EVALUATING THE ROBUSTNESS OF NEURAL NETWORKS: AN EXTREME VALUE THEORY APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

The robustness of neural networks to adversarial examples has received great attention due to security implications. Despite various attack approaches to crafting visually imperceptible adversarial examples, little has been developed towards a comprehensive measure of robustness. In this paper, we provide theoretical justification for converting robustness analysis into a local Lipschitz constant estimation problem, and propose to use the Extreme Value Theory for efficient evaluation. Our analysis yields a novel robustness metric called CLEVER, which is short for **C**ross **L**ipschitz **E**xtreme **V**alue for n**E**twork **R**obustness. The proposed CLEVER score is attack-agnostic and is computationally feasible for large neural networks. Experimental results on various networks, including ResNet, Inception-v3 and MobileNet, show that (i) CLEVER is aligned with the robustness indication measured by the ℓ_2 and ℓ_∞ norms of adversarial examples from powerful attacks, and (ii) defended networks using defensive distillation or bounded ReLU indeed give better CLEVER scores. To the best of our knowledge, CLEVER is the first attack-independent robustness metric that can be applied to any neural network classifier.

1 INTRODUCTION

Recent studies have highlighted the lack of robustness in state-of-the-art neural network models, e.g., a visually imperceptible adversarial image can be easily crafted to mislead a well-trained network (Szegedy et al., 2013; Goodfellow et al., 2015). Even worse, researchers have identified that these adversarial examples are not only valid in the digital space but also plausible in the physical world (Kurakin et al., 2016a; Evtimov et al., 2017). The vulnerability to adversarial examples call into question safety-critical applications and services deployed by neural networks, including autonomous driving systems and malware detection protocols, among others.

In the literature, studying adversarial examples of neural networks has twofold purposes: (i) *security implications*: devising effective attack algorithms for crafting adversarial examples; and (ii) *robustness analysis*: evaluating the intrinsic model robustness to adversarial perturbations to normal examples. Although in principle the means of tackling these two problems are expected to be independent, that is, the evaluation of a neural network's intrinsic robustness should be agnostic to attack methods, and vice versa, existing approaches extensively use different attack results as a measure of robustness of a target neural network. Specifically, given a set of normal examples, the attack success rate and distortion of the corresponding adversarial examples crafted from a particular attack algorithm are treated as robustness metrics. Consequently, the network robustness is entangled with the attack algorithms used for evaluation and the analysis is limited by the attack capabilities. More importantly, the dependency between robustness evaluation and attack approaches can cause biased analysis. For example, *adversarial training* is a commonly used technique for improving the robustness of a neural network; it can be accomplished by generating adversarial examples and retraining the network parameters with corrected labels. However, while such an adversarially trained network is made robust to attacks used to craft adversarial examples for training, it can still be vulnerable to unseen attacks.

Motivated by the evaluation criterion for assessing the quality of text and image generation that is completely independent of the underlying generative processes, such as the BLEU score for texts (Papineni et al., 2002) and the INCEPTION score for images (Salimans et al., 2016), we

aim to propose a comprehensive and attack-agnostic robustness metric for neural networks. Stemming from a perturbation analysis of an arbitrary neural network classifier, we derive an universal lower bound on the minimal distortion required to craft an adversarial example from an original one, where the lower bound applies to any attack algorithm and any ℓ_p norm for $p \geq 1$. We show that this lower bound associates with the maximum norm of the local gradients with respect to the original example, and therefore robustness evaluation becomes an estimation problem of the local Lipschitz constant. To efficiently and reliably estimate the local Lipschitz constant, we propose to use *extreme value theory* (De Haan & Ferreira, 2007) for robustness evaluation. In this context, the extreme value corresponds to the local Lipschitz constant of our interest, which can be inferred by a set of independently and identically sampled local gradients.With the aid of extreme value theory, we propose a robustness metric called CLEVER, which is short for **C**ross **L**ipschitz **E**xtreme **V**alue for n**E**twork **R**obustness. We note that CLEVER is an attack-independent robustness metric that applies to any neural network classifiers. In contrast, the robustness metric proposed in (Hein & Andriushchenko, 2017), albeit attack-agnostic, only applies to a neural network classifier with one hidden layer.

We highlight the main contributions of this paper as follows:

- We propose a novel robustness metric called CLEVER, which is short for **C**ross **L**ipschitz **E**xtreme **V**alue for n**E**twork **R**obustness. To the best of our knowledge, CLEVER is the first robustness metric that is attack-independent and can be applied to any arbitrary neural network classifier.
- The proposed CLEVER score is well supported by our theoretical analysis on formal robustness guarantees and the use of extreme value theory. Our robustness analysis extends the results in (Hein & Andriushchenko, 2017) from continuously differentiable functions to functions with finite non-differentiable points.
- We corroborate the effectiveness of CLEVER by conducting experiments on state-of-the-art models for ImageNet, including ResNet (He et al., 2016), Inception-v3 (Szegedy et al., 2016) and MobileNet (Howard et al., 2017). We also use CLEVER to investigate defended networks against adversarial examples, including the use of defensive distillation (Papernot et al., 2016) and bounded ReLU (Zantedeschi et al., 2017). Experimental results show that our CLEVER score well aligns with the attack-specific robustness indicated by the ℓ_2 and ℓ_∞ distortions of adversarial examples.

The remainder of the paper is organized as follows. Section 2 provides the background and related work on adversarial attack, defense, and theoretical robustness guarantees. In Section 3, we present a formal robustness analysis for classification models. In Section 4, we propose a novel way to accurately estimate cross Lipschitz constants and introduce the CLEVER score. We present empirical results on various networks in Section 5, and summarize the paper in Section 6.

2 Background and Related Work

2.1 Attacking Neural Networks using Adversarial Examples

One of the most popular formulations found in literature for crafting adversarial examples to mislead a neural network is to formulate it as a minimization problem, where the variable $\boldsymbol{\delta} \in \mathbb{R}^d$ to be optimized refers to the perturbation to the original example, and the objective function takes into account unsuccessful adversarial perturbations as well as a specific norm on $\boldsymbol{\delta}$ for assuring similarity. For instance, the success of adversarial examples can be evaluated by their cross-entropy loss (Szegedy et al., 2013; Goodfellow et al., 2015) or model prediction (Carlini & Wagner, 2017b). The norm constraint on $\boldsymbol{\delta}$ can be implemented in a clipping manner (Kurakin et al., 2016b) or treated as a penalty function (Carlini & Wagner, 2017b). The ℓ_p norm of $\boldsymbol{\delta}$, defined as $\|\boldsymbol{\delta}\|_p = (\sum_{i=1}^d |\boldsymbol{\delta}_i|^p)^{1/p}$ for any $p \geq 1$, is often used for crafting adversarial examples. In particular, when $p = \infty$, $\|\boldsymbol{\delta}\|_\infty = \max_{i \in \{1,\ldots,d\}} |\boldsymbol{\delta}_i|$ measures the maximal variation among all dimensions in $\boldsymbol{\delta}$. When $p = 2$, $\|\boldsymbol{\delta}\|_2$ becomes the Euclidean norm of $\boldsymbol{\delta}$. When $p = 1$, $\|\boldsymbol{\delta}\|_1 = \sum_{i=1}^p |\boldsymbol{\delta}_i|$ measures the total variation of $\boldsymbol{\delta}$. The state-of-the-art attack methods for ℓ_∞, ℓ_2 and ℓ_1 norms are the iterative fast gradient sign method (I-FGSM) (Goodfellow et al., 2015; Kurakin et al., 2016b), Carlini and Wagner's attack (CW attack) (Carlini & Wagner, 2017b), and elastic-net attacks to deep neural networks (EAD) (Chen et al., 2017a), respectively. These attacks fall into the category of *white-box*

attacks since the network model is assumed to be transparent to an attacker. Adversarial examples can also be crafted from a *black-box* network model using an ensemble approach (Liu et al., 2016), training a substitute model (Papernot et al., 2017), or employing zeroth-order optimization based attacks (Chen et al., 2017b).

2.2 Existing Defense Methods

Since the discovery of vulnerability to adversarial examples (Szegedy et al., 2013), various defense methods have been proposed to improve the robustness of neural networks. The rationale for defense is to make a neural network more resilient to adversarial perturbations, while ensuring the resulting defended model still attains similar test accuracy as the original undefended network. Papernot et al. proposed *defensive distillation* (Papernot et al., 2016), which uses the distillation technique (Hinton et al., 2015) and a modified softmax function at the final layer to retrain the parameters of a neural network with the prediction probabilities (i.e., soft labels) from the original network. Zantedeschi et al. showed that by changing the ReLU function to a bounded ReLU function, a neural network can be made more resilient. Another popular defense approach is *adversarial training*, which generates and augments adversarial examples with the original training data during the network training stage. On MNIST, the adversarially trained model proposed by Madry et al. (Madry et al., 2017) can successfully defend a majority of adversarial examples at the price of increased network capacity. In addition to network modification and adversarial training, *detection* methods such as feature squeezing (Xu et al., 2017) can also be used to identify adversarial examples. However, the CW attack is shown to be able to bypass 10 different detection methods (Carlini & Wagner, 2017a). In this paper, we focus on evaluating the intrinsic robustness of a neural network model to adversarial examples. The effect of detection methods is beyond our scope.

2.3 Theoretical Robustness Guarantees for Neural Networks

(Szegedy et al., 2013) compute global Lipschitz constant for each layer and use their product to explain the robustness issue in neural networks. However, using global Lipschitz constant can be impractical when the resulting bound on the distortion is loose. (Hein & Andriushchenko, 2017) gave a robustness lower bound using a local Lipschitz continuous condition and derived a closed-form bound for a multi-layer perceptron (MLP) with a single hidden layer and softplus activation. However, the analysis cannot be extended to a neural network with more than one hidden layer. (Wang et al., 2016) utilized terminologies from topology to study robustness. However, no robustness bounds or estimates were provided for neural networks. On the other hand, works done by (Ehlers, 2017; Katz et al., 2017a;b; Huang et al., 2017) focus on formally verifying the viability of certain properties in neural networks for any possible input, and transform this formal verification problem into satisfiability modulo theory (SMT) and integer linear programming (ILP) problems. However, this verification approach comes with high computational complexity and is only plausible for very small networks.

3 Formal Robustness Analysis for a Classifier

In this section, we formally define the notion of adversarial examples, minimum ℓ_p distortions, and lower/upper bounds. Under a very mild assumption on Lipschitz continuity of the classifier function, we obtain formal robustness guarantees against adversarial perturbations, i.e. the lower bound of minimum ℓ_p distortion. For quick reference, the important notations introduced are summarized in Table 1.

Definition 3.1 (perturbed example and adversarial example)**.** Let $\boldsymbol{x_0} \in \mathbb{R}^d$ be an input vector of a K-class classifier function $f : \mathbb{R}^d \to \mathbb{R}^K$ and the prediction is given as $f(\boldsymbol{x_0}) = \text{argmax}_{1 \le i \le K} f_i(\boldsymbol{x_0})$. Given $\boldsymbol{x_0}$, we say $\boldsymbol{x_a}$ is a *perturbed example* of $\boldsymbol{x_0}$ with noise $\boldsymbol{\delta} \in \mathbb{R}^d$ and ℓ_p-distortion Δ_p if $\boldsymbol{x_a} = \boldsymbol{x_0} + \boldsymbol{\delta}$ and $\Delta_p = \|\boldsymbol{\delta}\|_p$. An *adversarial example* is a perturbed example where we can find some $\boldsymbol{x_a}$ or $\boldsymbol{\delta}$ that can attack the classifier successfully. For un-targeted attack, a successful attack is to find a $\boldsymbol{x_a}$ such that $f(\boldsymbol{x_a}) \neq f(\boldsymbol{x_0})$. For targeted attack, a target class t ($t \neq f(\boldsymbol{x_0})$) is provided and a successful attack satisfies $f(\boldsymbol{x_a}) = t$.

Table 1: Table of Notation

Notation	Definition	Notation	Definition
d	dimensionality of the input vector	$\Delta_{p,\text{min}}$	minimum ℓ_p distortion of $\boldsymbol{x_0}$
K	number of output classes	β_L	lower bound of minimum distortion
$f : \mathbb{R}^d \to \mathbb{R}^K$	neural network classifier	β_U	upper bound of minimum distortion
$\boldsymbol{x_0} \in \mathbb{R}^d$	original input vector	L_q	Lipschitz constant
$\boldsymbol{x_a} \in \mathbb{R}^d$	adversarial example	L_{q,x_0}	local Lipschitz constant
$\boldsymbol{\delta} \in \mathbb{R}^d$	distortion $:= \boldsymbol{x_a} - \boldsymbol{x_0}$	$B_p(\boldsymbol{x_0}, R)$	hyper-ball with center $\boldsymbol{x_0}$ and radius R
$\|\boldsymbol{\delta}\|_p$	ℓ_p norm of distortion, $p \geq 1$	CDF	cumulative distribution function

Definition 3.2 (minimum distortion of $\boldsymbol{x_0}$)**.** Given an input vector $\boldsymbol{x_0}$ of a classifier function f, the *minimum* ℓ_p distortion of $\boldsymbol{x_0}$, denoted as $\Delta_{p,\text{min}}$, is defined as the smallest Δ_p of its adversarial examples.

Definition 3.3 (lower bound of minimum distortion, $\beta_L \leq \Delta_{p,\text{min}}$)**.** Suppose $\Delta_{p,\text{min}}$ is the minimum distortion of $\boldsymbol{x_0}$. A lower bound of $\Delta_{p,\text{min}}$, denoted by β_L, is defined such that any perturbed example of $\boldsymbol{x_0}$ with $\|\boldsymbol{\delta}\|_p \leq \beta_L$ cannot be successful.

Definition 3.4 (upper bound of minimum distortion, $\beta_U \geq \Delta_{p,\text{min}}$)**.** Suppose $\Delta_{p,\text{min}}$ is the minimum distortion of $\boldsymbol{x_0}$. An upper bound of $\Delta_{p,\text{min}}$, denoted by β_U, is defined such that there exists an adversarial example of $\boldsymbol{x_0}$ with $\|\boldsymbol{\delta}\|_p \geq \beta_U$.

The lower and upper bounds are instance specific because they depend on the input $\boldsymbol{x_0}$. Ideally, we would like the bounds to be as close to the minimum distortion of $\boldsymbol{x_0}$ as possible, meaning that β_L is as large as possible and β_U is as small as possible, as they reflect the *robustness* of a classifier. Below we show how to derive a formal robustness guarantee of a classifier with Lipschitz continuity assumption. Specifically, our analysis obtains a lower bound of ℓ_p minimum distortion $\beta_L = \min_{j \neq c} \frac{f_c(\boldsymbol{x_0}) - f_j(\boldsymbol{x_0})}{L_q}$.

Lemma 3.1 (Lipschitz continuity (Paulavičius & Žilinskas, 2006))**.** *Let $S \subset \mathbb{R}^d$ be a convex bounded closed set and let $h(\boldsymbol{x}) : S \to \mathbb{R}$ be a continuously differentiable function on an open set containing S. Then, $h(\boldsymbol{x})$ is a Lipschitz function with Lipschitz constant L_q if the following inequality holds for any $\boldsymbol{x}, \boldsymbol{y} \in S$:*

$$|h(\boldsymbol{x}) - h(\boldsymbol{y})| \leq L_q \|\boldsymbol{x} - \boldsymbol{y}\|_p, \tag{1}$$

where $L_q = \max\{\|\nabla h(\boldsymbol{x})\|_q : \boldsymbol{x} \in S\}, \nabla h(\boldsymbol{x}) = (\frac{\partial h}{\partial x_1}, \cdots, \frac{\partial h}{\partial x_d})^\top$ is the gradient of $h(\boldsymbol{x})$, and $\frac{1}{p} + \frac{1}{q} = 1, 1 \leq p, q \leq \infty$.

Given Lemma 3.1, we then provide a formal guarantee to the lower bound β_L.

Theorem 3.2 (Formal guarantee on lower bound β_L)**.** *Let $\boldsymbol{x_0} \in \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}^K$ be a multi-class classifier with continuously differentiable components f_i and let $c = \operatorname{argmax}_{1 \leq i \leq K} f_i(\boldsymbol{x_0})$ be the class which f predicts for $\boldsymbol{x_0}$. For all $\boldsymbol{\delta} \in \mathbb{R}^d$ with*

$$\|\boldsymbol{\delta}\|_p \leq \min_{j \neq c} \frac{f_c(\boldsymbol{x_0}) - f_j(\boldsymbol{x_0})}{L_q}, \tag{2}$$

$c = \operatorname{argmax}_{1 \leq i \leq K} f_i(\boldsymbol{x_0} + \boldsymbol{\delta})$ holds with $\frac{1}{p} + \frac{1}{q} = 1, 1 \leq p, q \leq \infty$ and L_q is Lipschitz constant for the function $g := f_c - f_j$ in ℓ_p norm. In other words, $\beta_L = \min_{j \neq c} \frac{f_c(\boldsymbol{x_0}) - f_j(\boldsymbol{x_0})}{L_q}$ is a lower bound of minimum distortion.

Intuitions behind Theorem 3.2 is shown in Figure 1, as a one-dimensional example. The function value $g(x)$ near point x_0 is inside the double-cone formed by two lines with slopes equal to $\pm L_q$, where L_q is the (local) Lipschitz constant of $g(x)$. When $g(x)$ is decreased to 0, an adversarial example is found. The minimal change δ to decrease $g(x)$ to 0, in the worst case where $g(x)$ follows the boundary of the double-cone, is $\frac{g(x_0)}{L_q}$. The complete proof is deferred to Appendix A.

We make a few remarks about the theorem.

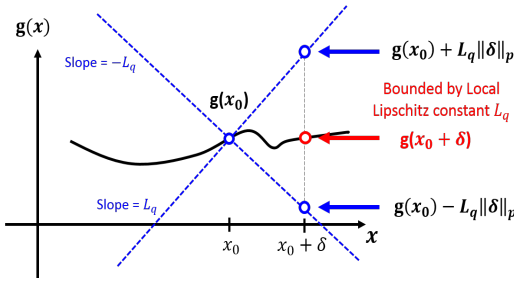


Figure 1: Intuitions behind Theorem 3.2.

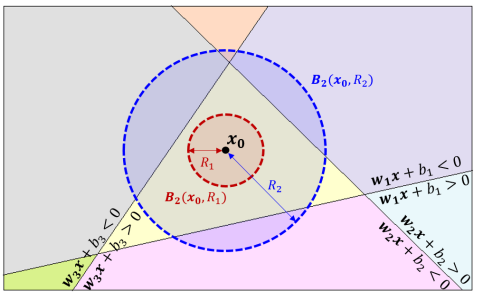


Figure 2: Illustration of Theorem 4.1.

Remark 1. *Because L_q is the Lipschitz constant of the function involving cross terms: $f_c(\boldsymbol{x}) - f_j(\boldsymbol{x})$, we also call it cross Lipschitz constant following (Hein & Andriushchenko, 2017).*

Remark 2. *An upper bound β_U on the minimum distortion $\Delta_{p,min}$ can be easily obtained by any successfully attack, since $\Delta_{p,min}$ is guaranteed to be less than the norm of an adversarial example.*

Corollary 3.2.1. [1] *Let L_{q,x_0} be local Lipschitz constant of function $g := f_c - f_j$ at $\boldsymbol{x_0}$ over some fixed ball $B_p(\boldsymbol{x_0}, R) := \{\boldsymbol{x} \in \mathbb{R}^d \mid \|\boldsymbol{x} - \boldsymbol{x_0}\|_p \leq R\}$ and let $\boldsymbol{\delta} \in B_p(\mathbf{0}, R)$. By Theorem 3.2, we get the bound in (Hein & Andriushchenko, 2017):*

$$\|\boldsymbol{\delta}\|_p \leq \min\left\{\min_{j \neq c} \frac{f_c(\boldsymbol{x_0}) - f_j(\boldsymbol{x_0})}{L_{q,x_0}}, R\right\}. \tag{3}$$

Remark 3. *The analysis in (Hein & Andriushchenko, 2017) implicitly assumes Lipschitz continuity on f_i because they require f_i to be continuously differentiable. Alternatively, here we provide a simple alternative derivation without using Mean Value Theorem and Holder's Inequality.*

To cover non-differentiable functions (a typical property of neural networks owing to ReLU), we consider the following extension.

Lemma 3.3. [2] *Let $S \subset \mathbb{R}^d$ be a convex bounded closed set and let $h(\boldsymbol{x}) : S \to \mathbb{R}$ be an absolute continuous function and the derivative exists at all but a finite number of points (denoted as Z). Then equation (1) holds with $L_q = \sup_{x \in S \setminus Z}\{\|\nabla f(x)\|_q\}$, and we can obtain the same conclusion as Theorem 3.2 and Corollary 3.2.1.*

4 Estimating cross Lipschitz constant via Extreme Value Theory

In Theorem 3.2, we show that the lower bound of minimum distortion of $\boldsymbol{x_0}$ is associated with two terms, $g(\boldsymbol{x_0})$, where $g(\boldsymbol{x}) = f_c(\boldsymbol{x}) - f_j(\boldsymbol{x})$, and the cross Lipschitz constant L_{q,x_0}, which is defined as $\max_{\boldsymbol{x} \in B_p(\boldsymbol{x}_0, R)} \|\nabla g(\boldsymbol{x})\|_q$. Note that $g(\boldsymbol{x_0})$ is readily available at the output of a classifier. Although $\nabla g(\boldsymbol{x})$ can be easily computed via back propagation, the calculation of cross Lipschitz constant is more involved as it requires computing the maximum value of $\|\nabla g(\boldsymbol{x})\|_q$ in a ball.

If we can exhaustively list all the $\boldsymbol{x} \in B_p(\boldsymbol{x}_0, R)$, then we can get the exact value $\max_{\boldsymbol{x} \in B_p(\boldsymbol{x}_0, R)} \|\nabla g(\boldsymbol{x})\|_q$. However, it is impossible to do this exhaustive search because we have large dimension d in image classifiers, where $d = 784$ for MNIST, $d = 3072$ for CIFAR, and $d = 150528$ for ImageNet. Instead of exhaustive search, one intuitive approach is to perform sampling on $\boldsymbol{x}$ and take the maximum value of $\|\nabla g(\boldsymbol{x}^{(i)})\|_q$, where $\boldsymbol{x}^{(i)}$ are the samples we generated. The problem with this approach is that we might need a significant amount of samples to obtain a good estimate of $\max \|\nabla g(\boldsymbol{x})\|_q$ and we don't know how good our estimate is compared to the true value. Fortunately, Extreme Value Theory tells us that the maximum value of random variables can only follow one of the three extreme value distributions, which is useful for us to estimate $\max \|\nabla g(\boldsymbol{x})\|_q$ with only a tractable number of samples.

[1] proof deferred to Appendix B
[2] proof deferred to Appendix C

Below we first give some intuitions on how $\|\nabla g(\boldsymbol{x})\|_q$ can be regarded as a random variable via our sampling approach, and then derive the cumulative density function for a simple one-hidden layer neural network in Theorem 4.1. In Section 4.2, we discuss Extreme Value Theory and how it can be applied to estimate $\max \|\nabla g(\boldsymbol{x})\|_q$.

4.1 Sampling on the Distribution of Gradient Norm

We generate samples $\boldsymbol{x}^{(i)}$ over a fixed ball $B_p(\boldsymbol{x_0}, R)$ uniformly and independently, where $\boldsymbol{x_0}$ is the input image vector of a classifier. This way, $\|\nabla g(\boldsymbol{x})\|_q$ can be regarded as a random variable Y with cumulative density function (CDF) $F_Y(y)$, which depends on the network architecture. We derive the CDF for a one-hidden-layer neural network in Theorem 4.1, whose proof is deferred to Appendix D.

Theorem 4.1 ($F_Y(y)$ of one-hidden-layer neural network)**.** *Consider a neural network $f : \mathbb{R}^d \to \mathbb{R}^K$ with input $\boldsymbol{x_0} \in \mathbb{R}^d$, a hidden layer with U hidden neurons, and rectified linear unit (ReLU) activation function. If we sample uniformly in a ball $B_p(\boldsymbol{x_0}, R)$, then the cumulative distribution function of $\|\nabla g(\boldsymbol{x})\|_q$, denoted as $F_Y(y)$, is piece-wise linear with at most $M = \sum_{i=0}^{d} \binom{U}{i}$ pieces, where $g(\boldsymbol{x}) = f_c(\boldsymbol{x}) - f_j(\boldsymbol{x})$ for some given c and j, and $\frac{1}{p} + \frac{1}{q} = 1, 1 \leq p, q \leq \infty$.*

Figure 2 illustrates Theorem 4.1 with $d = 2, q = 2$ and $U = 3$. The three hyperplanes $\boldsymbol{w}_i\boldsymbol{x} + b_i = 0$ divide the space into seven regions (with different colors). The red dash line encloses the ball $B_2(\boldsymbol{x_0}, R_1)$ and the blue dash line encloses a larger ball $B_2(\boldsymbol{x_0}, R_2)$. If we draw samples uniformly within the balls, the probability of $\|\nabla g(\boldsymbol{x})\|_2 = y$ is proportional to the intersected volumes of the ball and the regions with $\|\nabla g(\boldsymbol{x})\|_2 = y$.

4.2 Extreme Value Theory and the CLEVER scores

Suppose we have n samples $\{\|\nabla g(\boldsymbol{x}^{(i)})\|_q\}$, and denote them as a sequence of independent and identically distributed (iid) random variables $Y_1, Y_2, \cdots, Y_n$ with CDF $F_Y(y)$. The CDF of $M_n = \max\{Y_1, \cdots, Y_n\}$ is $F_Y^n(y)$, which is called the limit distribution of $F_Y(y)$. The Extreme Value Theory in Theorem 4.2 says that $F_Y^n(y)$, if exists, can only be one of the three family of extreme value distributions - the Gumbel class, the Fréchet class and the Reverse Weibull class.

Lemma 4.2 (Fisher-Tippett-Gnedenko (De Haan & Ferreira, 2007))**.** *If there exists a sequence of pairs of real numbers (a_n, b_n) such that $a_n > 0$ and $\lim_{n\to\infty} F_Y^n(a_n y + b_n) = G(y)$, where G is a non-degenerate distribution function, then G belongs to either the Gumbel class (Type I), the Fréchet class (Type II) or the Reverse Weibull class (Type III) with their CDFs as follows:*

$$\text{Gumbel class (Type I):} \quad G(y) = \exp\Big\{-\exp\Big[-\frac{y-a}{b}\Big]\Big\}, \quad y \in \mathbb{R},$$

$$\text{Fréchet class (Type II):} \quad G(y) = \begin{cases} 0, & \text{if } y < a, \\ \exp\{-\big(\frac{y-a}{b}\big)^{-c}\}, & \text{if } y \geq a, \end{cases}$$

$$\text{Reverse Weibull class (Type III):} \quad G(y) = \begin{cases} \exp\{-\big(\frac{a-y}{b}\big)^{c}\}, & \text{if } y < a, \\ 1, & \text{if } y \geq a, \end{cases}$$

where $a \in \mathbb{R}$, $b > 0$ and $c > 0$ are the location, scale and shape parameters, respectively.

Lemma 4.2 implies that the maximum values of the samples follow one of the three families of distributions. We are particularly interested in the Reverse Weibull class, as its CDF has a finite right end-point a, revealing the upper limit of the samples, also known as the *extreme value*. In our case, this is the unknown local cross Lipschitz constant $L_{q,\boldsymbol{x}_0}$ we would like to estimate. Here, we describe how to estimate $L_{q,\boldsymbol{x}_0}$ with Reverse Weibull class. We generate N_s samples of $\boldsymbol{x}^{(i)}$ uniformly from $B_p(\boldsymbol{x_0}, R)$ in each batch with a total of N_b batches, compute $\|\nabla g(\boldsymbol{x}^{(i)})\|_q$, and store the maximum values of each batch into a set S. We then perform maximum likelihood estimation on the parameters of the Reverse Weibull distribution with S, and the estimated location parameter, $\hat{a}$, is used as an estimate of $L_{q,\boldsymbol{x}_0}$. The flow of computing CLEVER score for targeted attacks is summarized in Algorithm 1. CLEVER also applies to un-targeted attacks by computing CLEVER scores over all possible targets and take the minimum of them.

Algorithm 1: Compute **CLEVER** Score for **t**argeted attack

Input: a K-class neural network $f(\boldsymbol{x})$, data example $\boldsymbol{x_0}$ with predicted class c, target class j, batch size N_b, number of samples per batch N_s, perturbation norm p, max perturbation R

Result: CLEVER Score $\mu \in \mathbb{R}_+$ for target class j

$S \leftarrow \{\emptyset\}$, $g(\boldsymbol{x}) \leftarrow f_c(\boldsymbol{x}) - f_j(\boldsymbol{x})$.
for $i \leftarrow 1$ **to** N_b **do**
  for $k \leftarrow 1$ **to** N_s **do**
    randomly select a point $\boldsymbol{x}^{(i,k)} \in B_p(\boldsymbol{x_0}, R)$
    compute $b_{ik} \leftarrow \|\nabla g(\boldsymbol{x}^{(i,k)})\|_p$ via back propagation
  end
  $S \leftarrow S \cup \{\max_k\{b_{ik}\}\}$
end
$\hat{a}$ = location parameter of maximum likelihood estimation of reverse Weibull distribution on S
$\mu \leftarrow \min(\frac{g(\boldsymbol{x_0})}{\hat{a}}, R)$

5 Experimental Results

5.1 Networks and Parameter Setup

We conduct experiments on CIFAR-10 (CIFAR for short), MNIST, and ImageNet data sets. For the former two smaller data sets CIFAR and MNIST, we evaluate CLEVER scores on four relatively small networks: a single hidden layer MLP with softplus activation (with the same parameters as in (Hein & Andriushchenko, 2017)), a 7-layer AlexNet-like CNN (with the same parameters as in (Carlini & Wagner, 2017b)), and the 7-layer CNN with defensive distillation (Papernot et al., 2016) (DD) and bounded ReLU (Zantedeschi et al., 2017) (BReLU) defense techniques employed.

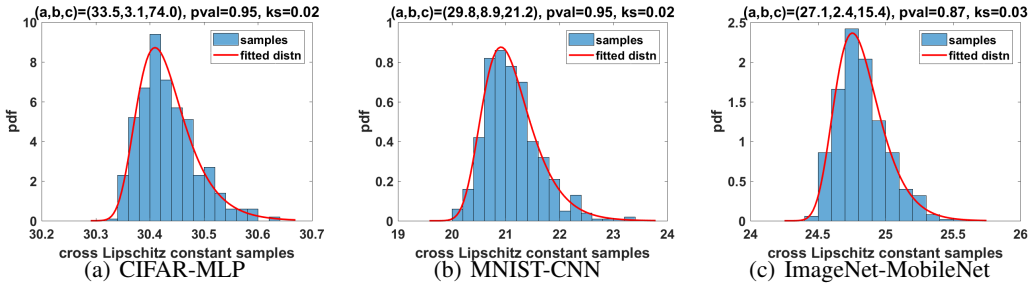
For ImageNet data set, we use three popular deep network architectures: a 50-layer Residual Network (He et al., 2016) (ResNet-50), Inception-v3 (Szegedy et al., 2016) and MobileNet (Howard et al., 2017). They were chosen for the following reasons: (i) they all yield (close to) state-of-the-art performance among equal-sized networks; and (ii) their architectures are significantly different with unique building blocks, i.e., residual block in ResNet, inception module in Inception net, and depthwise separable convolution in MobileNet. Therefore, they should be appropriate architectures to test our robustness metric. For MobileNet, we set the width multiplier to 1.0, which achieves a 70.6% accuracy on ImageNet. For all the three networks, we used the pretrained model from TF-slim library [3].

In all our experiments, we set the sampling parameters $N_b = 500$, $N_s = 1024$ and $R = 5$. We use 500 test-set images for CIFAR and MNIST and use 100 test-set images for ImageNet. For each image, we evaluate its CLEVER score for three targeted attack classes: a random target class, a least likely class (the lowest probability class when predicting the original example), and the top-2 class (which is usually the easiest target to attack). We only conduct experiments on targeted attack since it is strictly harder than un-targeted attack.

5.2 Fitting Gradient Norm samples with Reverse Weibull class

We fit the cross Lipschitz constant samples in S (see Algorithm 1) with Reverse Weibull class distribution to obtain the maximum likelihood estimate of the location parameter $\hat{a}$, scale parameter $\hat{b}$ and shape parameter $\hat{c}$, as introduced in Lemma 4.2. To validate that Reverse Weibull distribution is a good fit to the empirical distribution of the cross Lipschitz constant samples, we conduct Kolmogorov-Smirov Goodness-of-Fit test (a.k.a. K-S test) to calculate the KS statistics and p-values. Tested on CIFAR-MLP, MNIST-CNN, and ImageNet-MobileNet and as displayed on the top of each plot in Figure 3, the resulting high p-values and small KS scores empirically validate the use of Reverse Weibull distribution as an underlying distribution of the cross Lipschitz constant samples. Therefore, its location parameter $\hat{a}$ (i.e., the extreme value) can be used to calculate the CLEVER score. Figure 3 plots the probability distribution function of the cross Lipschitz constant

[3] https://github.com/tensorflow/models/tree/master/research/slim



(a) CIFAR-MLP (b) MNIST-CNN (c) ImageNet-MobileNet

Figure 3: The cross Lipschitz constant samples and the fitted Reverse Weibull distribution with the corresponding MLE estimates of location, scale and shape parameters (a, b, c), which are specified on the top of each plot. Here each plot corresponds to CIFAR-MLP, MNIST-CNN, and ImageNet-MobileNet. The score of Kolmogorov-Smirov Goodness-of-Fit test and p-values are also calculated and denoted by *ks* and *pval*. With small ks and high pval, the results show that the hypothesized Reverse Weibull distribution well fits to the empirical distribution of cross Lipschitz constant samples.

samples and the fitted Reverse Weibull distribution for various data sets and network architectures. The estimated MLE parameters, p-values and the KS scores are also specified accordingly.

5.3 Comparing CLEVER Score with Attack-induced Network Robustness

We apply the state-of-the-art white-box attack methods, iterative fast gradient sign method (I-FGSM) (Goodfellow et al., 2015; Kurakin et al., 2016b) and Carlini and Wagner's attack (CW attack) (Carlini & Wagner, 2017b), to find adversarial examples for 11 networks, including 4 networks trained on CIFAR, 4 networks trained on MNIST, and 3 networks trained on ImageNet. For CW attack, we run 1000 iterations for ImageNet and CIFAR, and 2000 iterations for MNIST, as MNIST has shown to be more difficult to attack (Chen et al., 2017a). Attack learning rate is individually tuned for each model: 0.001 for Inception-v3 and ResNet-50, 0.0005 for MobileNet and 0.01 for all other networks. For I-FGSM, we run 50 iterations and choose the optimal $\epsilon \in \{0.01, 0.025, 0.05, 0.1, 0.3, 0.50.81.0\}$ to achieve the smallest ℓ_∞ distortion for each individual image. For defensively distilled (DD) networks, 50 iterations of I-FGSM are not sufficient; we use 250 iterations for CIFAR-DD and 500 iterations for MNIST-DD to get a 100% success rate. For the problem to be non-trivial, images that are classified incorrectly are skipped. For comparison, we compute the CLEVER scores for the same set of images and attack targets. To the best of our knowledge, CLEVER is the first attack-independent robustness score that is capable of handling the large networks studied in this paper, so we directly compare it with the attack-induced distortion metrics in our study.

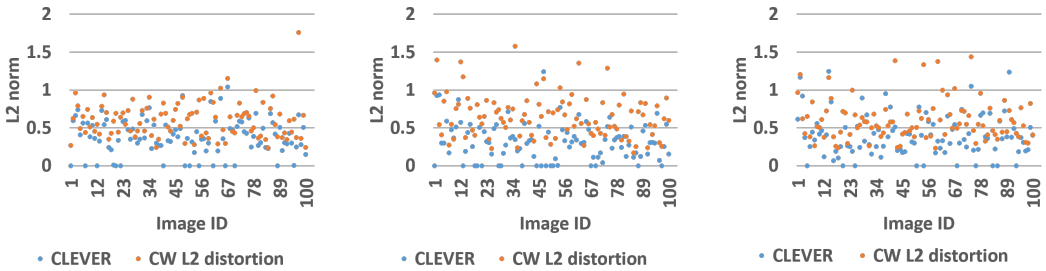
Table 2: Average ℓ_2 and ℓ_∞ distortions found by I-FGSM and CW attacks, and comparison with average CLEVER scores for ℓ_2 and ℓ_∞ norms. DD and BReLU represent Defensive Distillation and Bounded ReLU defending methods applied to the CNN network.

	Least Likely Target						Random Target						Top-2 Target					
	CW		I-FGSM		CLEVER		CW		I-FGSM		CLEVER		CW		I-FGSM		CLEVER	
	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞
MNIST-MLP	2.575	0.475	4.273	0.223	2.293	0.085	1.833	0.337	3.369	0.173	1.314	0.086	1.128	0.218	2.374	0.119	1.276	0.082
MNIST-CNN	2.377	0.601	4.417	0.313	1.153	0.079	2.005	0.550	3.902	0.264	1.084	0.076	1.504	0.451	3.242	0.211	0.861	0.006
MNIST-DD	2.644	0.578	4.957	0.283	1.569	0.101	2.240	0.531	4.253	0.238	1.367	0.089	1.542	0.412	3.010	0.165	1.372	0.089
MNIST-BReLU	2.349	0.601	5.170	0.276	2.096	0.333	1.923	0.544	4.544	0.238	1.576	0.259	1.404	0.442	3.778	0.196	1.207	0.176
CIFAR-MLP	1.123	0.086	1.896	0.039	0.598	0.011	0.673	0.051	1.214	0.024	0.565	0.011	0.262	0.019	0.689	0.013	0.581	0.011
CIFAR-CNN	0.836	0.053	1.067	0.033	0.228	0.005	0.372	0.042	0.837	0.023	0.216	0.005	0.188	0.022	0.552	0.013	0.206	0.004
CIFAR-DD	2.065	0.091	1.540	0.053	0.360	0.011	0.624	0.066	1.097	0.032	0.324	0.010	0.296	0.033	0.582	0.014	0.232	0.007
CIFAR-BReLU	0.407	0.045	0.928	0.030	0.299	0.006	0.303	0.034	0.732	0.022	0.215	0.005	0.152	0.018	0.494	0.012	0.117	0.002
Inception-v3	0.628	0.023	2.244	0.011	0.476	0.002	0.595	0.021	2.261	0.012	0.394	0.002	0.287	0.010	2.073	0.011	0.220	0.001
Resnet-50	0.767	0.031	2.410	0.015	0.389	0.002	0.647	0.025	2.098	0.012	0.315	0.002	0.212	0.010	1.682	0.010	0.126	0.001
MobileNet	0.837	0.025	2.195	0.010	0.576	0.002	0.603	0.018	2.066	0.010	0.382	0.002	0.190	0.006	1.771	0.010	0.136	0.001

We report that the attack success rates for all the networks are 100%, and thus the average distortion of adversarial examples can indicate the attack-specific robustness of each network. Table 2 shows the average ℓ_2 and ℓ_∞ distortions for each attack algorithm and attack target for all data sets, as well as the corresponding average CLEVER scores for ℓ_2 and ℓ_∞ norms. We evaluate the effectiveness of our CLEVER score by comparing the upper bound β_U (found by attacks) and CLEVER score, where CLEVER serves as an estimated lower bound, β_L. As expected, CLEVER is smaller than the

Table 3: Percentage of images in ImageNet where the CLEVER score for that image is greater than the adversarial distortion found by different attacks.

	Least Likely Target				Random Target				Top-2 Target			
	CW		I-FGSM		CW		I-FGSM		CW		I-FGSM	
	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞
MobileNet	7%	0%	0%	0%	4%	0%	0%	0%	0%	0%	0%	0%
Resnet-50	6%	0%	0%	0%	6%	0%	0%	0%	6%	0%	0%	0%
Inception-v3	22%	0%	0%	0%	17%	0%	0%	0%	14%	0%	0%	0%



(a) MobileNet (b) ResNet-50 (c) Inception-v3

Figure 4: Comparison of the CLEVER score and the ℓ_2 norm of adversarial distortion generated by CW attack with random targets for 100 images. The x-axis is image ID and the y-axis is the ℓ_2 distortion metric.

distortions of adversarial images in most cases. More importantly, since CLEVER is independent of attack algorithms, the reported CLEVER scores can roughly indicate the distortion of the *best possible* attack in terms of a specific ℓ_p distortion. The average ℓ_2 distortion found by CW attack is close to the ℓ_2 CLEVER score, indicating it is a strong ℓ_2 attack. In addition, when a defense mechanism (Defensive Distillation or Bounded ReLU) is used, the corresponding CLEVER scores are consistently increased, indicating that the network is indeed made more resilient to adversarial perturbations. Our CLEVER score can also be used as a security checkpoint for unseen attacks. For example, if there is an substantial gap in distortion between the CLEVER score and the considered attack algorithms, it may suggest the existence of a more effective attack that can close the gap.

Since CLEVER score is derived from an estimation of the robustness lower bound, we further verify the viability of CLEVER per each example, i.e., whether it is usually smaller than the upper bound found by attacks. Table 3 shows the percentage of inaccurate estimations where the CLEVER score is larger than the distortion of adversarial examples found by CW and I-FGSM attacks in three ImageNet networks. We found that CLEVER score provides an accurate estimation for most of the examples. For MobileNet and Resnet-50, our CLEVER score is a strict lower bound of these two attacks for more than 93% of tested examples. For Inception-v3, the condition of strict lower bound is worse (still more than 78%), but we found that in these cases the attack distortion only differs from our CLEVER score by a fairly small amount. For the purpose of visual illustration, Figure 4 shows the scatter plots of our CLEVER scores and the ℓ_2 distortions of CW attack for all tested examples on ImageNet. It can be observed that most of the adversarial examples are close to the corresponding CLEVER scores, which signifies the near-optimality of CW attack in terms of ℓ_2 distortion, as CLEVER suffices for an estimated capacity of the best possible attack.

6 Conclusion

In this paper, we propose the CLEVER score, a novel and generic metric to evaluate the robustness of a target neural network classifier to adversarial examples. Compared to the existing robustness evaluation approaches, our metric has the following advantages: (i) attack-agnostic; (ii) applicable to any neural network classifiers; (iii) comes with strong theoretical guarantees; and (iv) is computationally feasible for large neural networks. Our extensive experiments show that the CLEVER score well matches the practical robustness indication of a wide range of natural and defended networks.

REFERENCES

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263*, 2017a.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017b.

Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: Elastic-net attacks to deep neural networks via adversarial examples. *arXiv preprint arXiv:1709.04114*, 2017a.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. *arXiv preprint arXiv:1708.03999*, 2017b.

Laurens De Haan and Ana Ferreira. *Extreme value theory: an introduction*. Springer Science & Business Media, 2007.

Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. *arXiv preprint arXiv:1705.01320*, 2017.

Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2017.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR'15; arXiv preprint arXiv:1412.6572*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *arXiv preprint arXiv:1705.08475*, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pp. 3–29. Springer, 2017.

Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. *arXiv preprint arXiv:1702.01135*, 2017a.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Towards proving the adversarial robustness of deep neural networks. *arXiv preprint arXiv:1709.02802*, 2017b.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016a.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *ICLR'17; arXiv preprint arXiv:1611.01236*, 2016b.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, 2016.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM Asia Conference on Computer and Communications Security*, pp. 506–519, 2017.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Remigijus Paulavičius and Julius Žilinskas. Analysis of different norms and corresponding lipschitz constants for global optimization. *Technological and Economic Development of Economy*, 12(4): 301–306, 2006.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Beilun Wang, Ji Gao, and Yanjun Qi. A theoretical framework for robustness of (deep) classifiers under adversarial noise. *arXiv preprint arXiv:1612.00334*, 2016.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. *arXiv preprint arXiv:1707.06728*, 2017.

APPENDIX A: PROOF OF THEOREM 3.2

Proof. According to Lemma 3.1, the assumption that g is Lipschitz continuous with Lipschitz constant L_q gives

$$|g(\boldsymbol{x}) - g(\boldsymbol{y})| \leq L_q \|\boldsymbol{x} - \boldsymbol{y}\|_p. \tag{4}$$

Let $\boldsymbol{x} = \boldsymbol{x_0} + \boldsymbol{\delta}$ and $\boldsymbol{y} = \boldsymbol{x_0}$ in (4), we get

$$|g(\boldsymbol{x_0} + \boldsymbol{\delta}) - g(\boldsymbol{x_0})| \leq L_q \|\boldsymbol{\delta}\|_p,$$

which can be rearranged into the following form

$$g(\boldsymbol{x_0}) - L_q \|\boldsymbol{\delta}\|_p \leq g(\boldsymbol{x_0} + \boldsymbol{\delta}) \leq g(\boldsymbol{x_0}) + L_q \|\boldsymbol{\delta}\|_p. \tag{5}$$

In other words, the assumption of Lipschitz continuity on g gives bounds on $g(\boldsymbol{x_0} + \boldsymbol{\delta})$ in terms of $g(\boldsymbol{x_0})$, $\|\boldsymbol{\delta}\|_p$ and L_q, which will be useful in deriving the bounds on $\|\boldsymbol{\delta}\|_p$ in below.

First, if $g(\boldsymbol{x_0}) - L_q \|\boldsymbol{\delta}\|_p > 0$, then the bounds in (5) give

$$g(\boldsymbol{x_0} + \boldsymbol{\delta}) \geq g(\boldsymbol{x_0}) - L_q \|\boldsymbol{\delta}\|_p > 0.$$

By the definition of g, the above equation means that if $\|\boldsymbol{\delta}\|_p$ is less than $\frac{f_c(\boldsymbol{x_0}) - f_j(\boldsymbol{x_0})}{L_q}$, then the classifier decision on $\boldsymbol{x_0} + \boldsymbol{\delta}$ will never change from class c to class j:

$$\|\boldsymbol{\delta}\|_p < \frac{f_c(\boldsymbol{x_0}) - f_j(\boldsymbol{x_0})}{L_q} \quad \Rightarrow \quad f_c(\boldsymbol{x_0} + \boldsymbol{\delta}) - f_j(\boldsymbol{x_0} + \boldsymbol{\delta}) > 0. \tag{6}$$

On the other hand, if there exists a $\boldsymbol{\delta}$ such that $g(\boldsymbol{x_0} + \boldsymbol{\delta}) < 0$, then the bounds in (5) give

$$0 > g(\boldsymbol{x_0} + \boldsymbol{\delta}) \geq g(\boldsymbol{x_0}) - L_q \|\boldsymbol{\delta}\|_p.$$

Again, by the definition of g, the above equation means that if the classifier decision *can* be changed to class j, we need $\|\boldsymbol{\delta}\|_p$ to be at least larger than a positive constant $\frac{f_c(\boldsymbol{x_0}) - f_j(\boldsymbol{x_0})}{L_q}$:

$$f_c(\boldsymbol{x_0} + \boldsymbol{\delta}) - f_j(\boldsymbol{x_0} + \boldsymbol{\delta}) < 0 \quad \Rightarrow \quad \|\boldsymbol{\delta}\|_p > \frac{f_c(\boldsymbol{x_0}) - f_j(\boldsymbol{x_0})}{L_q} > 0. \tag{7}$$

We should be careful about the above statement - it is possible that there doesn't exist a $\boldsymbol{\delta}$ at all achieving $f_c(\boldsymbol{x_0} + \boldsymbol{\delta}) - f_j(\boldsymbol{x_0} + \boldsymbol{\delta})$, for example, if the functions satisfy $f_c(\boldsymbol{x}) > f_i(\boldsymbol{x}), \forall \boldsymbol{x}$. However, (7) says that if the classifier decision of $\boldsymbol{x_0} + \boldsymbol{\delta}$ can be changed, then the ℓ_p norm of this $\boldsymbol{\delta}$ has to be at least larger than $\frac{f_c(\boldsymbol{x_0}) - f_j(\boldsymbol{x_0})}{L_q}$.

Finally, to achieve $c = \operatorname{argmax}_{1 \leq i \leq K} f_i(\boldsymbol{x_0} + \boldsymbol{\delta})$, we take the minimum of the bound on $\|\boldsymbol{\delta}\|_p$ in (6) over $j \neq c$. I.e. if

$$\|\boldsymbol{\delta}\|_p \leq \min_{j \neq c} \frac{f_c(\boldsymbol{x_0}) - f_j(\boldsymbol{x_0})}{L_q},$$

the classifier decision can *never* be changed and the attack will *never* succeed. □

APPENDIX B: PROOF OF COROLLARY 3.2.1

Proof. By Lemma 3.1 and let $g = f_c - f_j$, we get $L_{q,x_0} = \max_{y \in B_p(x_0, R)} \|\nabla g(y)\|_q = \max_{y \in B_p(x_0, R)} \|\nabla f_j(y) - \nabla f_c(y)\|_q$, which then gives the bound in Theorem 2.1 of (Hein & Andriushchenko, 2017). □

APPENDIX C: PROOF OF LEMMA 3.3

Proof. For any x, y, let $\{(c_i x + (1 - c_i) y\}_{i=1}^{k-1}$ be the non-differentiable points between in line(x, y) and $0 = c_0 < c_1 < \cdots < c_{k-1} < c_k = 1$. Let $z_i = (c_i x + (1 - c_i) y)$ for all $i = 0, \ldots, k$. Then,

using the fundamental theorem of calculus on each interval separately, there exists $\bar{z}_i \in (z_i, z_{i-1})$ for each i such that

$$\begin{aligned} f(y) - f(x) &\leq \sum_{i=1}^{k} |f(z_i) - f(z_{i-1})| \\ &\leq \sum_{i=1}^{k} |\nabla f(\bar{z}_i)^T (z_i - z_{i-1})| \\ &\leq \sum_{i=1}^{k} L_q \|z_i - z_{i-1}\|_p \\ &= L_q \|x - y\|_p. \end{aligned}$$

□

APPENDIX D: PROOF OF THEOREM 4.1

Proof. The j_{th} output of a one-hidden-layer neural network can be written as

$$f_j(\boldsymbol{x}) = \sum_{r=1}^{U} \boldsymbol{V}_{jr} \cdot \sigma \left(\sum_{i=1}^{d} \boldsymbol{W}_{ri} \cdot x_i + b_r \right) = \sum_{r=1}^{U} \boldsymbol{V}_{jr} \cdot \sigma \left(\boldsymbol{w}_r \boldsymbol{x} + b_r \right),$$

where $\sigma(z) = \max(z, 0)$ is ReLU activation function, $\boldsymbol{W}$ and $\boldsymbol{V}$ are the weight matrices of the first and second layer respectively, and $\boldsymbol{w}_r$ is the r_{th} row of $\boldsymbol{W}$. Thus, we can compute $g(\boldsymbol{x})$ and $\|\nabla g(\boldsymbol{x})\|_q$ below:

$$\begin{aligned} g(\boldsymbol{x}) = f_c(\boldsymbol{x}) - f_j(\boldsymbol{x}) &= \sum_{r=1}^{U} \boldsymbol{V}_{cr} \cdot \sigma \left(\boldsymbol{w}_r \boldsymbol{x} + b_r \right) - \sum_{r=1}^{U} \boldsymbol{V}_{jr} \cdot \sigma \left(\boldsymbol{w}_r \boldsymbol{x} + b_r \right) \\ &= \sum_{r=1}^{U} (\boldsymbol{V}_{cr} - \boldsymbol{V}_{jr}) \cdot \sigma \left(\boldsymbol{w}_r \boldsymbol{x} + b_r \right) \end{aligned}$$

and

$$\|\nabla g(\boldsymbol{x})\|_q = \left\| \sum_{r=1}^{U} \mathbb{I}(\boldsymbol{w}_r \boldsymbol{x} + b_r)(\boldsymbol{V}_{cr} - \boldsymbol{V}_{jr}) \boldsymbol{w}_r^\top \right\|_q,$$

where $\mathbb{I}(z)$ is an univariate indicator function:

$$\mathbb{I}(z) = \begin{cases} 1, & \text{if } z > 0, \\ 0, & \text{if } z \leq 0. \end{cases}$$

As illustrated in Figure 2, the hyperplanes $\boldsymbol{w}_r \boldsymbol{x} + b_r = 0, r \in \{1, \ldots, U\}$ divide the d dimensional spaces $\mathbb{R}^d$ into different regions, with the interior of each region satisfying different inequality constraints, e.g. $\boldsymbol{w}_{r_+} \boldsymbol{x} + b_{r_+} > 0$ and $\boldsymbol{w}_{r_-} \boldsymbol{x} + b_{r_-} < 0$. Given $\boldsymbol{x}$, we can identify which region it belongs to by checking the sign of $\boldsymbol{w}_r \boldsymbol{x} + b_r$ for each r. Notice that the gradient norm is the same for all the points in the same region, i.e. for any $\boldsymbol{x}_1, \boldsymbol{x}_2$ satisfying $\mathbb{I}(\boldsymbol{w}_r \boldsymbol{x}_1 + b_r) = \mathbb{I}(\boldsymbol{w}_r \boldsymbol{x}_2 + b_r)\ \forall r$, we have $\|\nabla g(\boldsymbol{x}_1)\|_q = \|\nabla g(\boldsymbol{x}_2)\|_q$. Since there can be at most $M = \sum_{i=0}^{d} \binom{U}{i}$ different regions for a d-dimensional space with U hyperplanes, $\|\nabla g(\boldsymbol{x})\|_q$ can take at most M different values.

Therefore, if we perform uniform sampling in a ball $B_p(\boldsymbol{x_0}, R)$ centered at $\boldsymbol{x_0}$ with radius R and denote $\|\nabla g(\boldsymbol{x})\|_q$ as a random variable Y, the probability distribution of Y is discrete and its CDF is piece-wise constant with at most M pieces. Without loss of generality, assume there are $M_0 \leq M$ distinct values for Y and denote them as $m_{(1)}, m_{(2)}, \ldots, m_{(M_0)}$ in an increasing order, the CDF of Y, denoted as $F_Y(y)$, is the following:

$$F_Y(m_{(i)}) = F_Y(m_{(i-1)}) + \frac{\mathbb{V}_d(\{\boldsymbol{x} \mid \|\nabla g(\boldsymbol{x})\|_q = m_{(i)}\}) \cap \mathbb{V}_d(B_p(\boldsymbol{x_0}, R)))}{\mathbb{V}_d(B_p(\boldsymbol{x_0}, R))}, i = 1, \ldots, M_0,$$

where $F_Y(m_{(0)}) = 0$ with $m_{(0)} < m_{(1)}$, $\mathbb{V}_d(E)$ is the volume of E in a d dimensional space. □