函数的使用。

When we train a distilled network at temperature T and then test it at temperature 1, we effectively cause the inputs to the softmax to become larger by a factor of T . By minimizing the cross entropy during training, the output of the softmax is forced to be close to 1.0 for the correct class and 0.0 for all others. Since $Z(\cdot)$ is divided by T , the distilled network will learn to make the $Z(\cdot)$ values T times larger than they otherwise would be. (Positive values are forced to become about T times larger; negative values are multiplied by a factor of about T and thus become even more negative.) Experimentally, we verified this fact: the mean value of the L_1 norm of $Z(\cdot)$ (the logits) on the undistilled network is 5.8 with standard deviation 6.4; on the distilled network (with $T = 100$), the mean is 482 with standard deviation 457.

当我们在温度 T 下训练一个蒸馏网络，然后在温度 1 下测试它时，我们实际上使得 softmax 的输入增大了 T 倍。通过在训练期间最小化交叉熵，softmax 的输出被迫接近正确类别的 1.0 和所有其他类别的 0.0。由于 $Z(\cdot)$ 被 T 除以，蒸馏网络将学习使得 $Z(\cdot)$ 的值比原本大 T 倍。(正值被迫变得大约 T 倍；负值则被乘以大约 T 的因子，因此变得更加负。) 通过实验，我们验证了这一事实: 未蒸馏网络上 $Z(\cdot)$ (logits) 的 L_1 范数的均值为 5.8，标准差为 6.4；而在蒸馏网络上 (使用 $T = 100$)，均值为 482，标准差为 457。

Because the values of $Z(\cdot)$ are 100 times larger, when we test at temperature 1, the output of F becomes ϵ in all components except for the output class which has confidence $1 - 9\epsilon$ for some very small ϵ (for tasks with 10 classes). In fact, in most cases, ϵ is so small that the 32-bit floating-point value is rounded to 0 . For similar reasons, the gradient is so small that it becomes 0 when expressed as a 32-bit floating-point value.

因为 $Z(\cdot)$ 的值大了 100 倍，当我们在温度 1 下测试时，除了输出类别 (其置信度为某个非常小的 ϵ ，对于 10 类任务) 外，所有组件的 F 输出都变为 ϵ 。事实上，在大多数情况下，ϵ 是如此之小，以至于 32 位浮点值被四舍五入为 0。出于类似的原因，梯度也非常小，以至于在以 32 位浮点值表示时变为 0。

This causes the L-BFGS minimization procedure to fail to make progress and terminate. If instead we run L-BFGS with our stable objective function identified earlier, rather than the objective function $\text{loss}_{F,l}(\cdot)$ suggested by Szegedy et al. [46], L-BFGS does not fail. An alternate approach to fixing the attack would be to set

这导致 L-BFGS 最小化过程无法取得进展并终止。如果我们使用之前确定的稳定目标函数来运行 L-BFGS，而不是 Szegedy 等人 [46] 提出的目标函数 $\text{loss}_{F,l}(\cdot)$ ，则 L-BFGS 不会失败。修复攻击的另一种方法是设置

$$F'(x) = \text{softmax}(Z(x)/T)$$

where T is the distillation temperature chosen. Then minimizing $\text{loss}_{F',l}(\cdot)$ will not fail, as now the gradients do not vanish due to floating-point arithmetic rounding. This clearly demonstrates the fragility of using the loss function as the objective to minimize.

其中 T 是选择的蒸馏温度。然后最小化 $\text{loss}_{F',l}(\cdot)$ 将不会失败，因为现在梯度不会因浮点运算舍入而消失。这清楚地表明了将损失函数作为最小化目标的脆弱性。

JSMA-F (whereby we mean the attack uses the output of the final layer $F(\cdot)$) fails for the same reason that L-BFGS fails: the output of the $Z(\cdot)$ layer is very large and so softmax becomes essentially a hard maximum. This is the version of the attack that Papernot et al. use to attack defensive distillation in their paper [39].

JSMA-F(我们指的是攻击使用最终层的输出 $F(\cdot)$) 失败的原因与 L-BFGS 失败的原因相同: $Z(\cdot)$ 层的输出非常大，因此 softmax 实际上变成了一个硬最大值。这是 Papernot 等人在他们的论文 [39] 中用来攻击防御蒸馏的攻击版本。

JSMA-Z (the attack that uses the logits) fails for a completely different reason. Recall that in the $Z(\cdot)$ version of the attack, we use the input to the softmax for computing the gradient instead of the final output of the network. This removes any potential issues with the gradient vanishing, however this introduces new issues. This version of the attack is introduced by Papernot et al. [38] but it is not used to attack distillation; we provide here an analysis of why it fails.

JSMA-Z(使用 logits 的攻击) 失败的原因完全不同。回想一下，在攻击的 $Z(\cdot)$ 版本中，我们使用 softmax 的输入来计算梯度，而不是网络的最终输出。这消除了梯度消失的潜在问题，但引入了新的问题。Papernot 等人 [38] 引入了这种攻击版本，但它并未用于攻击蒸馏；我们在此提供了它失败的原因分析。

Since this attack uses the Z values, it is important to realize the differences in relative impact. If the smallest input to the softmax layer is -100, then, after the softmax layer, the corresponding output becomes practically zero. If this input changes from -100 to -90, the output will still be practically zero. However, if the largest input to the softmax layer is 10, and it changes to 0, this will have a massive impact on the softmax output.

由于该攻击使用了 Z 值，因此重要的是要意识到相对影响的差异。如果 softmax 层的最小输入为 -100，那么在经过 softmax 层后，相应的输出几乎变为零。如果该输入从 -100 改变为 -90，输出仍然几乎为零。然而，如果 softmax 层的最大输入为 10，并且它改变为 0，这将对 softmax 输出产生巨大影响。

Relating this to parameters used in their attack, α and β represent the size of the change at the input to the softmax layer. It is perhaps surprising that JSMA-Z works on undistilled networks, as it treats all changes as being of equal importance, regardless of how much they change the softmax output. If changing a single pixel would increase the target class by 10, but also increase the least likely class by 15, the attack will not increase that pixel.

将此与其攻击中使用的参数相关联，α 和 β 表示 softmax 层输入变化的大小。令人惊讶的是，JSMA-Z 在未蒸馏的网络上也能工作，因为它将所有变化视为同等重要，而不管它们对 softmax 输出的影响有多大。如果改变一个像素会使目标类别增加 10，但也会使最不可能的类别增加 15，那么攻击将不会增加该像素。

Recall that distillation at temperature T causes the value of the logits to be T times larger. In effect, this magnifies the sub-optimality noted above as logits that are extremely unlikely but have slight variation can cause the attack to refuse to make any changes.

回想一下，在温度 T 下的蒸馏使 logits 的值变为 T 倍。实际上，这放大了上述提到的次优性，因为极不可能但略有变化的 logits 会导致攻击拒绝进行任何更改。

Fast Gradient Sign fails at first for the same reason L-BFGS fails: the gradients are almost always zero. However, something interesting happens if we attempt the same division trick and divide the logits by T before feeding them to the softmax function: distillation still remains effective [36]. We are unable to explain this phenomenon.

快速梯度符号方法最初失败的原因与 L-BFGS 失败的原因相同: 梯度几乎总是为零。然而，如果我们尝试相同的除法技巧，并在将 logits 输入 softmax 函数之前将其除以 T ，则会发生一些有趣的事情: 蒸馏仍然有效 [36]。我们无法解释这一现象。

B. Applying Our Attacks

B. 应用我们的攻击

When we apply our attacks to defensively distilled networks, we find distillation provides only marginal value. We re-implement defensive distillation on MNIST and CIFAR-10 as described [39] using the same model we used for our evaluation above. We train our distilled model with temperature $T = 100$, the value found to be most effective [39].

当我们将攻击应用于防御性蒸馏网络时，我们发现蒸馏仅提供了边际价值。我们在 MNIST 和 CIFAR-10 上重新实现了防御性蒸馏，如 [39] 所述，使用我们在上述评估中使用的相同模型。我们以温度 $T = 100$ 训练我们的蒸馏模型，该值被发现是最有效的 [39]。

Table VI shows our attacks when applied to distillation. All of the previous attacks fail to find adversarial examples. In contrast, our attack succeeds with 100% success probability for each of the three distance metrics.

表 VI 显示了我们在蒸馏应用中的攻击。所有先前的攻击都未能找到对抗样本。相比之下，我们的攻击在每个距离度量上都以 100% 的成功概率取得成功。

When compared to Table IV, distillation has added almost no value: our L_0 and L_2 attacks perform slightly worse, and our L_∞ attack performs approximately equally. All of our attacks succeed with 100% success.

与表 IV 相比，蒸馏几乎没有增加价值: 我们的 L_0 和 L_2 攻击表现略差，而我们的 L_∞ 攻击表现大致相同。我们的所有攻击都以 100% 的成功率取得成功。

C. Effect of Temperature

C. 温度的影响

In the original work, increasing the temperature was found to consistently reduce attack success rate. On MNIST, this goes from a 91% success rate at $T = 1$ to a 24% success rate for $T = 5$ and finally 0.5% success at $T = 100$.

在原始工作中，发现增加温度会持续降低攻击成功率。在 MNIST 上，这一成功率从 91% 在 $T = 1$ 时的成功率下降到 24% 在 $T = 5$ 时的成功率，最后在 0.5% 时达到 $T = 100$ 的成功率。

	Best Case				Average Case				Worst Case			
	MNIST		CIFAR		MNIST		CIFAR		MNIST		CIFAR	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
Our L_0	10	100%	7.4	100%I	19	100%	15	100%I	36	100%	29	100%
Our L_2	1.7	100%	0.36	100%I	2.2	100%	0.60	100%|	2.9	100%	0.92	100%
Our L_∞	0.14	100%	0.002	100%I	0.18	100%	0.023	100%|	0.25	100%	0.038	100%

	最佳情况				平均情况				最坏情况			
	MNIST		CIFAR		MNIST		CIFAR		MNIST		CIFAR	
	均值	概率	均值	概率	均值	概率	均值	概率	均值	概率	均值	概率
我们的 L_0	10	100%	7.4	100%I	19	100%	15	100%I	36	100%	29	100%
我们的 L_2	1.7	100%	0.36	100%I	2.2	100%	0.60	100%|	2.9	100%	0.92	100%
我们的 L_∞	0.14	100%	0.002	100%I	0.18	100%	0.023	100%|	0.25	100%	0.038	100%

TABLE VI

Comparison of our attacks when applied to defensively distilled networks. Compare to Table IV for undistilled networks.

我们的攻击在防御性蒸馏网络上的比较。与未蒸馏网络的表 IV 进行比较。

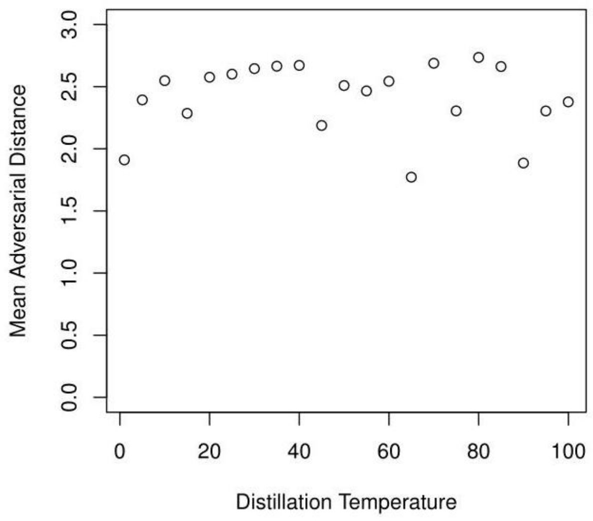


Fig. 8. Mean distance to targeted (with random target) adversarial examples for different distillation temperatures on MNIST. Temperature is uncorrelated with mean adversarial example distance.

图 8. 在 MNIST 上不同蒸馏温度下，针对 (随机目标) 对抗样本的平均距离。温度与平均对抗样本距离无关。

Fig. 9. Probability that adversarial examples transfer from one model to another, for both targeted (the adversarial class remains the same) and untargeted (the image is not the correct class).

图 9. 对抗样本从一个模型转移到另一个模型的概率，包括针对性 (对抗类别保持不变) 和非针对性 (图像不是正确类别)。

We re-implement this experiment with our improved attacks to understand how the choice of temperature impacts robustness. We train models with the temperature varied from $t = 1$ to $t = 100$.

我们使用改进的攻击重新实现此实验，以了解温度选择如何影响鲁棒性。我们训练的模型温度从 $t = 1$ 变化到 $t = 100$ 。

When we re-run our implementation of JSMA, we observe the same effect: attack success rapidly decreases. However, with our improved L_2 attack, we see no effect of temperature on the mean distance to adversarial examples: the correlation coefficient is $\rho = -0.05$. This clearly demonstrates the fact that increasing the distillation temperature does not increase the robustness of the neural network, it only causes existing attacks to fail more often.

当我们重新运行 JSMA 的实现时，我们观察到相同的效果: 攻击成功率迅速下降。然而，通过我们改进的 L_2 攻击，我们发现温度对对抗样本的平均距离没有影响: 相关系数为 $\rho = -0.05$ 。这清楚地表明，增加蒸馏温度并不会提高神经网络的鲁棒性，它只是导致现有攻击更频繁地失败。

D. Transferability

D. 可转移性

Recent work has shown that an adversarial example for one model will often transfer to be an adversarial on a different model, even if they are trained on different sets of training data [46], [11], and even if they use entirely different algorithms (i.e., adversarial examples on neural networks transfer to random forests [37]).

最近的研究表明，一个模型的对抗样本通常会转移为另一个模型的对抗样本，即使它们是在不同的训练数据集上训练的 [46]，[11]，并且即使它们使用完全不同的算法 (即，神经网络上的对抗样本可以转移到随机森林 [37])。

Therefore, any defense that is able to provide robustness against adversarial examples must somehow break this transferability property; otherwise, we could run our attack algorithm on an easy-to-attack model, and then transfer those adversarial examples to the hard-to-attack model.

因此，任何能够提供对抗样本鲁棒性的防御必须以某种方式打破这种可转移性特性；否则，我们可以在一个易于攻击的模型上运行我们的攻击算法，然后将这些对抗样本转移到一个难以攻击的模型上。

Even though defensive distillation is not robust to our stronger attacks, we demonstrate a second break of distillation by transferring attacks from a standard model to a defensively distilled model.

尽管防御性蒸馏对我们更强的攻击不具鲁棒性，但我们通过将攻击从标准模型转移到防御性蒸馏模型展示了第二次蒸馏的突破。

We accomplish this by finding high-confidence adversarial examples, which we define as adversarial examples that are strongly misclassified by the original model. Instead of looking for an adversarial example that just barely changes the classification from the source to the target, we want one where the target is much more likely than any other label.

我们通过寻找高置信度的对抗样本来实现这一点，我们将其定义为被原始模型强烈错误分类的对抗样本。我们希望找到一个目标标签的可能性远高于任何其他标签的对抗样本，而不是仅仅寻找一个微弱改变分类的对抗样本。

Recall the loss function defined earlier for L_2 attacks:

回顾之前为 L_2 攻击定义的损失函数:

$$f(x') = \max\left(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa\right).$$

The purpose of the parameter κ is to control the strength of adversarial examples: the larger κ , the stronger the classification of the adversarial example. This allows us to generate high-confidence adversarial examples by increasing κ .

参数 κ 的目的是控制对抗样本的强度: κ 越大，对抗样本的分类越强。这使我们能够通过增加 κ 来生成高置信度的对抗样本。

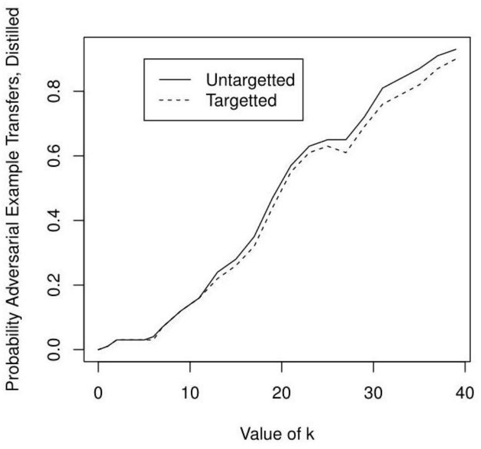


Fig. 10. Probability that adversarial examples transfer from the baseline model to a model trained with defensive distillation at temperature 100 .

图 10. 对抗样本从基线模型转移到以温度 100 进行防御蒸馏训练的模型的概率。

We first investigate if our hypothesis is true that the stronger the classification on the first model, the more likely it will transfer. We do this by varying κ from 0 to 40 .

我们首先调查我们的假设是否成立，即第一个模型的分类越强，它转移的可能性就越大。我们通过将 κ 从 0 变化到 40 来进行此验证。

Our baseline experiment uses two models trained on MNIST as described in Section IV, with each model trained on half of the training data. We find that the transferability success rate increases linearly from $\kappa = 0$ to $\kappa = 20$ and then plateaus at near- 100% success for $\kappa \approx 20$, so clearly increasing κ increases the probability of a successful transferable attack.

我们的基线实验使用了两个在 MNIST 上训练的模型，如第 IV 节所述，每个模型训练了一半的训练数据。我们发现，转移成功率从 $\kappa = 0$ 线性增加到 $\kappa = 20$ ，然后在接近 100% 的成功率下趋于平稳，对于 $\kappa \approx 20$ 来说，因此显然增加 κ 会增加成功可转移攻击的概率。

We then run this same experiment only instead we train the second model with defensive distillation, and find that adversarial examples do transfer. This gives us another attack technique for finding adversarial examples on distilled networks.

然后我们进行相同的实验，只是我们用防御蒸馏训练第二个模型，发现对抗样本确实会转移。这为我们在蒸馏网络上寻找对抗样本提供了另一种攻击技术。

However, interestingly, the transferability success rate between the unsecured model and the distilled model only reaches 100% success at $\kappa = 40$, in comparison to the previous approach that only required $\kappa = 20$.

然而，有趣的是，未加固模型与蒸馏模型之间的转移成功率仅在 100% 达到 $\kappa = 40$ 的成功率，而与之前的方法相比，后者仅需 $\kappa = 20$ 。

We believe that this approach can be used in general to evaluate the robustness of defenses, even if the defense is able to completely block flow of gradients to cause our gradient-descent based approaches from succeeding.

我们相信这种方法可以普遍用于评估防御的鲁棒性，即使防御能够完全阻止梯度流动，导致我们的基于梯度下降的方法无法成功。

IX. CONCLUSION

IX. 结论

The existence of adversarial examples limits the areas in which deep learning can be applied. It is an open problem to construct defenses that are robust to adversarial examples. In an attempt to solve this problem, defensive distillation was proposed as a general-purpose procedure to increase the robustness of an arbitrary neural network.

对抗样本的存在限制了深度学习的应用领域。构建对抗样本具有鲁棒性的防御是一个未解决的问题。为了解决这个问题，提出了防御蒸馏作为一种通用程序，以提高任意神经网络的鲁棒性。

In this paper, we propose powerful attacks that defeat defensive distillation, demonstrating that our attacks more generally can be used to evaluate the efficacy of potential defenses. By systematically evaluating many possible attack approaches, we settle on one that can consistently find better adversarial examples than all existing approaches. We use this evaluation as the basis of our three L_0, L_2 , and L_∞ attacks.

在本文中，我们提出了强大的攻击方法，能够击败防御性蒸馏，证明我们的攻击可以更广泛地用于评估潜在防御的有效性。通过系统地评估许多可能的攻击方法，我们确定了一种能够持续找到比所有现有方法更好的对抗样本的方法。我们将这种评估作为我们三种 L_0, L_2 和 L_∞ 攻击的基础。

We encourage those who create defenses to perform the two evaluation approaches we use in this paper:

我们鼓励那些创建防御的人采用本文中使用的两种评估方法:

- Use a powerful attack (such as the ones proposed in this paper) to evaluate the robustness of the secured model directly. Since a defense that prevents our L_2 attack will prevent our other attacks, defenders should make sure to establish robustness against the L_2 distance metric.
- 使用一种强大的攻击 (例如本文中提出的攻击) 直接评估安全模型的鲁棒性。由于防御能够阻止我们的 L_2 攻击将能够阻止我们的其他攻击，因此防御者应确保在 L_2 距离度量上建立鲁棒性。
- Demonstrate that transferability fails by constructing high-confidence adversarial examples on a unsecured model and showing they fail to transfer to the secured model.
- 通过在一个未安全的模型上构建高置信度的对抗样本并展示其无法转移到安全模型上，来证明可转移性失败。

ACKNOWLEDGEMENTS

致谢

We would like to thank Nicolas Papernot discussing our defensive distillation implementation, and the anonymous reviewers for their helpful feedback. This work was supported by Intel through the ISTC for Secure Computing, Qualcomm, Cisco, the AFOSR under MURI award FA9550-12-1-0040, and the Hewlett Foundation through the Center for Long-Term Cybersecurity.

我们要感谢 Nicolas Papernot 讨论我们的防御性蒸馏实现，以及匿名评审者提供的有益反馈。此项工作得到了英特尔通过安全计算 ISTC、Qualcomm、思科、AFOSR 在 MURI 奖项 FA9550-12-1-0040 下的支持，以及休利特基金会通过长期网络安全中心的支持。

REFERENCES

参考文献

[1] Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., GANCHEV, K., PETROV, S., AND COLLINS, M. Globally normalized transition-based neural networks. arXiv preprint arXiv:1603.06042 (2016).

[2] BASTANI, O., IOANNOU, Y., LAMPROPOULOS, L., VYTINIOTIS, D., NORI, A., AND CRIMINISI, A. Measuring neural net robustness with constraints. arXiv preprint arXiv:1605.07262 (2016). FLEPP, B., Goyal, P., Jackel, L. D., Monfort, M., Mulller, U., ZHANG, J., ET AL. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016).

[4] BOURZAC, K. Bringing big neural networks to self-driving cars, smartphones, and drones. http://spectrum.ieee.org/computing/embedded-systems/
bringing-big-neural-networks-to-selfdriving-cars-smartphones-and-drones, 2016.

[5] Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., SHIELDS, C., WAGNER, D., AND ZHOU, W. Hidden voice commands. In 25th USENIX Security Symposium (USENIX Security 16), Austin, TX (2016).

[6] CHANDOLA, V., BANERJEE, A., AND KUMAR, V. Anomaly detection: A survey. ACM computing surveys (CSUR) 41, 3 (2009), 15.

[7] CLEVERT, D.-A., UNTERTHINER, T., AND HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (ELUs). arXiv preprint arXiv:1511.07289 (2015).

[8] DAHL, G. E., StoKES, J. W., DENG, L., AND YU, D. Large-scale malware classification using random projections and neural networks. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (2013), IEEE, pp. 3422-3426.

[9] Deng, J., Dong, W., Socher, R., Li, L.-J., LI, K.-J., LI, K., And FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (2009), IEEE, pp. 248-255.

[10] Giusti, A., Guzzi, J., Cireşan, D. C., He, F.-L., Rodríguez, J. P., FONTANA, F., FAESSLER, M., FORSTER, C., SCHMIDHUBER, J., DI CARO, G., ET AL. A machine learning approach to visual perception of forest trails for mobile robots. IEEE Robotics and Automation Letters 1, 2 (2016), 661-667.

[11] GOODFELLOW, I. J., SHLENS, J., AND SZEGEDY, C. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014).

[12] GRAHAM, B. Fractional max-pooling. arXiv preprint arXiv:1412.6071 (2014).

[13] GRAVES, A., MOHAMED, A.-R., AND HINTON, G. Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (2013), IEEE, pp. 6645-6649.

[14] GROSSE, K., PAPERNOT, N., MANOHARAN, P., BACKES, M., AND MCDANIEL, P. Adversarial perturbations against deep neural networks for malware classification. arXiv preprint arXiv:1606.04435 (2016).

[15] GU, S., AND RIGAZIO, L. Towards deep neural network architectures robust to adversarial examples. arXiv preprint arXiv:1412.5068 (2014).

[16] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016), pp. 770-778.

[17] Hinton, G., Deng, L., Yu, D., Dahl., G., Rahman Mohamed, A., JAITLY, N., SENIOR, A., VANHOUCKE, V., NGUYEN, P., SAINATH, T., AND KINGSBURY, B. Deep neural networks for acoustic modeling in speech recognition. Signal Processing Magazine (2012).

18] Hinton, G., Deng, L., Yu, D., Dahl., G. E., MohamED, A.-R., T. N., ET AL. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine 29, 6 (2012), 82-97.

[19] HINTON, G., VINYALS, O., AND DEAN, J. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015).

[20] Huang, R., Xu, B., Schuurmans, D., AND SZEPESVÁRI, C. Learning with a strong adversary. CoRR, abs/1511.03034 (2015).

[21] HUANG, X., KWIATKOWSKA, M., WANG, S., AND WU, M. Safety verification of deep neural networks. arXiv preprint arXiv:1610.06940 (2016).

[22] JANGLOVÁ, D. Neural networks in mobile robot motion. Cutting Edge Robotics 1, 1 (2005), 243.

[23] KINGMA, D., AND BA, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[24] KRIZHEVSKY, A., AND HINTON, G. Learning multiple layers of features from tiny images.

[25] Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In Advances

[26] KURAKIN, A., GOODFELLOW, I., AND BENGIO, S. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016).

[27] LeCUN, Y., Bottou, L., Bengio, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 11 (1998), 2278-2324.

[28] LECUN, Y., CORTES, C., AND BURGES, C. J. The mnist database of handwritten digits, 1998.

[29] MAAS, A. L., HANNUN, A. Y., AND NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In Proc. ICML (2013), vol. 30.

[30] Melicher, W., Ur, B., SEGRETI, S. M., Komanduri, S., Bauer, L., CHRISTIN, N., AND CRANOR, L. F. Fast, lean and accurate: Modeling password guessability using neural networks. In Proceedings of USENIX Security (2016).

[31] MISHKIN, D., AND MATAS, J. All you need is a good init. arXiv preprint arXiv:1511.06422 (2015).

[32] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., ANTONOGLOU, I., WIERSTRA, D., AND RIEDMILLER, M. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).

[33] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., BELLEMARE, M. G., GRAVES, A., RIEDMILLER, M., FIDJELAND, A. K., OSTROVSKI, G., ET AL. Human-level control through deep reinforcement learning. Nature 518, 7540 (2015), 529-533.

[34] MOOSAVI-DEZFOOLI, S.-M., FAWZI, A., AND FROSSARD, P. Deep-fool: a simple and accurate method to fool deep neural networks. arXiv preprint arXiv:1511.04599 (2015).

[35] PAPERNOT, N., GOODFELLOW, I., SHEATSLEY, R., FEINMAN, R., AND MCDANIEL, P. cleverhans v1.0.0: an adversarial machine learning library. arXiv preprint arXiv:1610.00768 (2016).

[36] PAPERNOT, N., AND MCDANIEL, P. On the effectiveness of defensive distillation. arXiv preprint arXiv:1607.05113 (2016).

[37] PAPERNOT, N., MCDANIEL, P., AND GOODFELLOW, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277 (2016).

[38] PAPERNOT, N., MCDANIEL, P., JHA, S., FREDRIKSON, M., CELIK, settings. In 2016 IEEE European Symposium on Security and Privacy (EuroS&P) (2016), IEEE, pp. 372-387.

[39] PAPERNOT, N., MCDANIEL, P., WU, X., JHA, S., AND SWAMI, A. Distillation as a defense to adversarial perturbations against deep neural networks. IEEE Symposium on Security and Privacy (2016).

[40] PASCANU, R., STOKES, J. W., SANOSSIAN, H., MARINESCU, M., AND THOMAS, A. Malware classification with recurrent networks. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2015), IEEE, pp. 1916-1920.

[41] RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., BERG, A. C., AND FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115, 3 (2015), 211-252.

[42] SHAHAM, U., YAMADA, Y., AND NEGAHBAN, S. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. arXiv preprint arXiv:1511.05432 (2015).

[43] SILVER, D., HUANG, A., MADDISON, C. J., GUEZ, A., SIFRE, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., PANNEERSHELVAM, V., LANCTOT, M., ET AL. Mastering the game of Go with deep neural networks and tree search. Nature 529, 7587 (2016), 484-489.

[44] Springenberg, J. T., Dosovitskiy, A., Brox, T., And Ried-MILLER, M. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806 (2014).

[45] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the Inception architecture for computer vision. arXiv preprint arXiv:1512.00567 (2015).

[46] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Bruna, J., Erhan, D., GOODFELLOW, I., AND FERGUS, R. Intriguing properties of neural networks. ICLR (2013).

[47] WARDE-FARLEY, D., AND GOODFELLOW, I. Adversarial perturbations of deep neural networks. Advanced Structured Prediction, T. Hazan, G. Papandreou, and D. Tarlow, Eds (2016).

[48] YUAN, Z., Lu, Y., WANG, Z., AND XUE, Y. Droid-sec: Deep learning in android malware detection. In ACM SIGCOMM Computer Communication Review (2014), vol. 44, ACM, pp. 371-372.

APPENDIX

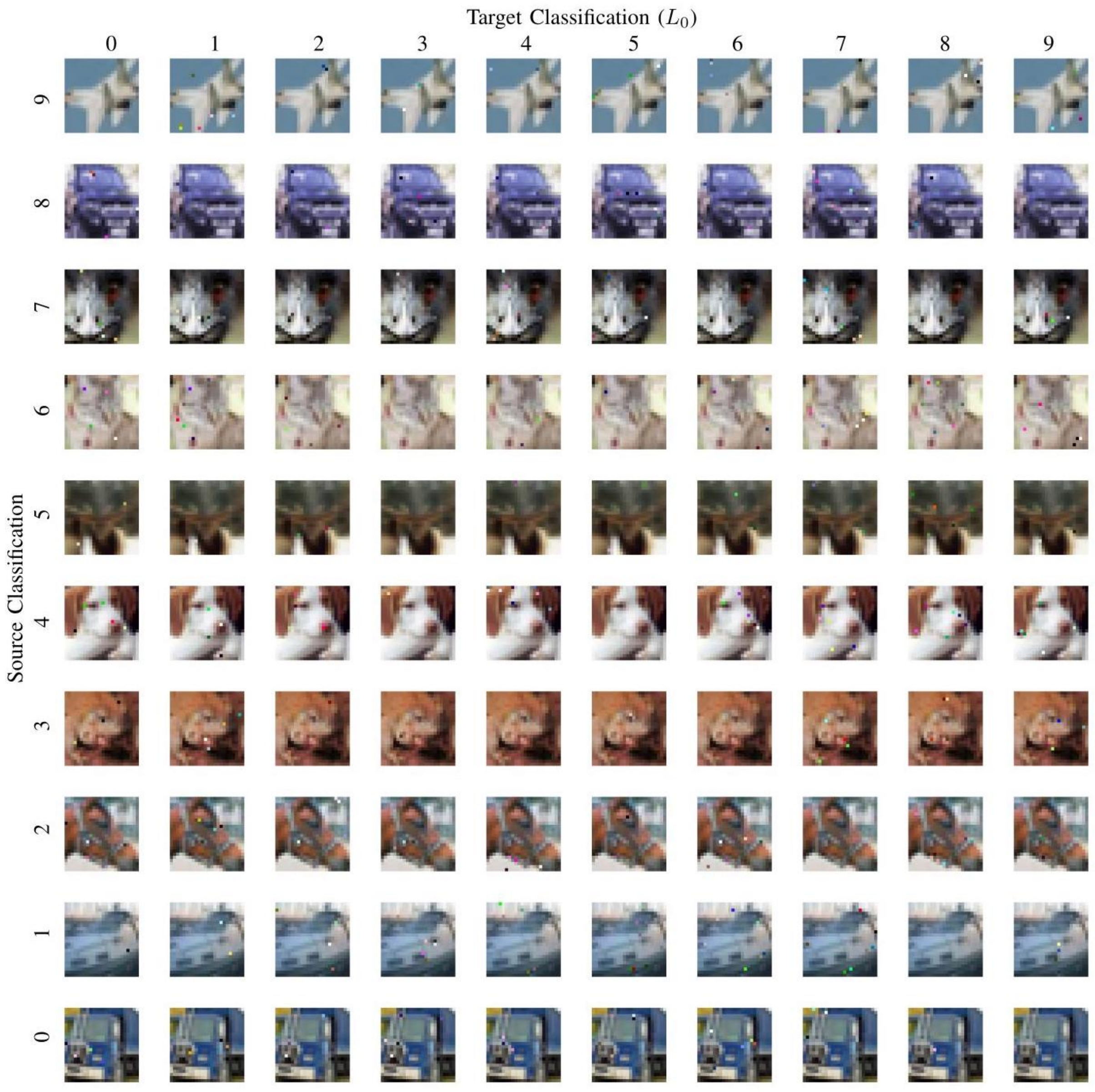


Fig. 11. Our L_0 adversary applied to the CIFAR dataset performing a targeted attack for every source/target pair. Each image is the first image in the dataset with that label.

图 11. 我们的 L_0 对手应用于 CIFAR 数据集，对每个源/目标对执行有针对性的攻击。每张图像是数据集中具有该标签的第一张图像。

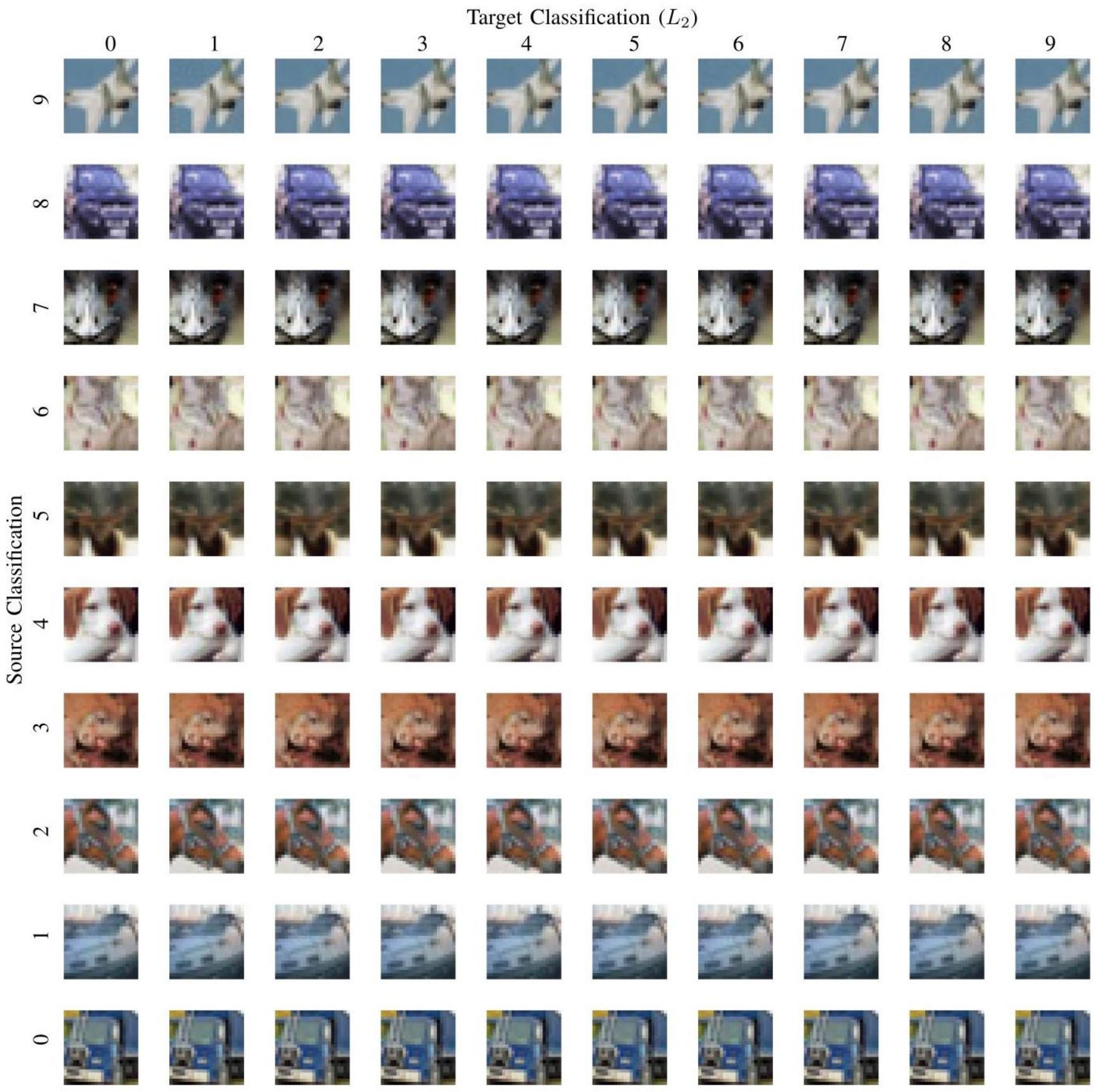


Fig. 12. Our L_2 adversary applied to the CIFAR dataset performing a targeted attack for every source/target pair. Each image is the first image in the dataset with that label.

图 12. 我们的 L_2 对手应用于 CIFAR 数据集，对每个源/目标对执行有针对性的攻击。每张图像是数据集中具有该标签的第一张图像。

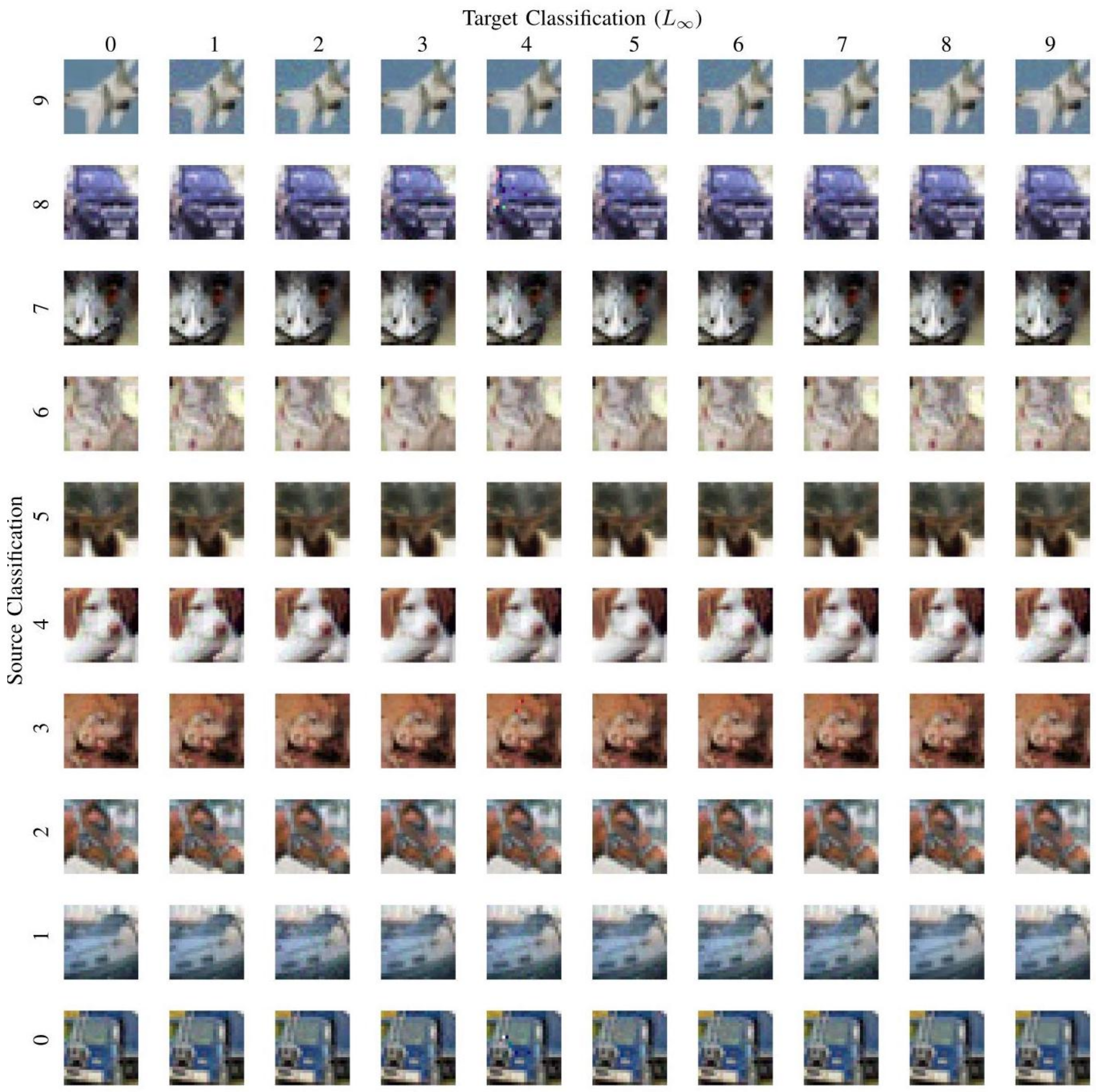


Fig. 13. Our L_∞ adversary applied to the CIFAR dataset performing a targeted attack for every source/target pair. Each image is the first image in the dataset with that label.

图 13. 我们的 L_∞ 对手应用于 CIFAR 数据集，对每个源/目标对执行有针对性的攻击。每张图像是数据集中具有该标签的第一张图像。