PROVEN: Verifying Robustness of Neural Networks with a Probabilistic Approach

Tsui-Wei Weng [1] Pin-Yu Chen [*2] Lam M. Nguyen [*2] Mark S. Squillante [*2] Akhilan Boopathy [1]
Ivan Oseledets [3] Luca Daniel [1]

Abstract

We propose a novel framework PROVEN to **PRO**babilistically **VE**rify **N**eural network's robustness with statistical guarantees. PROVEN provides probability certificates of neural network robustness when the input perturbation follow distributional characterization. Notably, PROVEN is derived from current state-of-the-art worst-case neural network robustness verification frameworks, and therefore it can provide probability certificates with little computational overhead on top of existing methods such as Fast-Lin, CROWN and CNN-Cert. Experiments on small and large MNIST and CIFAR neural network models demonstrate our probabilistic approach can tighten up robustness certificate to around $1.8\times$ and $3.5\times$ with at least a 99.99% confidence compared with the worst-case robustness certificate by CROWN and CNN-Cert.

1. Introduction

Although deep neural networks have achieved unprecedented performance in many machine learning tasks, their lack of robustness against adversarial examples (Goodfellow et al., 2015; Biggio & Roli, 2017) has raised serious concerns by the research communities, as many safety-critical tasks cannot afford the potential risks incurred by adversarial attempts. Adversarial examples for different applications have been shown to be easily crafted, including image classification (Szegedy et al., 2013) speech recognition (Cisse et al., 2017), malware detection (Wang et al., 2017) and sparse regression (Chen et al., 2018).

While there is a relentless arm race in crafting adversarial examples with stronger attacking performance and in developing effective countermeasures, assessing and verifying robustness properties of neural networks provides a comprehensive and attack-independent certificate. Given a well-trained neural network model, we are interested in measuring its robustness on an arbitrary natural example $\mathbf{x}_0$ by examining if the neighborhood of $\mathbf{x}_0$ has the same prediction results; this serves as a robustness proxy for evaluating the ease with which one can turn $\mathbf{x}_0$ into adversarial examples via adversarial manipulations. Conventionally, the concept of neighborhood is characterized by an ℓ_p ball centered at $\mathbf{x}_0$ with radius ϵ for any $p \geq 1$, where larger ϵ indicates greater robustness. Ideally, we would like to find the smallest adversarial distortion imposed on $\mathbf{x}_0$ that will change the model prediction, which is known as the *minimum adversarial distortion*. Unfortunately, computing minimum adversarial distortion on neural networks with ReLU activations is shown to be an NP-complete problem (Katz et al., 2017; Sinha et al., 2018), and hence formal verification methods such as Reluplex (Katz et al., 2017) and Planet (Ehlers, 2017) are computationally demanding and cannot scale to large realistic networks.

As detailed in Section 2, current robustness verification literature mainly focuses on efficient methods for finding a tight *lower bound* on minimum distortion as a certified robustness metric. Here the term *certified* means that numerical values generated by these approaches are indeed deterministic lower bounds of minimum adversarial distortion. However, their scope is still limited to the *worst-case* setting (i.e., no exceptions are allowed), and hence there lacks proper robustness indication once the perturbation exceeds the certified range. In particular, in cases when successful adversarial perturbations are rare events (but not totally unlikely) or when some distributional characterization of input perturbations are known a priori, a *probabilistic* certificate on the confidence of certified robustness is more viable and comprehensive. In this paper, we are interested in addressing the following questions:

(a) How to provide a confidence level on the robustness certificate when the input data point is perturbed by random noises?

(b) How does such *probabilistic* certificate compare with the certificate from worst-case analysis?

[*]Alphabetical order. [1]MIT EECS. [2]IBM Research. [3]Skoltech. Correspondence to: Tsui-Wei Weng <twweng@mit.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

Perhaps surprisingly, in our experiments we find that simple random perturbations can already achieve up to 100% misclassification rates on MNIST and CIFAR networks under reasonable perturbation ranges (see Table 2), suggesting that their threats to deep neural networks could be overlooked. With prior knowledge on the input noise distributions, our probabilistic approach is able to provide a more comprehensive robustness quantification in comparison to the prevailing worst-case analysis. More importantly, the probabilistic robustness quantification is readily applicable to understanding the sensitivity and reliability of a target model in many real-world scenarios. For example, such random noises can be caused by data quantization, input preprocessing, or environmental background noises.

In summary, we propose in this work a novel probabilistic framework PROVEN to **PRO**babilistically **VE**rify **N**eural network robustness. We show that it is possible to extend the conventional worst-case setting to a probabilistic setting based on existing worst-case certification frameworks with very little computational overhead, meaning that the probabilistic certificate can be easily derived and incorporated into worst-case robustness verification pipelines by methods such as Fast-Lin (Weng et al., 2018), CROWN (Zhang et al., 2018), and CNN-Cert (Boopathy et al., 2019).

Our Contributions

- A probabilistic framework PROVEN is proposed for verifying the robustness of neural networks with certificates under ℓ_p norm-ball bounded threat models, when the input noise follows a given distributional characterization (Gaussian and Sub-Gaussian distributions with bounded supports). The established theoretical results are based on an ℓ_∞ constraint on the perturbation but can be easily extended to other norms such as ℓ_1 and ℓ_2.
- Experimental results on large neural networks trained on MNIST and CIFAR datasets show that PROVEN greatly improved the robustness certificate (i.e., the certified lower bound) compared with the corresponding worst-case analysis results, even when the statistical risk is small. For example, with a confidence level of 99.99%, which means the robustness metric is almost 100% guaranteed to be certified, the robustness certificate improvement provided by PROVEN over worst-case analysis can be as high as 250%.
- In addition to the noticeable improvement in the verified robustness, PROVEN is a general tool that can be readily applied to neural networks with general activation functions, including but not limited to tanh, sigmoid and arctan, and general convolutional neural networks with various building blocks, as demonstrated in our experiments. Moreover, PROVEN is as computationally efficient as the worst-case analysis because its closed-form probabilistic certificate are byproducts of worst-case certification frameworks such as Fast-Lin (Weng et al., 2018), CROWN (Zhang et al., 2018), and CNN-Cert (Boopathy et al., 2019).

2. Background and Related Works

Given an input data example under a specified threat model, typically ℓ_p norm-ball bounded perturbation attacks, the goal of robustness verification aims to certify a perturbation level ϵ such that the top-1 prediction will not be altered by any means. However, certifying the largest possible ϵ, which is equivalent to finding the minimum perturbation required for a successful adversarial attack, has been shown to be NP-complete (Katz et al., 2017) and thus it is computationally infeasible for large realistic networks. Alternatively, recent studies have shown that solving for a non-trivial lower bound of the minimum adversarial perturbation can be made more scalable and computationally efficient (Weng et al., 2018; Gehr et al., 2018; Dvijotham et al., 2018b). Some analytical lower bounds, based solely on model weights, have been derived (Szegedy et al., 2013; Peck et al., 2017; Hein & Andriushchenko, 2017; Raghunathan et al., 2018) but they are either very close to trivial lower bounds (zero) or only applied to 1 or 2 hidden layers. It is worth noting that current robustness verification approaches mainly focus on "worst-case" analysis, whereas our approach takes a "probabilistic viewpoint". As demonstrated in extensive experiments, our probabilistic framework PROVEN is able to certify a significantly larger ϵ value than the corresponding worst-case analysis with 99.99% certification guarantees. This indicates that, while the conventional worst-case robustness certificate may be too conservative when there is some prior knowledge about the input perturbations distribution, PROVEN is more applicable in this situation.

In fact, deep neural networks are not only vulnerable to crafted adversarial noises but also to random noises: (Bibi et al., 2018) show that they can fool LeNet and AlexNet with additive Gaussian noises and (Fawzi et al., 2016) show random perturbations can indeed fool VGG networks; (Hosseini et al., 2017) show they fool the Google Cloud Vision API by random Gaussian noises, suggesting random perturbation can result in successful attacks. Meanwhile, the robustness of classifiers to various kinds of random noises, such as uniform noise in the ℓ_p unit ball and Gaussian noise with an arbitrary covariance matrix, has been studied in (Franceschi et al., 2018). Their analysis, however, requires the assumption of locally approximately flat decision boundaries on the neural networks, which is difficult to verify in reality. Recently, the robustness of classifiers to perturbations under the assumption of Gaussian distributed latent input vectors has been studied in (Fawzi et al., 2018); however, their results depend on the modulus of continuity constant, which could be arbitrarily large. Due to these limitations, the bounds in these papers cannot be directly used to deliver certified robustness metrics.

We note that only a few works have considered verifying neural network properties in the probabilistic setting. (Dvijotham et al., 2018a) verifies some properties (e.g., monotonicity and convexity) of deep generative models. The key differences are that in their setting the uncertainty source is the latent variable sampled from a distribution generated by the encoder (they consider only Gaussian distribution), whereas our uncertainty comes from input perturbations (we consider both Gaussian and general Subgaussian distributions with bounded support) and our focus is to verify neural network classifiers. A very recent work (Webb et al., 2019) uses a Monte Carlo approach to *estimate* the probability of rare events, whereas the violation probability from our framework is certifiable. We also highlight that the problem setting of this work is different from (Lecuyer et al., 2018; Li et al., 2018) as our goal is to study the effect of random noises on *standard* neural networks input and derive a closed-form probabilistic certificate; on the other hand, they study *smoothed* (or stochastic) classifiers by adding a noise layer into the network and thus modifying the prediction rule. The predicted class in their setting becomes the class that has the largest expected score (Lecuyer et al., 2018) or the largest probability (Li et al., 2018) over input with noises, which is different from the rule of standard networks.

3. PROVEN: a probabilistic framework to verify neural network robustness

In this section, we present a general probabilistic framework PROVEN together with related theoretical results to compute the certified bounds in probability that a classifier can never be fooled when the inputs of the classifier are perturbed with some given distributions. We first introduce the worst-case setting, where an input example can be perturbed by any perturbation bounded within an ℓ_p ball, and present corresponding worst-case analysis results in Sec. 3.1. We then show that it is possible to extend these worst-case analysis results to a probabilistic setting where the input perturbations follow some given distributions, and present our probabilistic framework and main theorem in Sec. 3.2. Lastly, we provide *closed-form* probabilistic bounds for various probabilistic distributions that the input perturbations can follow in Sec. 3.3.

3.1. Worst-case setting

Let $f(\mathbf{x}) : \mathbb{R}^{n_0} \rightarrow \mathbb{R}^K$ denote a K-class neural network classifier of interest, which takes an input $\mathbf{x}$ (e.g., an image) and outputs the corresponding logit scores over all classes. The ultimate goal is to efficiently find the largest ϵ^* such that the original predicted class c always has a larger score $f_c(\mathbf{x})$ than the score $f_t(\mathbf{x})$ for a targeted attack class t when the input is perturbed within the ℓ_p ball having radius ϵ^*. Let $g_t(\mathbf{x}) = f_c(\mathbf{x}) - f_t(\mathbf{x})$, we want to find the largest ϵ^* such that $g_t(\mathbf{x}) > 0$ for all $\mathbf{x}$ satisfying $\|\mathbf{x} - \mathbf{x_0}\|_p \leq \epsilon^*$ and $c = \text{argmax}_i f_i(\mathbf{x_0}), t \neq c$. This ϵ^* is a *certified lower bound* of the minimum adversarial distortion introduced in Sec. 1.

Although a neural network classifier f is a *nonlinear*, *non-convex* and complicated function, it has been first shown in Fast-Lin (Weng et al., 2018) and later in CROWN (Zhang et al., 2018) and CNN-Cert (Boopathy et al., 2019) that the output $f_i(\mathbf{x})$ and the margin function $g_t(\mathbf{x})$ of a general (convolutional) neural network classifier with general activation functions[1] can be bounded by two *linear functions* $g_t^L(\mathbf{x})$ and $g_t^U(\mathbf{x})$:

$$g_t^L(\mathbf{x}) \leq g_t(\mathbf{x}) \leq g_t^U(\mathbf{x}), \tag{1}$$

where $g_t^L(\mathbf{x}) : \mathbb{R}^{n_0} \rightarrow \mathbb{R}$ and $g_t^U(\mathbf{x}) : \mathbb{R}^{n_0} \rightarrow \mathbb{R}$ are two linear functions in terms of the input $\mathbf{x}$:

$$g_t^L(\mathbf{x}) = \mathbf{A}_{t,:}^L \mathbf{x} + d^L \quad \text{and} \quad g_t^U(\mathbf{x}) = \mathbf{A}_{t,:}^U \mathbf{x} + d^U \tag{2}$$

with $\mathbf{A}_{t,:}^L, \mathbf{A}_{t,:}^U \in \mathbb{R}^{1 \times n_0}$ being two constant row vectors and $d^L, d^U \in \mathbb{R}$ being two constants related to the network weights $\mathbf{W}^{(k)}$ and biases $\mathbf{b}^{(k)}$ as well as the parameters bounding the activation functions in each neuron. The superscripts L and U denote the parameters corresponding to the lower bound and the upper bound of $g_t(\mathbf{x})$.

Remark. We would like to highlight again that, same as Fast-Lin, CROWN and CNN-Cert, we *do not* assume the neural network $f(\mathbf{x})$ and its variant $g_t(\mathbf{x})$ to be linear. Through out this paper, $f(\mathbf{x})$ and $g_t(\mathbf{x})$ are a general (convolutional) neural network with general non-linear activation function and thus are always **non-linear** and **non-convex** functions. The core idea of their proposed worst-case frameworks is to use the **linear upper and lower bound** $g_t^U(\mathbf{x}), g_t^L(\mathbf{x})$ to **bound the non-linear neural network** $g_t(\mathbf{x})$, as in (1). Again, the *linear* bounds have superscripts L and U while the non-linear neural network functions does not. Importantly, the result in (1) is the key to develop our probabilistic framework PROVEN. We also highlight that $\mathbf{A}^L = \mathbf{A}^U$ in Fast-Lin due to the use of parallel linear bounds on non-linear activation functions, whereas $\mathbf{A}^L$ and $\mathbf{A}^U$ can be different in CROWN and CNN-Cert due to non-parallel linear bounds.

As the network output is bounded, the positiveness of the lower bound of $g_t(\mathbf{x})$ implies that $g_t(\mathbf{x})$ is positive, i.e.,

$$g_t^L(\mathbf{x}) > 0 \implies g_t(\mathbf{x}) > 0. \tag{3}$$

[1]including but not limited to ReLU, tanh, arctan, sigmoid

Table 1: Table of Notation

Notation	Definition	Notation	Definition
K	number of output classes	CDF	cumulative distribution function
$f : \mathbb{R}^{n_0} \to \mathbb{R}^K$	neural network classifier	pdf	probability density function
$\mathbf{x_0} \in \mathbb{R}^{n_0}$	original input vector	F_X	CDF of a random variable X
$c = \text{argmin}_i\, f_i(\mathbf{x_0})$	predicted class of input $\mathbf{x_0}$	f_X	pdf of a random variable X
$g_t(\mathbf{x}) = f_c(\mathbf{x}) - f_t(\mathbf{x})$	margin function at $\mathbf{x}$ for class t	$\mathbb{P}[g_t(X) > a]$	probability that $g_t(X)$ is greater than a
$g_t^L(\mathbf{x}) : \mathbb{R}^{n_0} \to \mathbb{R}$	linear lower bound of $g_t(\mathbf{x})$	γ_L	theoretical lower bound of $\mathbb{P}[g_t(X) > a]$
$g_t^U(\mathbf{x}) : \mathbb{R}^{n_0} \to \mathbb{R}$	linear upper bound of $g_t(\mathbf{x})$	γ_U	theoretical upper bound of $\mathbb{P}[g_t(X) > a]$

Here, a *worst-case* analysis can be performed by minimizing the linear function $g_t^L(\mathbf{x})$ over all possible inputs in the set $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x_0}\|_p \leq \epsilon\}$, which yields a closed-form solution as presented in Fast-Lin, CROWN and CNN-Cert. Therefore, the condition of whether $g_t^L(\mathbf{x}) > 0$ can be conveniently checked given some ϵ using the closed-form solutions; the largest ϵ such that $g_t^L(\mathbf{x}) > 0$ is called the *certified lower bound*, which can be computed by bisection with respect to ϵ.

3.2. Our proposed probabilistic framework: PROVEN

In addition to considering the worst-case condition for $g_t(\mathbf{x}) > 0$ over the norm ball constrained on the input $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x_0}\|_p \leq \epsilon\}$, we now show that it is possible to formulate a probabilistic setting and derive bounds with guarantees by building upon the above results that the neural network output can be bounded by two linear functions. We start by presenting the problem formulation of the probabilistic setting and then present our main theoretical results in Theorem 3.1.

Problem formulation. Consider a general **non-linear** neural network classifier $f(\mathbf{x})$ and an input example $\mathbf{x_0}$. Let the predicted class of $\mathbf{x_0}$ be c, the targeted attack class t, and the **non-linear** margin function $g_t(\mathbf{x}) := f_c(\mathbf{x}) - f_t(\mathbf{x})$. Suppose the perturbed input random vector X follows some given distribution $\mathcal{D}$, i.e., $X \sim \mathcal{D}$. We are interested in the probability of the margin function $g_t(\mathbf{x})$ being greater than some value $a \in \mathbb{R}$, i.e., $\mathbb{P}\left[g_t(X) > a\right]$.

Given that the general neural networks are highly **non-linear** and **non-convex** in $\mathbf{x}$, it is hard to directly compute the distribution of the **non-linear** function $g_t(X)$ given the input $X \sim \mathcal{D}$. Fortunately, we can still derive *analytic lower and upper bounds* for $\mathbb{P}\left[g_t(X) > 0\right]$ based on the **linear** function $g_t^L(X)$ thanks to the relations of **linear** $g_t^L(\mathbf{x})$ and **non-linear** $g_t(\mathbf{x})$ in Eq. (1). In other words, we can obtain probabilistic guarantees based on the worst-case analysis result where the **non-linear** margin function $g_t(x)$ can be bounded by two **linear** functions. The following theorem provides such theoretical guarantees on $\mathbb{P}\left[g_t(X) > 0\right]$.

Theorem 3.1 (Probabilistic bounds of network output) *Let $f(\mathbf{x})$ be a K-class neural network classifier function, $\mathbf{x_0}$ an input example, and ϵ such that $\|\mathbf{x} - \mathbf{x_0}\|_p \leq \epsilon$, $p \geq 1$. Let $c = argmax_i f_i(\mathbf{x_0})$, let $t(\neq c)$ be some targeted class, and define the margin function $g_t(\mathbf{x}) = f_c(\mathbf{x}) - f_t(\mathbf{x})$. Suppose the input random vector $X \in \mathbb{R}^{n_0}$ follows some given distribution $\mathcal{D}$ with mean $\mathbf{x_0}$ and let $a \in \mathbb{R}$ be some real number. There exists an explicit lower bound γ_L and an explicit upper bound γ_U on the probability $\mathbb{P}\left[g_t(X) > a\right]$ such that*

$$\gamma_L \leq \mathbb{P}\left[g_t(X) > a\right] \leq \gamma_U, \tag{4}$$

where

$$\gamma_L = 1 - F_{g_t^L(X)}(a), \quad \gamma_U = 1 - F_{g_t^U(X)}(a), \tag{5}$$

$F_Z(z)$ is the cumulative distribution function (CDF) of the random variable Z, and $g_t^L(\mathbf{x})$, $g_t^U(\mathbf{x})$ satisfy Equation (1).

Proof. Let $h_1 : \mathbb{R}^d \to \mathbb{R}$, $h_2 : \mathbb{R}^d \to \mathbb{R}$, and $h_1(\mathbf{x}) \geq h_2(\mathbf{x})$, $\forall \mathbf{x} \in \mathbb{R}^d$. Let $X \in \mathbb{R}^d$ be a random vector. Since the cumulative distribution function (CDF) is nondecreasing (Shaked & Shanthikumar, 2007), we have

$$\mathbb{P}[h_1(X) > a] \geq \mathbb{P}[h_2(X) > a]. \tag{6}$$

From the results in (Weng et al., 2018), we know that the relationships in Eq. (1), i.e.,

$$g_t^L(\mathbf{x}) \leq g_t(\mathbf{x}) \leq g_t^U(\mathbf{x}),$$

satisfy $\|\mathbf{x} - \mathbf{x_0}\|_p \leq \epsilon$ for all $\mathbf{x}$. Hence, upon applying Eq. (6) to Eq. (1) and using the fact that $\mathbb{P}\left[Z > a\right] = 1 - F_Z(a)$, we obtain

$$\gamma_L \leq \mathbb{P}\left[g_t(X) > a\right] \leq \gamma_U,$$

with $\gamma_L = 1 - F_{g_t^L(X)}(a)$, $\gamma_U = 1 - F_{g_t^U(X)}(a)$. □

As discussed in Section 3.1, the neural network output and the margin function can be bounded by two linear functions as derived in Fast-Lin and its extensions CROWN and CNN-Cert. Here, we take an additional step to investigate the relationship between the margin function and its linear bounds in the probabilistic setting. Specifically, Theorem 3.1 shows that the probability of the neural network margin function being greater than some value a can also be bounded by the CDFs of its linear bounds. Note that in the worst-case analysis of Section 3.1, we usually concern ourselves with the margin function $g_t(x) > 0$, i.e., $a = 0$. Analogously, in the probabilistic setting, we concern ourselves with the probability of the margin function $g_t(x) > 0$. This is indeed the guarantee provided by Theorem 3.1: when the input $X \sim \mathcal{D}$, the result guarantees that the probability of $g_t(X) > a$ is at least γ_L and at most γ_U.

3.3. Evaluating the probabilistic bounds γ_L and γ_U

Theorem 3.1 provides a theoretical lower bound γ_L and upper bound γ_U for $\mathbb{P}\left[g_t(X) > a\right]$. In practice, we would like to numerically compute such bounds. Below we show it is possible to obtain explicit forms for γ_L and γ_U in terms of $\mathbf{A}^L_{t,:}, \mathbf{A}^U_{t,:}, d^L, d^U$, as well as the parameters of the probability distributions of input perturbations. By Theorem 3.1, γ_L and γ_U only depend on the CDFs $F_{g^L_t(X)}$ and $F_{g^U_t(X)}$, and we observe that $g^L_t(X)$ and $g^U_t(X)$ are both linear functions of random vector X as follows:

$$g^L_t(X) = \sum_{i=1}^{n_0} \mathbf{A}^L_{t,i} X_i + d^L,$$
$$g^U_t(X) = \sum_{i=1}^{n_0} \mathbf{A}^U_{t,i} X_i + d^U.$$

Hence, the problem of computing the CDFs $F_{g^L_t(X)}$ and $F_{g^U_t(X)}$ becomes a problem of computing the CDFs of a weighted sum of X_i given $X \sim \mathcal{D}$. We primarily consider the following two cases:

(i) When X_i are independent random variables with probability density function (pdf) f_{X_i};

(ii) When X follows a multivariate normal distribution with mean $\mathbf{x_0}$ and covariance Σ.

It also appears that these results may be extended to address some forms of negative correlation (Panconesi & Srinivasan, 1997; Dubhashi & Ranjan, 1998).

3.3.1. Case (i)

When X_i are independent random variables with probability density function f_{X_i}, there are two approaches for computing the CDFs of the weighted sum.

Approach 1: Direct convolutions. The pdf of the weighted sum is simply the convolution of the pdfs for each of the weighted random variables $\mathbf{A}^L_{t,i} X_i$. Specifically, we have

$$f_{\mathbf{A}^L_{t,:}X} = \circledast_{i=1}^{n_0} f_{\mathbf{A}^L_{t,i}X_i},$$

where $\circledast_{i=1}^{N} h_i$ denotes convolution over the N functions h_1 to h_N. The CDF of $\mathbf{A}^L_{t,:}X$ can therefore be obtained from the pdf $f_{\mathbf{A}^L_{t,:}X}$ and we obtain $F_{g^L_t(X)}(z) = F_{\mathbf{A}^L_{t,:}X}(z - d^L)$; similarly, $F_{g^U_t(X)}(z) = F_{\mathbf{A}^U_{t,:}X}(z - d^U)$. Hence, we have

$$\gamma_L = 1 - F_{\mathbf{A}^L_{t,:}X}(a - d^L), \quad \gamma_U = 1 - F_{\mathbf{A}^U_{t,:}X}(a - d^U).$$

Approach 2: Probabilistic inequalities. Approach 1 is useful in cases where n_0 is not large. However, for large n_0, it might not be easy to directly compute the CDF through convolutions. For such cases, an alternative approach can be based on applying the probabilistic inequalities on the CDFs. Since we want to provide *guarantees* on the desired probability in (4), we need to find a lower bound on γ_L and an upper bound on γ_U via the probabilistic inequalities. These results are given in the following corollary.

Corollary 3.2 *Let X_i be independent random variables following Sub-Gaussian distributions with bounded support $X_i \in [\mathbf{x_0}_i - \epsilon, \mathbf{x_0}_i + \epsilon], \forall i \in [n_0]$, and symmetric around the mean $\mathbf{x_0}_i$. Define*

$$\mu_L = \mathbf{A}^L_{t,:}\mathbf{x_0} + d^L,\ \mu_U = \mathbf{A}^U_{t,:}\mathbf{x_0} + d^U.$$

Then, we have

$$\gamma_L \geq \begin{cases} 1 - \exp\left(-\frac{(\mu_L - a)^2}{2\epsilon^2 \|\mathbf{A}^L_{t,:}\|_2^2}\right), & \textit{if } \mu_L - a \geq 0 \\ 0, & \textit{otherwise}; \end{cases}$$

$$\gamma_U \leq \begin{cases} \exp\left(-\frac{(\mu_U - a)^2}{2\epsilon^2 \|\mathbf{A}^U_{t,:}\|_2^2}\right), & \textit{if } -\mu_U + a \geq 0 \\ 1, & \textit{otherwise}. \end{cases}$$

Proof. Let $W_i = \mathbf{A}^L_{t,i}(X_i - \mathbf{x_0}_i)$ and $\mu_L = \mathbf{A}^L_{t,:}\mathbf{x_0} + d^L$. We then have $-|\mathbf{A}^L_{t,i}|\epsilon \leq W_i \leq |\mathbf{A}^L_{t,i}|\epsilon$ where W_i is symmetric with respect to zero since X_i is symmetric. By using the fact that the sum of independent symmetric random variables is still a symmetric random variable (Chow & Teicher, 2003), we derive

$$\begin{aligned} \gamma_L &= \mathbb{P}\left[g^L_t(X) > a\right] \\ &= \mathbb{P}\left[\sum_{i=1}^{n_0} W_i > a - \mu_L\right] \\ &= \mathbb{P}\left[\sum_{i=1}^{n_0} W_i < -a + \mu_L\right] \\ &= 1 - \mathbb{P}\left[\sum_{i=1}^{n_0} W_i \geq -a + \mu_L\right]. \end{aligned}$$

From the Hoeffding inequality (Resnick, 2014), we obtain the following upper bound on the term $\mathbb{P}\left[\sum_{i=1}^{n_0} W_i \geq -a + \mu_L\right]$ when $-a + \mu_L > 0$:

$$\mathbb{P}\left[\sum_{i=1}^{n_0} W_i \geq -a + \mu_L\right] \leq \exp\left(-\frac{(\mu_L - a)^2}{2\epsilon^2 \|\mathbf{A}^L_{t,:}\|_2^2}\right),$$

and thus $\gamma_L \geq 1 - \exp\left(-\frac{(\mu_L - a)^2}{2\epsilon^2 \|\mathbf{A}^L_{t,:}\|_2^2}\right)$. When $-a + \mu_L \leq 0$, we use the trivial bound of $\gamma_L = 0$. Similarly, for γ_U, we can define μ_U correspondingly and directly apply the Hoeffding inequality to obtain $\gamma_U \leq \exp\left(-\frac{(\mu_U - a)^2}{2\epsilon^2 \|\mathbf{A}^U_{t,:}\|_2^2}\right)$, or use the trivial bound of $\gamma_U = 1$. □

Remark. The above result can be further extended to correlated random variables with Hoeffding inequality similarly with the assumption that the sum of W_i is symmetric.

3.3.2. Case (ii)

When X follows a multivariate normal distribution with mean $\mathbf{x_0}$ and covariance Σ, we are able to obtain an explicit form for the CDFs $F_{g_t^L(X)}$ and $F_{g_t^U(X)}$ based on the fact that the sum of normally distributed random variables still follows the normal distribution (Chow & Teicher, 2003). Note that we include here both cases where (a) X_i are independent Gaussian random variables (Σ is a diagonal matrix) and (b) X_i are correlated random variables (Σ is a general covariance matrix and positive semidefinite). Our result is stated in the following corollary.

Corollary 3.3 *Let X follow a multivariate normal distribution with mean $\mathbf{x_0}$ and covariance Σ. Define*

$$\mu_L = \mathbf{A}^L_{t,:}\mathbf{x_0} + d^L,\ \sigma_L^2 = \mathbf{A}^L_{t,:}\Sigma(\mathbf{A}^L_{t,:})^\top,$$
$$\mu_U = \mathbf{A}^U_{t,:}\mathbf{x_0} + d^U,\ \sigma_U^2 = \mathbf{A}^U_{t,:}\Sigma(\mathbf{A}^U_{t,:})^\top,$$

where $\top$ denotes the transpose operator. We then have

$$\gamma_L \approx \frac{1}{2} - \frac{1}{2} erf(\frac{a - \mu_L}{\sigma_L\sqrt{2}}), \quad \gamma_U \approx \frac{1}{2} - \frac{1}{2} erf(\frac{a - \mu_U}{\sigma_U\sqrt{2}})$$

with $erf(\cdot)$ as the error function.

Proof. The result is obtained in a straightforward manner from the fact (Chow & Teicher, 2003) that if $X \sim \mathcal{N}(\mu, \Sigma)$, then its linear combination $Z = wX + v$ also follows the normal distribution: $Z \sim \mathcal{N}(w\mu + v, w\Sigma w^\top)$. The CDF of Z is then given by $\frac{1}{2}(1 + \text{erf}(\frac{z-\mu_Z}{\sigma_Z\sqrt{2}}))$, leading to the stated approximations. □

Remark. Note that in our framework, all possible inputs have to lie in the ℓ_p ball with given radius ϵ. Thus, in order to apply the Gaussian perturbation in our setting, we need to set an upper limit on the variance of the input such that 99.7% of the density is within the ℓ_p ball, i.e., the 3-σ rule. See Sec. 4 **Experiment (b)** for more details.

Connection to ℓ_1 and ℓ_2 norms. Our foregoing probabilistic analysis is established under the ℓ_∞ norm constraint. We note that this presented analysis can be easily extended to ℓ_1 and ℓ_2 norms by using the norm inequalities: $\|\mathbf{x}\|_1 \leq \sqrt{n_0}\|\mathbf{x}\|_2 \leq n_0\|\mathbf{x}\|_\infty$.

Table 2: **Attacks with Uniform & Bernoulli noises**: success rate over 100 randomly selected images.

Perturbed ℓ_∞ magnitude	$\epsilon = 0.25$		$\epsilon = 0.20$	
MNIST model	Uniform	Bernoulli	Uniform	Bernoulli
2-layer CNN, ReLU	25%	72%	15%	65%
2-layer CNN, tanh	91%	99%	83%	98%
2-layer CNN, sigmoid	92%	100%	15%	44%
2-layer CNN, arctan	7%	44%	22%	22%
3-layer CNN, ReLU	69%	90%	53%	99%
3-layer CNN, tanh	11%	25%	0%	41%
3-layer CNN, sigmoid	14%	24%	30%	76%
3-layer CNN, arctan	24%	83%	55%	73%
Perturbed ℓ_∞ magnitude	$\epsilon = 0.025$		$\epsilon = 0.020$	
CIFAR model	Uniform	Bernoulli	Uniform	Bernoulli
5×[2048], ReLU	15%	16%	13%	15%
6×[2048], ReLU	17%	20%	14%	20%
5-layer CNN, ReLU	22%	31%	17%	28%

4. Experiments

In this section, we conduct two major experiments:

(a) attack neural network models with random noises

(b) calculate the robustness bounds certified by PROVEN with various confidence levels

The goal of the Experiment (a) is to validate the issues that neural networks are also vulnerable to random noises in addition to the specifically crafted adversarial noises, and Experiment (b) is to demonstrate the effectiveness and capability of our proposed probabilistic framework PROVEN and compare with the conservative worst-case certifications.

Model, Dataset, and Setup. We use the pre-trained MNIST and CIFAR networks provided in (Weng et al., 2018) and (Zhang et al., 2018) and denote a network with m layers and n neurons per layer as $m \times [n]$ in the Tables. In addition to the pre-trained fully-connected models, we also train (1) 2 layer and 3 layer MNIST CNNs with 5 filters and ReLU, tanh, sigmoid and arctan activations (2) MNIST ResNet with 3 residual blocks (3) CIFAR 5-layer CNN, (4) 7-layer CNN on TinyImageNet. In addition, we also train some *robust* models by adversarial training (Madry et al., 2018) with 2 and 3 layer CNN structures. All the data are normalized to the range $[-0.5, 0.5]$. We implement PROVEN in Python and perform experiments on a laptop with 8 Intel Cores i7-4700 HQ CPU at 2.40 GHz and a AMD Zen sever CPU. Our code is available at `https://github.com/lilyweng/PROVEN`.

Experiment (a). We generate random noises with range $[-\epsilon, \epsilon]$ following uniform distribution and Bernoulli distribution with coin probability 0.5, and apply the random noises on each pixel. We generate at most 3×10^5 samples to perform random attacks on 100 randomly selected test images with correct prediction. We report the success rate and the corresponding ϵ for various MNIST and CIFAR (convolution) neural networks in Table 2. The results show that random noises can indeed achieve successful attack with surprisingly rates, up to 100%, thus validating the issues of random noises and the necessity to analyze its effects, as one of the main purpose of this work. We note that for $\epsilon = 0.25$, the attacks on CIFAR models almost all succeed and thus we decrease the ϵ by ten-fold.

Experiment (b). We apply Corollaries 3.2 and 3.3 (i.e. Approach 2 in Sec. 3.3) to compute the largest ϵ (denoted as ϵ_{PROVEN}) that PROVEN can certify with confidence of at least γ_L when the input follows the two cases discussed in Section 3.3. The certified lower bound computed by the worst-case analysis in Fast-Lin (Weng et al., 2018), CROWN (Zhang et al., 2018) and CNN-Cert (Boopathy et al., 2019) is denoted as $\epsilon_{\text{worst-case}}$. Below is the setting of the input distributions in our simulations:

- **Case (i)**. X_i are independent random variables following Sub-Gaussian distribution with bounded support $[\mathbf{x_0}_i - \epsilon_{\text{worst-case}}, \mathbf{x_0}_i + \epsilon_{\text{worst-case}}]$ with mean $\mathbf{x_0}_i$. The results are presented in Table 3.
- **Case (ii)**. X follows a multivariate normal distribution with mean $\mathbf{x_0}$ and covariance Σ. We consider both situations where Σ is a positive diagonal matrix or a positive semidefinite matrix with diagonals whose square roots are less than or equal to $\epsilon_{\text{worst-case}}/3$. The results are presented in Figure 1 and Tables 8 and 9 in the Appendix.

Note that in all the Tables, we express γ_L as a percentage. We report ϵ_{PROVEN} for the following values: $\{99.99, 75, 50, 25, 5\}\%$ and calculate the improvement of ϵ_{PROVEN} over $\epsilon_{\text{worst-case}}$ obtained by 99.99% in the last column in Table 3 for Case (i). The results in Table 3 are averaged over 10 randomly selected images in the test sets for MLP networks and 100 images for the CNN networks. On the other hand, we also investigate how robust it is for the results in Table 3(a) by computing the average ϵ_{PROVEN} over randomly chosen $\{10, 50, 100\}$ images in 100 random trials. We report the mean and standard deviation in Table 3(b) and show that the variation of using 10 sample average in Table 3(a) is less $\sim 10\%$ and the average ϵ_{PROVEN} and improvement has less deviations when we use 50 or 100 samples.

Result on small and large ReLU networks. We perform simulations on both small 2-3 layer MNIST networks with 20 neurons per layer and large 2-7 layer MNIST and CIFAR networks with 1024 or 2048 neurons per layer; the full results are summarized in the appendix and we extract some results in Tables 3(a). These results show that on the small networks, PROVEN can certify up to $1.8\times$ with respect to the certified lower bound at the expense of decreasing the confidence by only $\eta = 10^{-2}$. In other words, PROVEN guarantees that at least 99.99% of the ϵ computed (e.g., 0.03828 in MNIST $3\times[20]$ is a certified lower bound as compared to 0.02236 for the $\epsilon_{\text{worst-case}}$ delivered by CROWN (Zhang et al., 2018), where the improvement we obtained for this model is $1.7\times$. For large MLP networks, PROVEN can certify up to $1.8\times$, which is significant. Interestingly, when the bounding technique is better, it also helps our probabilistic bounds – the improvement is significant, and even for the large CIFAR network with around 10,000 neurons, we can still obtain around $1.15\times$ tightness. For input Gaussian perturbations , the results are presented in Fig. 1 in the Appendix.

Results on large networks with general activations and CNNs. We also ran experiments on various MNIST and CIFAR networks with non-ReLU activations, e.g., tanh, sigmoid and arctan and CNN structures. The results are summarized in Table 3. In comparison to the same architecture but with ReLU activations, the improvement of these activations are better than the non-adaptive bounding technique in general, and can achieve up to 300% on CNN networks. Note that the computational overhead of our approach compared to the worst-case certification frameworks Fast-Lin, CROWN and CNN-Cert is very little, as we only need to perform a few additional binary searches on the ϵ that will satisfy Corollary 3.2.

Discussions. We observed that the improvement of certificate increase when we use better worst-case framework (e.g. CROWN, CNN-Cert). For MNIST 3×20 network, the PROVEN certificate can be improved from $1.3\times$ to $1.7\times$ at 99.99% confidence level. Similar result can be observed by comparing Table 3a and 5a. We also observed the gap generally becomes smaller when the network becomes deeper and our hypothesis is that for deeper networks, linear bounds become looser and variance also increases.

5. Conclusions and future works

We proposed a novel probabilistic framework PROVEN to verify the robustness of neural networks and derived theoretical bounds on the robustness certificate with statistical guarantees. PROVEN is a general tool that can build on top of existing state-of-the-art neural network robustness certification algorithms including Fast-Lin, CROWN and CNN-Cert and hence can be readily applied to certify fully-connected and convolutional neural networks with different activation functions. Experimental results on large neural networks demonstrated significant benefits of PROVEN over the standard worst-case analysis results. Future works include extending our analysis to non-symmetric noise and tighter bounds for correlated bounded Sub-Gaussian noise.

Table 3: The largest ϵ that PROVEN can certify with confidence of at least $\gamma_L = \{99.99, 75, 50, 25, 5\}\%$ when X_i are independent random variables in Case (i). We compare the largest ϵ that PROVEN can certify with 99.99% with the largest ϵ from state-of-the-art worst-case certification algorithms CROWN (Zhang et al., 2018) (for MLPs), CNN-Cert (Boopathy et al., 2019) (for CNNs) and show in the last column that PROVEN can certify more than the worst-case analysis by giving up 0.01% confidence.

(a) Compare PROVEN with worst-case bounds on various neural networks models

Certification Method	Worst-case	Our probabilistic approach: PROVEN					Certification
Guarantees γ_L	100%†	99.99%†	75%	50%	25%	5%	Bound increase†
MNIST 3×[20], ReLU, ada	0.02236	**0.03828**	0.03966	0.03981	0.03995	0.04009	**1.7X**
MNIST 2×[1024], ReLU, ada	0.03158	**0.05560**	0.05756	0.05779	0.05798	0.05818	**1.8X**
MNIST 3×[1024], ReLU, ada	0.02397	**0.03524**	0.03583	0.03589	0.03595	0.03601	**1.5X**
MNIST 4×[1024], ReLU, ada	0.00962	**0.01288**	0.01293	0.01294	0.01295	0.01295	**1.3X**
CIFAR 5×[2048], ReLU, ada	0.00228	**0.00264**	0.00265	0.00265	0.00265	0.00265	**1.2X**
CIFAR 7×[1024], ReLU, ada	0.00189	**0.00209**	0.00210	0.00210	0.00210	0.00210	**1.1X**
MNIST 2×[1024], tanh	0.02232	**0.02915**	0.03005	0.03013	0.03022	0.03033	**1.3X**
MNIST 3×[1024], tanh	0.01121	**0.01360**	0.01376	0.01378	0.01380	0.01381	**1.2X**
MNIST 4×[1024], sigmoid	0.01778	**0.02170**	0.02224	0.02229	0.02232	0.02237	**1.2X**
MNIST 2×[1024], arctan	0.02105	**0.02796**	0.02907	0.02915	0.02924	0.02936	**1.3X**
MNIST 2-layer CNN, ReLU	0.04565	**0.06367**	0.06884	0.06989	0.07082	0.07181	**1.4X**
MNIST 2-layer CNN, tanh	0.0331	**0.09987**	0.13538	0.1437	0.15135	0.15981	**3.0X**
MNIST 2-layer CNN, sigmoid	0.09242	**0.18777**	0.2218	0.22906	0.23553	0.24243	**2.0X**
MNIST 2-layer CNN, arctan	0.03747	**0.13114**	0.18872	0.20279	0.21577	0.23028	**3.5X**
MNIST 3-layer CNN, ReLU	0.04609	**0.06301**	0.0674	0.06828	0.06904	0.06986	**1.4X**
MNIST 3-layer CNN, tanh	0.03348	**0.05917**	0.06676	0.06828	0.06962	0.07108	**1.8X**
MNIST ResNet-3, ReLU	0.01751	**0.01827**	0.01864	0.01869	0.01876	0.01881	**1.0X**
CIFAR 5-layer CNN, ReLU	0.00402	**0.00465**	0.00471	0.00472	0.00473	0.00473	**1.2X**
TinyImagenet, 7-layer CNN, ReLU	0.07245	**0.07367**	0.07367	0.07368	0.07369	0.0737	**1.0X**
MNIST 2-layer (robust)-CNN, ReLU	0.09304	**0.11424**	0.12224	0.1238	0.12515	0.12658	**1.2X**
MNIST 2-layer (robust)-CNN, tanh	0.12795	**0.37451**	0.76167	0.90881	1.06778	1.2689	**2.9X**
MNIST 3-layer (robust)-CNN, tanh	0.20596	**0.24122**	0.27452	0.28091	0.28649	0.29239	**1.2X**

(b) With input perturbations being independent random variables in case (i), we perform 100 random trials to randomly choose $\{10, 50, 100\}$ input samples (images) in each trial and then compute the average of the largest ϵ that can be certified by worst-case analysis (Boopathy et al., 2019) (denoted as $\epsilon_{\text{worst-case}}$) and by PROVEN with 99.99% confidence (denoted as ϵ_{PROVEN}) together with the improved certification of ϵ_{PROVEN} over $\epsilon_{\text{worst-case}}$ (denoted as Improv.). The mean and std of the average ϵ and the improvements converges as the number of samples increases. Note that in order to show the numbers converge, here we calculate the index "Improv. = Certification Bound increase-1" and express in %.

		10 samples			50 samples			100 samples		
Models	bound	$\epsilon_{\text{worst-case}}$	ϵ_{PROVEN}	Improv.	$\epsilon_{\text{worst-case}}$	ϵ_{PROVEN}	Improv.	$\epsilon_{\text{worst-case}}$	ϵ_{PROVEN}	Improv.
MNIST 3×[1024], ReLU,ada	Mean	0.02559	0.03703	44.75%	0.02581	0.03734	44.70%	0.02579	0.03733	44.74%
	std	0.00165	0.00222	1.12%	0.00076	0.00102	0.57%	0.00054	0.00071	0.43%
MNIST 3×[1024], tanh	Mean	0.01195	0.01375	15.17%	0.01193	0.01374	15.22%	0.01192	0.01374	15.25%
	std	0.00065	0.00068	2.66%	0.00030	0.00030	1.27%	0.00020	0.00021	0.77%
MNIST 4×[1024], ReLU,ada	Mean	0.00998	0.01329	33.18%	0.00994	0.01325	33.24%	0.00997	0.01328	33.21%
	std	0.00051	0.00066	0.57%	0.00021	0.00027	0.27%	0.00014	0.00018	0.15%
CIFAR 5×[2048], ReLU,ada	Mean	0.00224	0.00264	18.07%	0.00222	0.00262	17.93%	0.00222	0.00263	18.06%
	std	0.00020	0.00025	2.39%	0.00009	0.00011	1.12%	0.00005	0.00006	0.55%
CIFAR 5×[2048], arctan	Mean	0.00091	0.00100	9.28%	0.00091	0.00100	9.32%	0.00092	0.00100	9.32%
	std	0.00008	0.00009	3.17%	0.00003	0.00003	1.15%	0.00001	0.00002	0.56%
CIFAR 7×[1024], ReLU,ada	Mean	0.00176	0.00195	10.68%	0.00174	0.00192	10.73%	0.00174	0.00193	10.70%
	std	0.00018	0.00020	1.87%	0.00007	0.00008	0.75%	0.00003	0.00004	0.37%

Acknowledgement

T.-W. Weng, L. Daniel and I. Oseledets acknowledge partial support ofMIT-Skoltech program. T.-W. Weng and L. Daniel further acknowledge partial support of MIT-IBM Watson AI Lab and MIT-Sensetime program.

References

Bibi, A., Alfadly, M., and Ghanem, B. Analytic expressions for probabilistic moments of pl-dnn with gaussian input. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Biggio, B. and Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *arXiv preprint arXiv:1712.03141*, 2017.

Boopathy, A., Weng, T.-W., Chen, P.-Y., Liu, S., and Daniel, L. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *AAAI*, Jan 2019.

Chen, P.-Y., Vinzamuri, B., and Liu, S. Is ordered weighted ℓ_1 regularized regression robust to adversarial perturbation? a case study on oscar. *arXiv preprint arXiv:1809.08706*, 2018.

Chow, Y. S. and Teicher, H. *Probability Theory: Independence, Interchangeability, Martingales*. Springer, third edition, 2003.

Cisse, M. M., Adi, Y., Neverova, N., and Keshet, J. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *NIPS*, 2017.

Dubhashi, D. P. and Ranjan, D. Balls and bins: A study in negative dependence. *Random Structures and Algorithms*, 13(2):99–124, 1998.

Dvijotham, K., Garnelo, M., Fawzi, A., and Kohli, P. Verification of deep probabilistic models. In *arXiv preprint arXiv:1812.02795*, 2018a.

Dvijotham, K., Stanforth, R., Gowal, S., Mann, T., and Kohli, P. A dual approach to scalable verification of deep networks. *arXiv preprint arXiv:1803.06567*, 2018b.

Ehlers, R. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pp. 269–286. Springer, 2017.

Fawzi, A., Moosavi-Dezfooli, S.-M., and Frossard, P. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pp. 1632–1640, 2016.

Fawzi, A., Fawzi, H., and Fawzi, O. Adversarial vulnerability for any classifier. *arXiv preprint arXiv:1802.08686*, 2018.

Franceschi, J.-Y., Fawzi, A., and Fawzi, O. Robustness of classifiers to uniform ℓ_p and gaussian noise. *arXiv preprint arXiv:1802.07971*, 2018.

Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy (SP)*, volume 00, pp. 948–963, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *ICLR, arXiv preprint arXiv:1412.6572*, 2015.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pp. 2263–2273, 2017.

Hosseini, H., Xiao, B., and Poovendran, R. Google's cloud vision api is not robust to noise. In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*, pp. 101–105. IEEE, 2017.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. *arXiv preprint arXiv:1802.03471*, 2018.

Li, B., Chen, C., Wang, W., and Carin, L. Second-order adversarial attack and certifiable robustness. *arXiv preprint arXiv:1809.03113*, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *ICLR, arXiv preprint arXiv:1706.06083*, 2018.

Panconesi, A. and Srinivasan, A. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

Peck, J., Roels, J., Goossens, B., and Saeys, Y. Lower bounds on the robustness to adversarial perturbations. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 804–813, 2017.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. *ICLR, arXiv preprint arXiv:1801.09344*, 2018.

Resnick, S. I. *A Probability Path*. Birkhäuser, 2014.

Shaked, M. and Shanthikumar, G. *Stochastic Orders*. Springer, 2007.

Sinha, A., Namkoong, H., and Duchi, J. Certifiable distributional robustness with principled adversarial training. *ICLR, arXiv preprint arXiv:1710.10571*, 2018.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Wang, Q., Guo, W., Zhang, K., Ororbia II, A. G., Xing, X., Liu, X., and Giles, C. L. Adversary resistant deep neural networks with an application to malware detection. In *SIGKDD*. ACM, 2017.

Webb, S., Rainforth, T., Teh, Y. W., and Kumar, M. P. A statistical approach to assessing neural network robustness. *ICLR, arXiv preprint arXiv:1811.07209*, 2019.

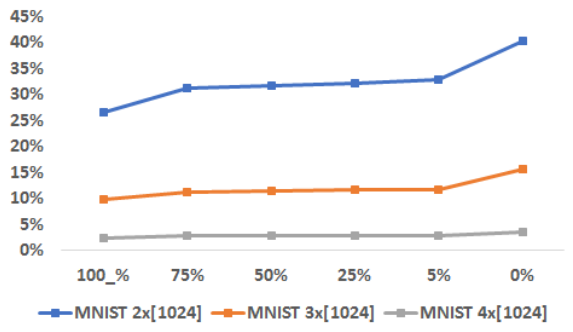
Weng, T.-W., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Boning, D., Dhillon, I. S., and Daniel, L. Towards fast computation of certified robustness for relu networks. *ICML, arXiv preprint arXiv:1804.09699*, 2018.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

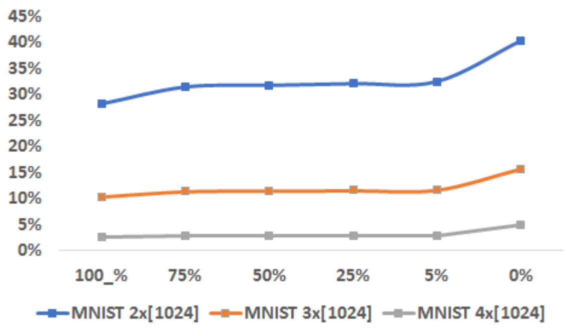
Table 4: **(Full table)** success rates with random attacks using Uniform noises and Bernoulli noises on 100 randomly chosen test images.

Perturbed ℓ_∞ magnitude	$\epsilon = 0.30$		$\epsilon = 0.25$		$\epsilon = 0.20$	
MNIST model	Uniform	Bernoulli	Uniform	Bernoulli	Uniform	Bernoulli
MNIST 2-layer CNN, ReLU	25%	67%	25%	72%	15%	65%
MNIST 2-layer CNN, tanh	35%	59%	91%	99%	83%	98%
MNIST 2-layer CNN, sigmoid	83%	100%	92%	100%	15%	44%
MNIST 2-layer CNN, arctan	18%	58%	7%	44%	22%	22%
MNIST 3-layer CNN, ReLU	72%	89%	69%	90%	53%	99%
MNIST 3-layer CNN, tanh	80%	90%	11%	25%	0%	41%
MNIST 3-layer CNN, sigmoid	7%	31%	14%	24%	30%	76%
MNIST 3-layer CNN, arctan	7%	79%	24%	83%	55%	73%
MNIST 2-layer (robust)-CNN, ReLU	12%	35%	8%	20%	4%	14%
MNIST 2-layer (robust)-CNN, tanh	16%	54%	14%	38%	10%	26%
MNIST 3-layer (robust)-CNN, ReLU	9%	48%	6%	18%	6%	10%
MNIST 3-layer (robust)-CNN, tanh	13%	27%	12%	21%	8%	15%
MNIST LeNet No Pool, ReLU	11%	55%	6%	26%	4%	12%
MNIST ResNet-3, ReLU	98%	98%	98%	98%	98%	100%
Perturbed ℓ_∞ magnitude	$\epsilon = 0.030$		$\epsilon = 0.025$		$\epsilon = 0.020$	
CIFAR model	Uniform	Bernoulli	Uniform	Bernoulli	Uniform	Bernoulli
CIFAR 5×[2048], ReLU	15%	18 %	15%	16%	13%	15%
CIFAR 6×[2048], ReLU	17%	- %	17%	20%	14 %	20 %
CIFAR 5-layer CNN, ReLU	23%	42 %	22 %	31%	17%	28%

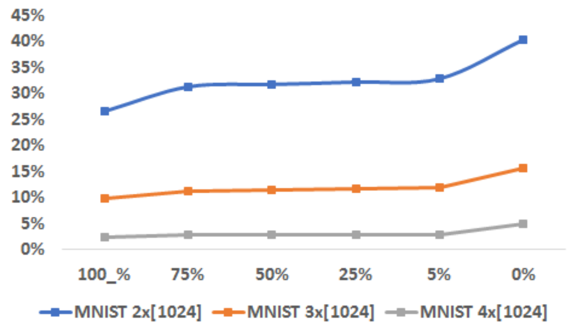
Figure 1: We plot the improvement of the largest ϵ certified by PROVEN with various confidence ($\gamma_L = \{99.99, 75, 50, 25, 5\}\%$) over the largest ϵ certified by worst-case robustness certification algorithms (Weng et al., 2018; Zhang et al., 2018). We consider both input perturbations being independent/correlated Gaussian random variables as in Case (ii) and indedepent random variables as in Case (i). The x-axis label in the figure: γ_L; y-axis label: Certification improvement of PROVEN over $\epsilon_{\text{worst-case}}$. The models are 2-4 layers MNIST networks with 1024 nodes per layer and ReLU actiavations.



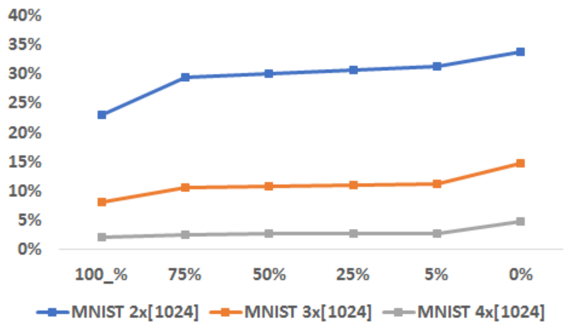
(a) Case (ii) Gaussian i.i.d.



(b) Case (ii) Positive correlated Gaussian



(c) Case (ii) General correlated Gaussian



(d) Case (i) Bounded independent inputs

Table 5: The largest ϵ that PROVEN can certify with confidence of at least $\gamma_L = \{99.99, 75, 50, 25, 5\}\%$ when X_i are independent random variables in Case (i). We compare the largest ϵ that PROVEN can certify with 99.99% with the largest ϵ from state-of-the-art worst-case robustness certification algorithms Fast-Lin (Weng et al., 2018) and show in the last column that PROVEN can certify more than the worst-case analysis by giving up 0.01% confidence. The results comparing PROVEN with CROWN (Zhang et al., 2018) are shown in the main paper in Table 3a with additional 'ada' after network names.

(a) Relu activation

Certification Method	Worst-case (Fast-Lin)	Our probabilistic approach: PROVEN					Certification
Guarantees γ_L	100%[†]	99.99%[†]	75%	50%	25%	5%	improvement[†]
MNIST 2×[20]	0.02722	**0.04394**	0.04782	0.04824	0.04859	0.04897	**61.4%**
MNIST 3×[20]	0.02127	**0.02694**	0.02831	0.02847	0.02860	0.02874	**26.7%**
MNIST 2×[1024]	0.02904	**0.03572**	0.03758	0.03778	0.03796	0.03814	**23.0%**
MNIST 3×[1024]	0.02082	**0.02253**	0.02303	0.02309	0.02313	0.02318	**8.2 %**
MNIST 4×[1024]	0.00796	**0.00813**	0.00817	0.00818	0.00818	0.00818	**2.1 %**
CIFAR 5×[2048]	0.00183	**0.00186**	0.00186	0.00186	0.00186	0.00186	**1.6 %**
CIFAR 7×[1024]	0.00189	**0.00192**	0.00192	0.00193	0.00193	0.00193	**1.6 %**

Table 6: The largest ϵ that PROVEN can certify with confidence of at least $\gamma_L = \{99.99, 75, 50, 25, 5\}\%$ when X_i are independent random variables in Case (i). We compare the largest ϵ that PROVEN can certify with 99.99% with the largest ϵ from state-of-the-art worst-case certification algorithms Fast-Lin and CROWN (Weng et al., 2018; Zhang et al., 2018) and show in the last column that PROVEN can certify more than the worst-case analysis by giving up 0.01% confidence.

(a) Sub-Gaussian noises, bounds

Certification Method	Worst-case	Our probabilistic approach: PROVEN					Certification
Guarantees γ_L	100%[†]	99.99%[†]	75%	50%	25%	5%	Improvement[†]
MNIST 2×[20], ReLU ada	0.02746	**0.04912**	0.05212	0.05246	0.05276	0.05307	**78.9 %**
MNIST 3×[20], ReLU ada	0.02236	**0.03828**	0.03966	0.03981	0.03995	0.04009	**71.2 %**
MNIST 2×[1024], ReLU ada	0.03158	**0.05560**	0.05756	0.05779	0.05798	0.05818	**76.1 %**
MNIST 3×[1024], ReLU ada	0.02397	**0.03524**	0.03583	0.03589	0.03595	0.03601	**47.1 %**
MNIST 4×[1024], ReLU ada	0.00962	**0.01288**	0.01293	0.01294	0.01295	0.01295	**33.9 %**
CIFAR 5×[2048], ReLU ada	0.00228	**0.00264**	0.00265	0.00265	0.00265	0.00265	**15.8 %**
CIFAR 7×[1024], ReLU ada	0.00189	**0.00209**	0.00210	0.00210	0.00210	0.00210	**10.6 %**
MNIST 2×[1024], tanh	0.02232	**0.02915**	0.03005	0.03013	0.03022	0.03033	**30.6%**
MNIST 3×[1024], tanh	0.01121	**0.01360**	0.01376	0.01378	0.01380	0.01381	**21.3 %**
MNIST 4×[1024], tanh	0.00682	**0.00745**	0.00750	0.00750	0.00751	0.00751	**9.2 %**
CIFAR 5×[2048], tanh	0.00081	**0.00085**	0.00085	0.00085	0.00085	0.00085	**4.9 %**
MNIST 2×[1024], sigmoid	0.02785	**0.03285**	0.03404	0.03419	0.03426	0.03441	**18.0%**
MNIST 3×[1024], sigmoid	0.01856	**0.02296**	0.02342	0.02348	0.02353	0.02358	**23.7 %**
MNIST 4×[1024], sigmoid	0.01778	**0.02170**	0.02224	0.02229	0.02232	0.02237	**22.1 %**
MNIST 2×[1024], arctan	0.02105	**0.02796**	0.02907	0.02915	0.02924	0.02936	**32.8%**
MNIST 3×[1024], arctan	0.01250	**0.01462**	0.01486	0.01488	0.01490	0.01493	**17.0 %**
MNIST 4×[1024], arctan	0.00726	**0.00829**	0.00836	0.00837	0.00838	0.00838	**14.2 %**
MNIST 2-layer CNN, ReLU	0.04565	**0.06367**	0.06884	0.06989	0.07082	0.07181	**1.4X**
MNIST 2-layer CNN, tanh	0.0331	**0.09987**	0.13538	0.1437	0.15135	0.15981	**3.0X**
MNIST 2-layer CNN, sigmoid	0.09242	**0.18777**	0.2218	0.22906	0.23553	0.24243	**2.0X**
MNIST 2-layer CNN, arctan	0.03747	**0.13114**	0.18872	0.20279	0.21577	0.23028	**3.5X**
MNIST 3-layer CNN, ReLU	0.04609	**0.06301**	0.0674	0.06828	0.06904	0.06986	**1.4X**
MNIST 3-layer CNN, tanh	0.03348	**0.05917**	0.06676	0.06828	0.06962	0.07108	**1.8X**
MNIST 3-layer CNN, sigmoid	0.07477	**0.13204**	0.14844	0.15186	0.15471	0.15781	**1.8X**
MNIST 3-layer CNN, arctan	0.02868	**0.05514**	0.06272	0.06425	0.06559	0.06702	**1.9X**
MNIST ResNet-3, ReLU	0.01751	**0.01827**	0.01864	0.01869	0.01876	0.01881	**1.0X**
CIFAR 5-layer CNN, ReLU	0.00402	**0.00465**	0.00471	0.00472	0.00473	0.00473	**1.2X**
TinyImagenet, 7-layer CNN, ReLU	0.07245	**0.07367**	0.07367	0.07368	0.07369	0.0737	**1.0X**
MNIST 2-layer (robust)-CNN, ReLU	0.09304	**0.11424**	0.12224	0.1238	0.12515	0.12658	**1.2X**
MNIST 2-layer (robust)-CNN, tanh	0.12795	**0.37451**	0.76167	0.90881	1.06778	1.2689	**2.9X**
MNIST 3-layer (robust)-CNN, ReLU	0.10494	**0.11984**	0.1253	0.12631	0.12717	0.12809	**1.1X**
MNIST 3-layer (robust)-CNN, tanh	0.20596	**0.24122**	0.27452	0.28091	0.28649	0.29239	**1.2X**

Table 7: **Subgaussian noises**: With input perturbations being independent random variables in case (i), we randomly choose $\{10, 50, 100\}$ input samples (images) in each trial and then compute the average of the largest ϵ that can be certified by worst-case framework CNN-Cert (Boopathy et al., 2019) (denoted as $\epsilon_{\text{worst-case}}$) and by PROVEN with 99.99% confidence (denoted as ϵ_{PROVEN}) together with the improved certification of ϵ_{PROVEN} over $\epsilon_{\text{worst-case}}$ (denoted as Improv.). We present the mean and std of the average ϵ and the improvements for $\{10, 50, 100\}$ samples in a total of 100 random trials, showing that the mean and std converge as the number of samples increases.

		10 samples			50 samples			100 samples		
Models	bound	$\epsilon_{\text{worst-case}}$	ϵ_{PROVEN}	Improv.	$\epsilon_{\text{worst-case}}$	ϵ_{PROVEN}	Improv.	$\epsilon_{\text{worst-case}}$	ϵ_{PROVEN}	Improv.
MNIST 3×[1024], ReLU,ada	Mean	0.02559	0.03703	44.75%	0.02581	0.03734	44.70%	0.02579	0.03733	44.74%
	std	0.00165	0.00222	1.12%	0.00076	0.00102	0.57%	0.00054	0.00071	0.43%
MNIST 3×[1024], tanh	Mean	0.01195	0.01375	15.17%	0.01193	0.01374	15.22%	0.01192	0.01374	15.25%
	std	0.00065	0.00068	2.66%	0.00030	0.00030	1.27%	0.00020	0.00021	0.77%
MNIST 4×[1024], ReLU,ada	Mean	0.00998	0.01329	33.18%	0.00994	0.01325	33.24%	0.00997	0.01328	33.21%
	std	0.00051	0.00066	0.57%	0.00021	0.00027	0.27%	0.00014	0.00018	0.15%
CIFAR 5×[2048], ReLU,ada	Mean	0.00224	0.00264	18.07%	0.00222	0.00262	17.93%	0.00222	0.00263	18.06%
	std	0.00020	0.00025	2.39%	0.00009	0.00011	1.12%	0.00005	0.00006	0.55%
CIFAR 5×[2048], arctan	Mean	0.00091	0.00100	9.28%	0.00091	0.00100	9.32%	0.00092	0.00100	9.32%
	std	0.00008	0.00009	3.17%	0.00003	0.00003	1.15%	0.00001	0.00002	0.56%
CIFAR 7×[1024], ReLU,ada	Mean	0.00176	0.00195	10.68%	0.00174	0.00192	10.73%	0.00174	0.00193	10.70%
	std	0.00018	0.00020	1.87%	0.00007	0.00008	0.75%	0.00003	0.00004	0.37%

Table 8: **Gaussian correlated noises**: compare PROVEN with worst-case certification CNN-Cert (Boopathy et al., 2019)

Certification Method	Worst-case	Our probabilistic approach: PROVEN					Certification
Guarantees γ_L	100%[†]	99.99%[†]	75%	50%	25%	5%	Improvement[†]
MNIST 2-layer CNN, ReLU	0.04565	**0.06975**	0.07203	0.07256	0.0731	0.07388	**1.5X**
MNIST 2-layer CNN, tanh	0.0331	**0.14265**	0.1617	0.16626	0.17091	0.17782	**4.3X**
MNIST 2-layer CNN, sigmoid	0.09242	**0.22809**	0.24401	0.24769	0.25141	0.25684	**2.5X**
MNIST 2-layer CNN, arctan	0.03747	**0.20091**	0.23355	0.24136	0.24946	0.2616	**5.4X**
MNIST 3-layer CNN, ReLU	0.04609	**0.06816**	0.07004	0.07046	0.07089	0.07152	**1.5X**
MNIST 3-layer CNN, tanh	0.03348	**0.06809**	0.0714	0.07216	0.07293	0.07405	**2.0X**
MNIST 3-layer CNN, sigmoid	0.07477	**0.15139**	0.15852	0.16012	0.16171	0.16403	**2.0X**
MNIST 3-layer CNN, arctan	0.02868	**0.06406**	0.06734	0.06811	0.06888	0.07	**2.2X**
MNIST ResNet-3, ReLU	0.01751	**0.01868**	0.01883	0.01884	0.01887	0.0189	**1.1X**
CIFAR 5-layer CNN, ReLU	0.00402	**0.00465**	0.00471	0.00472	0.00473	0.00473	**1.2X**
Tiny Imagenet, 7-layer CNN, ReLU	0.07245	**0.07368**	0.0737	0.07371	0.07372	0.07372	**1.0X**
MNIST 2-layer (robust)-CNN, ReLU	0.09304	**0.12361**	0.1269	0.12764	0.12838	0.12946	**1.3X**
MNIST 2-layer (robust)-CNN, tanh	0.12795	**0.88968**	1.3172	1.43648	1.56153	1.74872	**7.0X**
MNIST 3-layer (robust)-CNN, ReLU	0.10494	**0.12618**	0.12829	0.12875	0.12921	0.12989	**1.2X**
MNIST 3-layer (robust)-CNN, tanh	0.20596	**0.28015**	0.29364	0.29681	0.29994	0.30452	**1.4X**

Table 9: **Gaussian iid noises**: compare PROVEN with worst-case certification CNN-Cert (Boopathy et al., 2019)

Certification Method	Worst-case	Our probabilistic approach: PROVEN					Certification
Guarantees γ_L	100%[†]	99.99%[†]	75%	50%	25%	5%	Improvement[†]
MNIST 2-layer CNN, ReLU	0.04565	**0.06975**	0.07204	0.07256	0.0731	0.07388	**1.5X**
MNIST 2-layer CNN, tanh	0.0331	**0.14261**	0.16169	0.16626	0.1709	0.17781	**4.3X**
MNIST 2-layer CNN, sigmoid	0.09242	**0.22811**	0.24399	0.24769	0.25141	0.25682	**2.5X**
MNIST 2-layer CNN, arctan	0.03747	**0.20094**	0.23356	0.24136	0.24949	0.26153	**5.4X**
MNIST 3-layer CNN, ReLU	0.04609	**0.06816**	0.07004	0.07046	0.07089	0.07152	**1.5X**
MNIST 3-layer CNN, tanh	0.03348	**0.06808**	0.07139	0.07216	0.07292	0.07405	**2.0X**
MNIST 3-layer CNN, sigmoid	0.07477	**0.1514**	0.15852	0.16012	0.16171	0.16403	**2.0X**
MNIST 3-layer CNN, arctan	0.02868	**0.06405**	0.06734	0.06811	0.06888	0.07001	**2.2X**
MNIST ResNet-3, ReLU	0.01751	**0.01868**	0.01883	0.01884	0.01887	0.0189	**1.1X**
TinyImageNet, 7-layer CNN, ReLU	0.07245	**0.07368**	0.0737	0.07371	0.07372	0.07372	**1.0X**
MNIST 2-layer (robust)-CNN, ReLU	0.09304	**0.1236**	0.1269	0.12764	0.12838	0.12946	**1.3X**
MNIST 2-layer (robust)-CNN, tanh	0.12795	**0.88787**	1.31724	1.43648	1.56164	1.74802	**6.9X**
MNIST 3-layer (robust)-CNN, ReLU	0.10494	**0.12618**	0.12829	0.12875	0.12921	0.12989	**1.2X**
MNIST 3-layer (robust)-CNN, tanh	0.20596	**0.28014**	0.29365	0.29681	0.29995	0.30454	**1.4X**