Measuring Robustness of Deep Neural Networks from the Lens of Statistical Model Checking

Hao Bu
School of Mathematical Sciences
Peking University
Beijing, China
buhao@pku.edu.cn

Meng Sun✉
School of Mathematical Sciences
Peking University
Beijing, China
sunm@pku.edu.cn

2023 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-8867-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IJCNN54540.2023.10191319

***Abstract*—Measuring robustness of deep neural networks (DNNs) is an important topic for trustworthy AI. Existing methods for verifying local robustness of DNNs usually face the scalability problem and have difficulties to deal with non-linear activation functions and complex semantic perturbations. Existing methods for measuring global robustness usually rely on large datasets, so it is difficult for users without a large dataset to compare the global robustness of different networks. In this paper, we propose two algorithms to measure the local and global robustness of DNNs from the lens of statistical model checking. Compared with the local robustness estimation method using the Okamoto bound, our method can provide the same theoretical guarantee using only 48.6%-74.1% of running time. Our global robustness estimation algorithm can provide high-quality estimation using only 10 images for CIFAR-10 and 50 images for ImageNet, thus can be used in a wider range of scenarios.**

I. Introduction

Since the discovery of adversarial examples, researchers have proposed many techniques to measure the robustness of DNNs. There are two types of robustness for DNNs: *local robustness* and *global robustness*.

For local robustness, researchers mainly focus on L_p robustness. Given a network, an input image and a radius, people want to know if for all images in the L_p ball centered at the input with the radius, the network produces the same result. Later, researchers extend the L_p robustness to the semantic domain (e.g. brightness and rotation) [1], [2], which is more challenging to verify. There are two main drawbacks for existing local robustness verification methods:

(1) Although for some norms (e.g. L_∞ and L_1) and piece-wise linear activation functions (e.g. ReLU), there exist sound and complete verification methods, the verification problem has been proved to be NP-hard [3]. Existing methods usually can only deal with relatively small networks.

(2) For non-linear activation functions (e.g. sigmoid) or complex semantic perturbations (e.g. rotation), it is hard to find a method that is both sound and complete. Researchers usually relax the problem to partially solve it [1], [4].

For global robustness, people consider DNN's robustness on *all* possible images in the environment instead of a *single* image. As it is infeasible to enumerate all possible images, researchers usually use a large dataset to empirically measure the global robustness [5]. For example, for Gaussian noise and CIFAR-10, people generate a dataset containing 50000 images with Gaussian noise, and regard DNN's accuracy on it as its global robustness towards Gaussian noise. However, in many cases, high-quality DNNs are trained by commercial companies, and the relevant datasets are regarded as digital assets of the company. When users obtain pre-trained networks from the provider, they can only get the weights and structure of the network, and cannot obtain the training or testing datasets. Therefore, it is hard for users to compare the global robustness of different networks without owning a large dataset.

In this paper, we try to alleviate these problems from the lens of statistical model checking. Model checking [6] is a formal method to check if a transition system meets a given property. It also faces the scalability problem, and for some complex systems, there is no sound and complete algorithm to verify them. Statistical model checking [7], [8] is proposed to overcome these problems. Different from traditional model checking which simply returns "Yes" or "No", statistical model checking estimates the probability (a real number in $[0, 1]$) of the property being met. As statistical model checking is model agnostic, it can also be used to measure the local robustness of DNNs. Specifically, we try to estimate the probability that perturbed images are correctly classified, while existing techniques simply return "Yes" (all perturbed images can be correctly classified) or "No" (there exists a perturbed image that is misclassified). By introducing some statistical uncertainty, we can deal with large networks, non-linear activation functions and complex semantic perturbations.

Existing statistical model checkers [8]–[11] mainly use the Okamoto bound [12] to decide the sampling number. Although it can be directly used for DNN local robustness estimation, the distribution of DNN local robustness has some special characteristics, which lead to a new method that can decrease the running time by 25.9%-51.4% compared with the Okamoto bound. Moreover, we use a real number in $[0, 1]$ to measure the local robustness, which contains more information than simply returning "Yes" or "No". Using this extra information, we can estimate the global robustness of DNNs using only a few images (10 images for CIFAR-10 and 50 images for

This work was sponsored by the National Natural Science Foundation of China under grant No. 62172019 and CCF-Huawei Populus Grove Fund.

ImageNet), and the estimation results are strongly correlated with the existing method using large datasets (50000 images for CIFAR-10 and 250000 images for ImageNet).

The contributions of this paper are as follows:

(1) We propose an efficient and sound algorithm to measure the local robustness of DNNs. It can deal with complex networks and perturbations.

(2) We propose a global robustness estimation method which can provide high-quality estimation using only a few images. It is useful for users without a large dataset to compare the global robustness of different DNNs.

II. Related Work

A. Estimation Algorithms in Statistical Model Checking

The estimation problem of statistical model checking can be regarded as the parameter estimation problem of Bernoulli variables. The parameter p is the probability that the given property is satisfied.

Problem 1: X is a Bernoulli variable with parameter p: $Pr(X = 1) = p, Pr(X = 0) = 1 - p$. Given $\varepsilon, \delta > 0$, we hope to propose an algorithm such that:

$$Pr(|\hat{p} - p| > \varepsilon) \leq \delta$$

where $\hat{p}$ is the estimation result of the algorithm.

Almost all model checkers [8]–[11] use the Okamoto bound to decide the sampling number.

Theorem 1 (Okamoto bound) [12]: For any $p \in [0, 1], n > 0$, $X_1, X_2, \cdots, X_n$ are independent Bernoulli variables with parameter p. Define $\hat{p_n} \triangleq \frac{\sum_{i=1}^{n} X_i}{n}$, then

$$Pr(|\hat{p_n} - p| > \varepsilon) \leq 2 \cdot \exp(-2n\varepsilon^2)$$

Therefore, we can perform $n = \lceil \frac{1}{2\varepsilon^2} \ln(\frac{2}{\delta}) \rceil$ samplings and use the proportion $\hat{p_n}$ as the estimation of p.

In addition to the Okamoto bound, there are some sequential algorithms to deal with Problem 1. Frey [13] proposes the first sound sequential algorithm to deal with the problem, and Chen [14] proposes a similar method later. These two methods both come from the statistics field. To guarantee the soundness, they need to enumerate all possible conditions crudely before sampling, which introduces huge extra time cost. For example, Frey's method needs to calculate a "critical value" before sampling, but we cannot obtain the critical value for some ε and δ within 168 hours on a server using Frey's program (this is also confirmed in [15]).

UPPAAL-SMC [16] uses a naive sequential algorithm with no proof, and it is thought to be unsound [13], [15]. Jegourel *et al.* [15] propose a sequential algorithm using the Massart bound. However, their implementation is inconsistent with their proof. In their implementation (Algorithm 1 in [15]), they use the same samplings both for computing the confidence interval and for estimation, but in their proof (Theorem 5.1 and Theorem 5.3 in [15]), the confidence interval is separately given as a precondition rather than calculated using the same samplings for estimation. These two methods [15], [16] both come from the software engineering field and their soundness is unclear.

Our local robustness estimation algorithm has better average performance than the Okamoto bound. Moreover, compared with these four sequential algorithms [13]–[16], our algorithm is much easier to analyze and has a clear soundness proof. The extra time cost of our algorithm is also negligible.

B. Measuring Local Robustness of Deep Neural Networks

Given a network, an image and a perturbation, most existing methods [1]–[4] try to answer if the network can correctly classify all possible perturbed images, which is computationally hard. As an alternative, researchers estimate the probability (a real number in $[0, 1]$) that perturbed images are correctly classified rather than simply return "Yes" or "No". As these methods typically treat the network as a black box and use statistical methods for estimation, they can work efficiently on complex networks and perturbations. Webb *et al.* [17] are the first to estimate the local robustness of DNNs using statistical methods, but they give no theoretical guarantee for their method. Levy *et al.* [18] propose a method assuming the "highest incorrect confidence score" follows the normal distribution. However, this assumption usually does not hold, and they do not analyze the extra uncertainty caused by the Box-Cox transformation and the Anderson-Darlin goodness-of-fit test. In contrast, our local robustness estimation algorithm has rigorous theoretical guarantee thus is more reliable.

Cardelli *et al.* [19] estimate the local robustness of Bayesian neural networks (BNNs). They first use the statistical method in [15] to transform a BNN to several normal DNNs, then they use existing DNN verification techniques [4] to verify the local robustness of these DNNs. Therefore, their method can be regarded as the combination of two existing methods. Our local robustness estimation algorithm can be integrated into their framework to provide more solid theoretical guarantee (as the soundness of [15] is unclear) or to improve scalability (as [4] cannot scale to large networks).

Moreover, [20]–[23] use statistical methods to deal with the DNN local robustness certification problem. Specifically, they try to answer if the correctly classified probability p is lower than a given threshold, which is orthogonal to our problem.

C. Measuring Global Robustness of Deep Neural Networks

Global robustness refers to DNN's robustness towards all possible images in the environment. In practice, researchers usually use a large dataset to empirically measure the global robustness of DNNs. CIFAR-10-C (ImageNet-C) [5] is a widely-used dataset containing 15 common semantic perturbations (e.g. contrast, fog and Gaussian noise). As an example, for CIFAR-10 and Gaussian noise, there are 50000 images with Gaussian noise in CIFAR-10-C. Given a network, people define its classification accuracy on these 50000 images as its *empirical robustness* towards Gaussian noise. Similarly, in ImageNet-C, there are 250000 images for each semantic perturbation. These perturbed images are obtained by adding semantic perturbations to clean images from the CIFAR-10 and ImageNet datasets. It is standard practice to use empirical robustness to measure the global robustness of DNNs.

However, empirical robustness relies on large datasets. It is hard for DNN users to calculate the empirical robustness without owning a large dataset. In contrast, our global robustness estimation algorithm only uses a small number of images thus can be used in a wider range of scenarios.

III. Preliminary

A. Semantic Perturbation and L_p Perturbation

We use the following definition [23] which covers the traditional L_p perturbation and common semantic perturbations.

Definition 1 (Perturbation) [23]: A perturbation $\mathcal{T}$ is a function with a parameter vector θ. Given an image $x \in \mathbb{R}^m$, the perturbed result $\mathcal{T}_\theta(x)$ is a random vector with probability distribution $\mathcal{P}_{\mathcal{T}_\theta(x)}$.

Example 1 (L_p perturbation): $\mathcal{P}_{\mathcal{T}_\theta(x)}$ is the uniform distribution in the L_p ball centered at x with a radius of θ.

Example 2 (Gaussian noise): $\mathcal{P}_{\mathcal{T}_\theta(x)}$ is an m-dimensional normal distribution, its mean is x, and its covariance matrix is $\theta^2 I_{m\times m}$.

Example 3 (Rotation): $\mathcal{T}_\theta(x)$ is the result of x rotated by θ degrees.

B. Clopper-Pearson Confidence Interval

Given a confidence level $\delta > 0$, if we perform n Bernoulli trails and find x successes ($X = 1$) and $n - x$ failures ($X = 0$), then the Clopper-Pearson confidence interval [24] for p is $[B^{-1}(x, n-x+1, \frac{\delta}{2})), B^{-1}(x+1, n-x, 1-\frac{\delta}{2})]$, where B^{-1} is the inverse of the incomplete beta function.

Theorem 2 (Soundness of the Clopper-Pearson confidence interval) [24]: For any $p \in [0, 1]$ and sampling number n,

$$Pr(p \in [B^{-1}(x, n-x+1, \frac{\delta}{2})), B^{-1}(x+1, n-x, 1-\frac{\delta}{2})]) \geq 1-\delta$$

IV. Problem Formulation

Problem 2 (Local robustness estimation): Given a neural network $\mathcal{N}$, a correctly classified image x, a perturbation $\mathcal{T}$, a probability distribution of the parameter vector $\mathcal{P}_\theta$ and $\varepsilon, \delta > 0$, we hope to propose an algorithm to estimate the probability $p \triangleq Pr(\mathcal{N}(x) = \mathcal{N}(\mathcal{T}_\theta(x)))$, such that:

$$Pr(|\hat{p} - p| > \varepsilon) \leq \delta$$

where $\hat{p}$ is the estimation result of the algorithm.

Example 4: Let the perturbation $\mathcal{T}$ be rotation (see Example 3), $\mathcal{P}_\theta$ is the uniform distribution in $[-10°, +10°]$. If the network can correctly classify the image when it is rotated by $[-8°, +5°]$, then $p = 0.65$. Suppose $\varepsilon = 0.01, \delta = 0.05$, we hope the estimation result $\hat{p}$ is in $[0.64, 0.66]$ with at least 95% probability.

In global robustness estimation problems, we regard empirical robustness (see Section II-C) as the oracle because it is the standard method to measure global robustness.

Problem 3 (Global robustness estimation): Given a perturbation $\mathcal{T}$ and t neural networks $\mathcal{N}_1, \mathcal{N}_2, \cdots, \mathcal{N}_t$, suppose their empirical robustness values are $r_1, r_2, \cdots, r_t$, we hope to use a small number of images to calculate $\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_t$, such that the Pearson correlation coefficient between $r_1, r_2, \cdots, r_t$ and $\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_t$ is as high as possible.

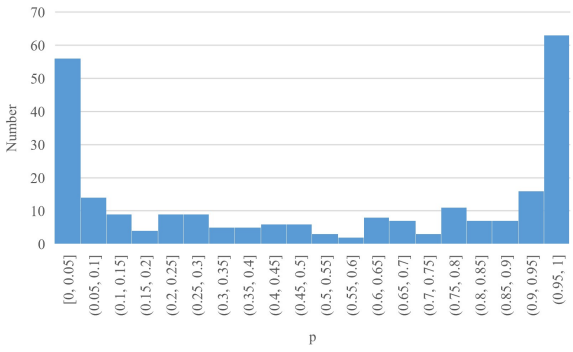

Fig. 1. The distribution of p on 250 different images (dataset: CIFAR-10, network: DenseNet121, perturbation: glass blur).

The Pearson correlation coefficient is widely used to measure the linear correlation between two lists of data. It ranges in $[-1, +1]$, and when it is greater than 0, there is a positive correlation between these two lists of data.

V. Local Robustness Estimation Method

A. Motivation

The Okamoto bound is valid for any p in $[0, 1]$, however, researchers find that for different p, the difficulties of estimation in Problem 1 are different [25]. Specifically, when p is near 0 or 1, the sampling number needed is less than that when p is near 0.5. Moreover, we find that in DNN local robustness estimation problems, the probability $p = Pr(\mathcal{N}(x) = \mathcal{N}(\mathcal{T}_\theta(x)))$ is more likely near 0 or 1. Fig. 1 presents an example distribution of p, where about 50% images have local robustness values near 0 or 1. Therefore, if we can reduce the sampling number for p near 0 and 1, we can reduce the average running time significantly.

B. Overview

For simplicity, from now on, we assume $0 < \varepsilon < \frac{1}{3}$. For $\varepsilon \geq \frac{1}{3}$, we can directly use the Okamoto bound because even for $\delta = 10^{-9}$, we only need less than 100 samplings. As we have mentioned before, for different p, the estimation difficulties are different. This can be captured by following theorems.

Theorem 3: $Pr(|\hat{p}_n - p| > \varepsilon) \leq F(p, \varepsilon, n)$, where

$$F(p, \varepsilon, n) \triangleq \begin{cases} f^n(p, \varepsilon) & p \leq \varepsilon \\ 2 \cdot \exp(-2n\varepsilon^2) & \frac{1-\varepsilon}{2} \leq p \leq \frac{1+\varepsilon}{2} \\ f^n(1-p, \varepsilon) & p \geq 1-\varepsilon \\ f^n(p, \varepsilon) + f^n(1-p, \varepsilon) & otherwise \end{cases}$$

$$f(p, \varepsilon) \triangleq (\frac{p}{p+\varepsilon})^{p+\varepsilon} \cdot (\frac{1-p}{1-p-\varepsilon})^{1-p-\varepsilon} \qquad (\varepsilon < 1-p)$$

Theorem 4: For any ε and n, when $0 \leq p_1 < p_2 \leq \frac{1}{2}$, $F(p_1, \varepsilon, n) \leq F(p_2, \varepsilon, n)$; when $\frac{1}{2} \leq p_1 < p_2 \leq 1$, $F(p_1, \varepsilon, n) \geq F(p_2, \varepsilon, n)$. For any ε and p, when $n_1 > n_2$, $F(p, \varepsilon, n_1) < F(p, \varepsilon, n_2)$.

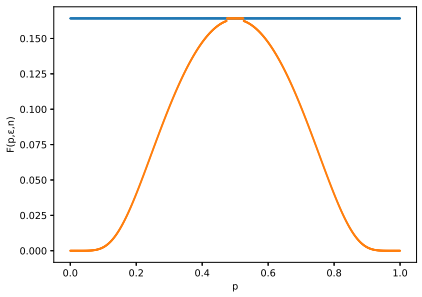

Fig. 2. Okamoto bound (blue) and Theorem 3 (orange) when $n = 500, \varepsilon = 0.05$.

Due to limited space, the proofs of Theorem 3 and Theorem 4 are omitted here and presented in [26].

Fig. 2 presents the Okamoto bound and Theorem 3 when $n = 500, \varepsilon = 0.05$ for estimating $Pr(|\hat{p}_n - p| > \varepsilon)$. When p is near 0.5, these two bounds are similar. However, when p is near 0 or 1, Theorem 3 provides a much better bound than the Okamoto bound. For example, suppose $p = 0.2, \varepsilon = 0.05, \delta = 0.05$, according to Theorem 3, using 500 samplings is enough to ensure $Pr(|\hat{p}_n - p| > \varepsilon) \leq \delta$ (in Fig. 2, when $p = 0.2$, the orange line is lower than 0.05). However, if we use the Okamoto bound, we need to perform $n = \lceil \frac{1}{2\varepsilon^2} \ln(\frac{2}{\delta}) \rceil = 738$ samplings. Therefore, if we know p in advance, we can reduce the sampling number using Theorem 3. However, the problem itself is estimating p, thus it is impossible to know the exact value of p in advance.

Inspired by existing methods [15], we can perform some samplings in advance to approximate p. Specifically, we calculate the Clopper-Pearson confidence interval of p, and use the worst case to calculate the sampling number needed. As the bound in Theorem 3 has good monotonicity with regard to p (see Theorem 4), the worst case happens either at the endpoints of the confidence interval or at $p = 0.5$.

Example 5: Suppose $\varepsilon = 0.01, \delta = 0.05$, according to the Okamoto bound, we need $n = \lceil \frac{1}{2\varepsilon^2} \ln(\frac{2}{\delta}) \rceil = 18445$ samplings. As an alternative, we can perform 100 samplings in advance to calculate the Clopper-Pearson confidence interval of p, then use Theorem 3 to reduce the sampling number. If $p = 0.1$, we perform 100 samplings and find 8 successes, then the confidence interval (99% confidence, $\delta' = 1\%$) of p is $[0.026, 0.176]$. According to Theorem 4, the worst case for $p \in [0.026, 0.176]$ is $p = 0.176$ (the right endpoint). Using Theorem 3, only 11347 samplings are needed to ensure $Pr(|\hat{p}_n - p| > \varepsilon) \leq \delta - \delta' = 4\%$. Therefore, we use 100+11347=11447 samplings in total, which is 37.9% less than the Okamoto bound. However, if $p = 0.5$, we perform 100 samplings and find 53 successes, then the confidence interval is $[0.398, 0.659]$, and the worst case for $p \in [0.398, 0.659]$ is $p = 0.5$. Using Theorem 3, another 19559 samplings are needed. Therefore, we use 100+19559=19659 samplings in total, which is 6.6% more than the Okamoto bound.

It is tricky to decide the sampling number in the first stage for computing the confidence interval of p. More samplings lead to a tighter interval, thus may reduce the sampling number in the next stage, especially when p is near 0 or 1. However, the loss outweighs the gain if too many samplings are performed. Moreover, when p is near 0.5 (see Example 6), it is meaningless to compute the confidence interval as the bound in Theorem 3 is similar to the Okamoto bound. Therefore, we add another stage at the beginning of the algorithm, using a small number of samplings to obtain a rough estimation of p. Then we use this rough estimation to guide further samplings.

C. Algorithm

Our local robustness estimation algorithm for Problem 2 is shown in Algorithm 1. It contains three stages. In the first stage (lines 4-6), we perform about 1% of the Okamoto bound (referred as M) samplings to obtain a rough estimation p_1 for p. In the second stage, we use p_1 to decide how many samplings to perform in this stage. We consider 20 candidates (from $\lceil 0.01 \cdot M \rceil$ to $\lceil 0.20 \cdot M \rceil$), and for each of them, we estimate its total cost (lines 10-14). Specifically, for candidate N, we assume we will get $round(N \cdot p_1)$ successes from N samplings. Then we use N and $round(N \cdot p_1)$ to calculate the confidence interval and estimate the sampling number needed in the third stage using Theorem 3. The total cost of a candidate N equals to N plus the estimated sampling number in the third stage.

(1) If all candidates need more samplings than the Okamoto bound M, we directly perform M samplings, return the estimation and do not enter the third stage (lines 15-18).

(2) If one of the candidates has lower cost than the Okamoto bound, we choose the candidate with the least cost as the sampling number $size_2$ in the second stage. Then we perform $size_2$ samplings and calculate the confidence interval of p (lines 20-22). In the third stage, we decide the sampling number $size_3$ using Theorem 3, perform $size_3$ samplings and return the estimation (lines 24-26).

Algorithm 2 provides the detail for calculating the sampling number in the third stage using Theorem 3. It is straightforward because the bound $F(p, \varepsilon, n)$ in Theorem 3 has good monotonicity over both p and n (see Theorem 4). In most cases, the sampling number has a closed form (lines 1-6, Algorithm 2), only in one case, the sampling number needs to be calculated by binary search (lines 7-12, Algorithm 2). The initial lower bound for binary search is 0, and the initial upper bound is the Okamoto bound. The sampling method is shown in Algorithm 3. We generate the perturbed images $\mathcal{T}_\theta(x)$ according to the given $\mathcal{P}_\theta$, then we count the number of images that have the same label as x.

Example 6: Suppose $\varepsilon = \delta = 0.01$, the Okamoto bound $M = \lceil \frac{1}{2\varepsilon^2} \ln(\frac{2}{\delta}) \rceil = 26492$. In the first stage, we perform 100 samplings (line 4, Algorithm 1).

(1) Suppose $p = Pr(\mathcal{N}(x) = \mathcal{N}(\mathcal{T}_\theta(x))) = 0.1$. In the first stage, we find 14 successes, thus $p_1 = 0.14$. Then we estimate the costs of 20 candidates (from $\lceil 0.01 \cdot M \rceil = 265$ to $\lceil 0.20 \cdot M \rceil = 5299$). As an example, for $\lceil 0.01 \cdot M \rceil = 265$, we assume

Algorithm 1: DNN Local Robustness Estimation

Input: The neural network $\mathcal{N}$, input x, perturbation function $\mathcal{T}$, probability distribution of the parameter vector $\mathcal{P}_\theta$, user-defined bounds ε, δ

Output: Estimated probability

1 $M = \lceil \frac{1}{2\varepsilon^2} \ln(\frac{2}{\delta}) \rceil$ `# Okamoto bound`
2 $\delta' = 0.05 \cdot \delta$
3 `# First stage`
4 $size_1 = max(min(\lceil 0.01 \cdot M \rceil, 100), 10)$
5 $correct_1 = DNN\_sample(\mathcal{N}, x, \mathcal{T}, \mathcal{P}_\theta, size_1)$
6 $p_1 = \frac{correct_1}{size_1}$ `# rough estimation for` p
7 `# Second stage`
8 $size_2\_list = [\lceil 0.01 \cdot M \rceil, \lceil 0.02 \cdot M \rceil, \cdots, \lceil 0.20 \cdot M \rceil]$
9 $total\_cost = []$
10 **for** i *in* $range(20)$ **do**
11 $N = size_2\_list[i]$
12 $N_{correct} = round(N \cdot p_1)$
13 $lb, ub = Clopper\_Pearson(N, N_{correct}, \delta')$
14 $total\_cost.append(N + bound\_estimate(lb, ub, \frac{\delta - \delta'}{1 - \delta'}, \varepsilon))$
15 **if** $min(total\_cost) \geq M$ **then**
16 $size_2 = M$ `# directly using the Okamoto bound`
17 $correct_2 = DNN\_sample(\mathcal{N}, x, \mathcal{T}, \mathcal{P}_\theta, size_2)$
18 **return** $\frac{correct_2}{size_2}$
19 **else**
20 $size_2 = size_2\_list[min\_index(total\_cost)]$
21 $correct_2 = DNN\_sample(\mathcal{N}, x, \mathcal{T}, \mathcal{P}_\theta, size_2)$
22 $lb, ub = Clopper\_Pearson(size_2, correct_2, \delta')$
23 `# Third stage`
24 $size_3 = bound\_estimate(lb, ub, \frac{\delta - \delta'}{1 - \delta'}, \varepsilon)$
25 $correct_3 = DNN\_sample(\mathcal{N}, x, \mathcal{T}, \mathcal{P}_\theta, size_3)$
26 **return** $\frac{correct_3}{size_3}$

Algorithm 2: Function `bound_estimate` Using Theorem 3

Input: The lower bound lb, upper bound ub, required confidence δ, user-defined bound ε

Output: The smallest n such that for any $p \in [lb, ub]$, $F(p, \varepsilon, n) \leq \delta$

if $ub \leq \varepsilon$ **then**
  return $\lceil \log_{f(ub,\varepsilon)} \delta \rceil$
else if $lb \geq 1 - \varepsilon$ **then**
  return $\lceil \log_{f(1-lb,\varepsilon)} \delta \rceil$
else if $(\frac{1-\varepsilon}{2} \leq ub \leq \frac{1+\varepsilon}{2}) \vee (\frac{1-\varepsilon}{2} \leq lb \leq \frac{1+\varepsilon}{2}) \vee (ub \geq \frac{1+\varepsilon}{2} \wedge lb \leq \frac{1-\varepsilon}{2})$ **then**
  return $\lceil \frac{1}{2\varepsilon^2} \ln(\frac{2}{\delta}) \rceil$
else
  if $ub < \frac{1-\varepsilon}{2}$ **then**
    $b = ub$
  else
    $b = lb$
  return the smallest n such that $f^n(b, \varepsilon) + f^n(1 - b, \varepsilon) \leq \delta$ `# using binary search`

Algorithm 3: Function `DNN_sample`

Input: The neural network $\mathcal{N}$, input x, perturbation function $\mathcal{T}$, probability distribution of the parameter vector $\mathcal{P}_\theta$, sampling size $size$

Output: The number of perturbed images that are classified as $\mathcal{N}(x)$

$correct = 0$
$original\_label = \mathcal{N}(x)$
for i *in* $range(size)$ **do**
  Sample θ from $\mathcal{P}_\theta$
  if $\mathcal{N}(\mathcal{T}_\theta(x)) = original\_label$ **then**
    $correct += 1$
return $correct$

we will find $round(265 \cdot p_1) = 37$ successes in 265 samplings (line 12, Algorithm 1). Then we calculate the confidence interval using $(265, 37, 0.05 \cdot \delta)$, and the result is $[0.076, 0.227]$ (line 13, Algorithm 1). Using the confidence interval, we calculate the sampling number in the third stage, and the result is 18759. Therefore, the total cost for $\lceil 0.01 \cdot M \rceil = 265$ is $265 + 18759 = 19024$ (line 14, Algorithm 1). After calculating the costs of 20 candidates, we find $\lceil 0.04 \cdot M \rceil = 1060$ has the least cost ($1060 + 15792 = 16852$), and the cost is lower than the Okamoto bound (26492), so we perform 1060 samplings and find 114 successes (lines 20-21, Algorithm 1). The confidence interval is $[0.077, 0.144]$ (line 22, Algorithm 1). In the third stage, we use the confidence interval to decide the sampling number, perform 14372 samplings and find 1464 successes. Therefore, the algorithm returns $\frac{1464}{14372} \approx 0.1019$ as the estimation of p (lines 24-26, Algorithm 1). The total sampling number is $100 + 1060 + 14372 = 15532$, which is 41.4% lower than the Okamoto bound.

(2) Suppose $p = 0.4$. In the first stage, we find 41 successes, thus $p_1 = 0.41$. Then we estimate the costs of 20 candidates and find all of them need more samplings than the Okamoto bound. Therefore, we directly perform 26492 samplings and find 10550 successes. The algorithm returns $\frac{10550}{26492} \approx 0.3982$ as the estimation of p (lines 16-18, Algorithm 1). The total sampling number is $100 + 26492 = 26592$, which is 0.4% higher than the Okamoto bound.

Note that we do *not* perform samplings when we compare the 20 candidates. Instead, for each candidate N, we *assume* we will obtain $round(N \cdot p_1)$ successes after N samplings, and we estimate the sampling number in the third stage using the assumed sampling result ($round(N \cdot p_1)$ successes in N samplings). Therefore, the time cost of this part (lines 8-15, Algorithm 1) is negligible. Although p_1 is calculated using only a few samplings, we find it is usually enough to select a relatively good candidate for the second stage. We also tried to consider more candidates, but the performance of the algorithm does not change much, indicating that using 20 candidates is fine enough.

Another hyperparameter of the algorithm is δ'. We require the overall error rate of the algorithm is below δ, and it is divided into two parts, one part δ' for calculating the confidence interval, and the other part $\delta - \delta'$ (can be further improved to $\frac{\delta - \delta'}{1 - \delta'}$) for the final estimation. There is a tradeoff in the decision of δ'. In practice, we find that the algorithm performs well when $0.01 \cdot \delta \leq \delta' \leq 0.10 \cdot \delta$, and we choose

$\delta' = 0.05 \cdot \delta$ in our algorithm.

D. Soundness of the Algorithm

Theorem 5 (Soundness of Algorithm 2): For any ε, δ and $0 \leq lb < ub \leq 1$, Algorithm 2 returns the smallest n such that for any $p \in [lb, ub]$, $F(p, \varepsilon, n) \leq \delta$.

The proof of Theorem 5 can be found in [26].

Theorem 6 (Soundness of Algorithm 1): Given the neural network $\mathcal{N}$, input x, perturbation function $\mathcal{T}$, probability distribution of the parameter vector $\mathcal{P}_\theta$, user-defined bounds ε, δ, Algorithm 1 returns an estimation $\hat{p}$ for the probability $p = Pr(\mathcal{N}(x) = \mathcal{N}(\mathcal{T}_\theta(x)))$ satisfying

$$Pr(|\hat{p} - p| > \varepsilon) \leq \delta$$

Proof: The algorithm contains two estimation methods. The first is directly using the Okamoto bound (lines 16-18, Algorithm 1). The second is performing several samplings to compute a confidence interval (lines 20-22, Algorithm 1), then using the confidence interval to decide the sampling number for the final estimation (lines 24-26, Algorithm 1). The soundness of the first method is directly guaranteed by Theorem 1. For the second method, as the Clopper-Pearson confidence interval is sound (Theorem 2), we have $Pr(p \in [lb, ub]) \geq 1 - \delta'$. Moreover, $Pr(|p - \hat{p}| > \varepsilon | p \in [lb, ub]) \leq \frac{\delta - \delta'}{1 - \delta'}$ (using Theorem 3 and Theorem 5). Note that

$$\begin{aligned}
&Pr(|p - \hat{p}| > \varepsilon) \\
=&Pr(|p - \hat{p}| > \varepsilon, p \in [lb, ub]) + Pr(|p - \hat{p}| > \varepsilon, p \notin [lb, ub]) \\
=&Pr(|p - \hat{p}| > \varepsilon | p \in [lb, ub]) \cdot Pr(p \in [lb, ub]) \\
&+ Pr(|p - \hat{p}| > \varepsilon | p \notin [lb, ub]) \cdot Pr(p \notin [lb, ub]) \\
\leq&\frac{\delta - \delta'}{1 - \delta'} \cdot Pr(p \in [lb, ub]) + 1 \cdot (1 - Pr(p \in [lb, ub])) \leq \delta
\end{aligned}$$

Therefore, for any $size_2 > 0, 0 < \delta' < \delta < 1$ and ε, the second method (lines 21-26, Algorithm 1) is sound, i.e. $Pr(|\hat{p} - p| > \varepsilon) \leq \delta$ ($\hat{p} = \frac{correct_3}{size_3}$).

Note that most part of the algorithm (lines 1-15, Algorithm 1) is only used to decide which method to use (along with the corresponding parameters like $size_2$). As these two methods are both sound, the algorithm itself is also sound. □

From the above proof, we know that the two estimation methods are both sound. Therefore, we can even select one of them arbitrarily without harming the soundness of the algorithm. However, if we make the decision using carefully-designed rules (like lines 1-15 in Algorithm 1), we can significantly improve the performance of the algorithm.

VI. Global Robustness Estimation Method

The standard method to measure the global robustness of a network is to evaluate it on a large dataset (see Section II-C). For each image in the dataset, there are only two possible results: "Yes" (correctly classified) or "No" (incorrectly classified). This method performs well when the dataset contains a large number of images. However, sometimes users hope to compare the global robustness of different DNNs using only a few images as they do not have a large dataset. In this case, we need to make full use of every image we have. A natural idea is to use the aforementioned local robustness estimation method, which returns a real number in $[0, 1]$ for each image thus contains more information than a simple "Yes" or "No". Our global robustness estimation method is shown in Algorithm 4.

Algorithm 4: DNN Global Robustness Estimation

Input: t neural networks $\mathcal{N}_1, \mathcal{N}_2, \cdots, \mathcal{N}_t$, perturbation function $\mathcal{T}$, n inputs $x_1, x_2, \cdots, x_n$, user-defined parameters $\varepsilon, \delta, \mathcal{P}_\theta$

Output: Global robustness of the t neural networks

```
result = []
for i from 1 to t do
    sum = 0
    for j from 1 to n do
        sum += local_robustness(N_i, x_j, T, P_θ, ε, δ)
    result.append(sum / n)
return result
```

For each network, we compute its local robustness value on each image (using Algorithm 1) and use the average value as its global robustness value. There are three parameters that need to be provided by the user. Lower ε and δ lead to a more precise local robustness estimation, but also lead to longer running time. We find that setting $\varepsilon = \delta = 0.05$ can already produce a good estimation. The parameter $\mathcal{P}_\theta$ controls the "level" of the perturbation. If the perturbation is too strong (or weak), the robustness values of all networks will concentrate near 0 (or 1), making it difficult to distinguish between different networks. Therefore, in practice, we set a medium $\mathcal{P}_\theta$ to make the results more distinguishable.

VII. Evaluation

We perform three experiments to evaluate our method. All experiments are conducted on a machine with a Xeon Gold 5118 2.30 GHz CPU and an NVIDIA Titan Xp GPU. The Python source code and configurations of experiments can be found in [26].

A. Simulation

In this subsection, we compare the sampling numbers of our method and the Okamoto bound under different ε, δ, p. As we only care about the sampling numbers here, we replace the `DNN_sample` function by a binomial distribution generator with the given p and sampling size. For each pair of (ε, δ), we consider 501 different p $(0.000, 0.002, 0.004, \cdots, 0.998, 1.000)$ and repeat the simulation 100 times. The results are shown in Table I. The three columns refer to the average (maximum / minimum) ratio of our sampling numbers under different p to the Okamoto bound. For example, when $\varepsilon = \delta = 0.01$, the sampling number of our method is 77.4% of the Okamoto bound in average (with regard to 501 different p), 101.1% of the Okamoto bound at most and 8.0% of the Okamoto bound at least. The lower ε and δ are, the more advantage we have over the Okamoto bound. When $\varepsilon = \delta = 0.001$, our method can reduce 30.7% samplings in average, and 98.8% samplings at most. Even in

TABLE I
SAMPLING NUMBERS OF OUR METHOD AND THE OKAMOTO BOUND.

ε	δ	Okamoto	Our method		
			Average	Max	Min
0.05	0.05	738	0.946	1.048	0.392
0.03	0.03	2334	0.877	1.029	0.238
0.01	0.01	24692	0.774	1.011	0.080
0.003	0.001	422273	0.716	1.004	0.023
0.002	0.001	950113	0.706	1.005	0.016
0.001	0.001	3800452	0.693	1.004	0.012

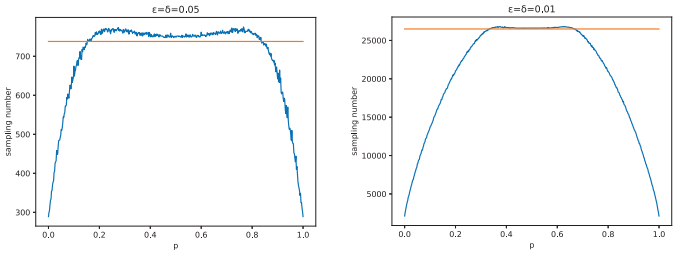

Fig. 3. Sampling numbers of our method (blue) and the Okamoto bound (orange) under different p.

the worst case, our method only uses 0.4% more samplings than the Okamoto bound.

Fig. 3 shows the detailed performance of our method in two representative cases. Our method can dramatically reduce the sampling number when p is near 0 or 1. When p is near 0.5, our method directly uses the Okamoto bound. However, as our method performs extra samplings in the first stage (lines 4-6, Algorithm 1), the total sampling number is slightly higher than the Okamoto bound. There is a small bounce when p is near 0.2 in the $\varepsilon = \delta = 0.05$ case. It is because the algorithm sometimes underestimates p in the first stage thus uses the second estimation method (lines 20-26, Algorithm 1), which is not a good choice in this situation.

B. Local Robustness Estimation

In this subsection, we evaluate our method and the Okamoto bound on different configurations to estimate the local robustness of DNNs. For CIFAR-10, we train four popular networks and achieve about 90% testing accuracy. For ImageNet, we use four pre-trained networks provided in PyTorch. We consider six widely-used semantic perturbations from [5] and the traditional L_∞ perturbation (see Example 1). For each configuration, we run these two methods on 100 different images and report the average results (shown in Table II).

Our method only uses 48.6%-74.1% time of the Okamoto bound. In DNN local robustness estimation problems, p is more likely to be near 0 or 1 (see Fig. 1). As our method can significantly reduce the sampling number when p is near 0 or 1 (see Fig. 3), the performance of our method in practice ("Ratio" column in Table II) is much better than that in simulation ("Average" column in Table I). Moreover, the average result difference between our method and the Okamoto bound is much lower than the provided ε, which is consistent with the soundness of both methods.

C. Global Robustness Estimation

In this subsection, we use a small number of images to estimate DNN's global robustness. The goal of estimation is to achieve a high correlation coefficient with the standard method (see Problem 3 in Section IV). We use the following baseline method: "*For each given image, we generate a perturbed image. We use the classification accuracy on these perturbed images as the global robustness of the network.*" This method is widely used [5], [27] and performs well when we have a large dataset. Since the results vary with the images used, we repeat all experiments 25 times.

For CIFAR-10, we train eight networks and consider four perturbations. We randomly select 10 images from the CIFAR-10 testing set and estimate the global robustness of the eight networks using our method and the baseline method. Then we calculate the Pearson correlation coefficient with the standard method (calculated on CIFAR-10-C containing 50000 images). As shown in Table III, the baseline method achieves high correlation coefficients with the standard method. However, the correlation coefficients of our method are even higher. Note that we only use 10 images while the standard method uses 50000 images to estimate the global robustness.

For ImageNet, we consider 13 pre-trained networks and three perturbations. We use different numbers of images to estimate the global robustness and calculate the correlation coefficient with the standard method (calculated on ImageNet-C containing 250000 images). As shown in Fig. 4, the more images we use, the higher the correlation coefficient is. Moreover, our method consistently achieves higher correlation coefficients than the baseline method. Note that ImageNet contains 1000 different classes of images, so it is surprising that we can use 50 images to achieve high correlation coefficients (0.90-0.95) with the standard method using 250000 images.

VIII. CONCLUSION

In this paper, we propose two algorithms to estimate the local and global robustness of DNNs. Our local robustness estimation algorithm is sound and efficient. It can also be used to replace existing algorithms in statistical model checkers. Our global robustness estimation algorithm provides high-quality estimation using only a few images. It is useful for users to compare the global robustness of different networks.

REFERENCES

[1] M. Balunovic, M. Baader, G. Singh, T. Gehr, and M. Vechev, "Certifying geometric robustness of neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, 2019.

[2] J. Mohapatra, T.-W. Weng, P.-Y. Chen, S. Liu, and L. Daniel, "Towards verifying robustness of neural networks against a family of semantic perturbations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 244–252.

[3] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Proceedings of the 29th International Conference on Computer Aided Verification*, 2017, pp. 97–117.

[4] W. Ruan, X. Huang, and M. Kwiatkowska, "Reachability analysis of deep neural networks with provable guarantees," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018, pp. 2651–2659.

TABLE II
CONFIGURATIONS AND RESULTS OF OUR EXPERIMENTS.

Dataset	Perturbation	Network	ε	δ	Parameter	Time(s)		Ratio	Average difference of results
						Our	Okamoto		
CIFAR-10	Contrast	VGG16	0.005	0.001	U[0,0.5]*	17.23	25.94	0.664	0.001035
	Fog	DenseNet121	0.01	0.01	(1.5, 1.75)	35.38	60.42	0.586	0.002240
	Gaussian noise	ResNet101	0.02	0.02	0.1	3.22	5.61	0.575	0.004450
	Glass blur	MobileNetV2	0.05	0.05	(0.4,1,2)	56.13	82.17	0.683	0.008850
ImageNet	Contrast	ShuffleNetV2	0.01	0.01	U[0,0.5]*	44.68	91.89	0.486	0.001495
	Impulse noise	GoogLeNet	0.02	0.02	0.06	16.49	33.91	0.486	0.003086
	Shot noise	RegNet	0.05	0.05	20	0.44	0.60	0.741	0.007082
	L_inf	MNASNet	0.03	0.03	0.8	7.40	14.62	0.506	0.003614

*: Uniform distribution in [0, 0.5]

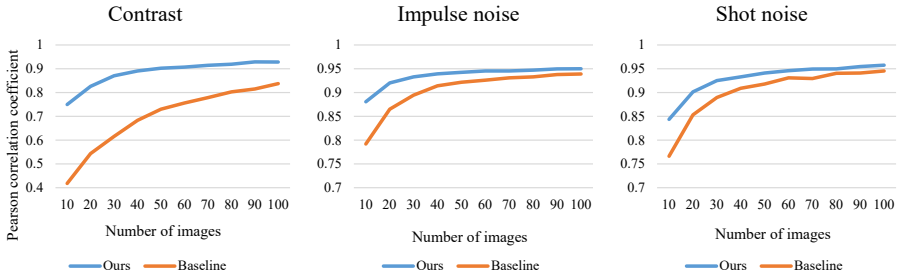

Fig. 4. Pearson correlation coefficients of our method and the baseline method with the standard method under different p.

TABLE III
PEARSON CORRELATION COEFFICIENTS OF OUR METHOD AND THE BASELINE METHOD WITH THE STANDARD METHOD.

	Ours	Baseline
Contrast	0.9631	0.8264
Fog	0.9139	0.7963
Gaussian noise	0.9835	0.9539
Glass blur	0.9473	0.8424

[5] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *Proceedings of the International Conference on Learning Representations*, 2019.

[6] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[7] H. L. Younes and R. G. Simmons, "Probabilistic verification of discrete event systems using acceptance sampling," in *Proceedings of the 14th International Conference on Computer Aided Verification*, 2002, pp. 223–235.

[8] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet, "Approximate probabilistic model checking," in *Proceedings of the 5th International Workshop on Verification, Model Checking, and Abstract Interpretation*, 2004, pp. 73–84.

[9] J.-P. Katoen, M. Khattri, and I. Zapreevt, "A markov reward model checker," in *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems*, 2005, pp. 243–244.

[10] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proceedings of the 23rd International Conference on Computer Aided Verification*, 2011, pp. 585–591.

[11] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin, "COSMOS: a statistical model checker for the hybrid automata stochastic logic," in *Proceedings of the Eighth International Conference on Quantitative Evaluation of SysTems*, 2011, pp. 143–144.

[12] M. Okamoto, "Some inequalities relating to the partial sum of binomial probabilities," *Annals of the institute of Statistical Mathematics*, vol. 10, no. 1, pp. 29–35, March 1959.

[13] J. Frey, "Fixed-width sequential confidence intervals for a proportion," *The American Statistician*, vol. 64, no. 3, pp. 242–249, 2010.

[14] Z. Chen and X. Chen, "Exact group sequential methods for estimating a binomial proportion," *Journal of Probability and Statistics*, vol. 2013, 2013.

[15] C. Jegourel, J. Sun, and J. S. Dong, "Sequential schemes for frequentist estimation of properties in statistical model checking," *ACM Transactions on Modeling and Computer Simulation*, vol. 29, no. 4, pp. 1–22, October 2019.

[16] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "UPPAAL SMC tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, August 2015.

[17] S. Webb, T. Rainforth, Y. W. Teh, and M. P. Kumar, "A statistical approach to assessing neural network robustness," in *Proceedings of the International Conference on Learning Representations*, 2019.

[18] N. Levy and G. Katz, "RoMA: A method for neural network robustness measurement and assessment," 2021, *arXiv:2110.11088*.

[19] L. Cardelli, M. Kwiatkowska, L. Laurenti, N. Paoletti, A. Patane, and M. Wicker, "Statistical guarantees for the robustness of bayesian neural networks," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019, pp. 5693–5700.

[20] T. Baluta, Z. L. Chua, K. S. Meel, and P. Saxena, "Scalable quantitative verification for deep neural networks," in *Proceedings of the 43th International Conference on Software Engineering*, 2021, pp. 312–323.

[21] K. Tit, T. Furon, and M. Rousset, "Efficient statistical assessment of neural network corruption robustness," in *Proceedings of the Advances in Neural Information Processing Systems*, 2021.

[22] M. Pautov, N. Tursynbek, M. Munkhoeva, N. Muravev, A. Petiushko, and I. Oseledets, "CC-Cert: A probabilistic approach to certify general robustness of neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[23] H. Bu and M. Sun, "Certifying semantic robustness of deep neural networks," in *Proceedings of the 27th International Conference on Engineering of Complex Computer Systems*, 2023.

[24] C. J. Clopper and E. S. Pearson, "The use of confidence or fiducial limits illustrated in the case of the binomial," *Biometrika*, vol. 26, no. 4, pp. 404–413, 1934.

[25] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *The Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, December 1952.

[26] Project website of this paper. https://github.com/H-Bu/robust-estimate.

[27] B. Wang, S. Webb, and T. Rainforth, "Statistically robust neural network classification," in *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, 2021, pp. 1735–1745.