Data-Driven Certification of Neural Networks with Random Input Noise

Brendon G. Anderson, *Member, IEEE*, Somayeh Sojoudi, *Member, IEEE*

***Abstract*—Methods to certify the robustness of neural networks in the presence of input uncertainty are vital in safety-critical settings. Most certification methods in the literature are designed for adversarial or worst-case inputs, but researchers have recently shown a need for methods that consider random input noise. In this paper, we examine the setting where inputs are subject to random noise coming from an arbitrary probability distribution. We propose a robustness certification method that lower-bounds the probability that network outputs are safe. This bound is cast as a chance-constrained optimization problem, which is then reformulated using input-output samples to make the optimization constraints tractable. We develop sufficient conditions for the resulting optimization to be convex, as well as on the number of samples needed to make the robustness bound hold with overwhelming probability. Case studies on synthetic, MNIST, and CIFAR-10 networks experimentally demonstrate that this method is able to certify robustness against various input noise regimes over larger uncertainty regions than prior state-of-the-art techniques.**

***Index Terms*—Robustness Certification, Learning, Optimization, Stochastic/Uncertain Systems**

I. Introduction

REAL-WORLD data is inherently uncertain. Such uncertainty comes in a variety of forms, including random measurement noise, adversarial attacks, and even structural perturbations in the underlying graph topology of networked systems [1]–[3]. Despite their excellent performance in a variety of decision and control tasks, e.g., distributed control using graph neural networks [4], researchers have found that neural networks are highly sensitive to uncertainties in their inputs [5]–[7]. This sensitive behavior is intolerable when using neural networks to operate safety-critical control systems, such as the power grid [8]. As a result, a large emphasis has been placed by researchers on the development of methods that certify the robustness properties of neural networks.

Much of the literature on robustness certification has revolved around adversarial inputs, i.e., inputs with small-magnitude perturbations that are designed to cause a worst-case prediction [9]–[12]. However, as argued in [13] and [14], random input uncertainty better models reality in many applications. Areas that commonly use a probabilistic model of uncertainty include stochastic control and finance, where unpredictable measurement errors and state disturbances are assumed to be random [15], [16]. The stochastic framework also naturally encapsulates applications where unbounded uncertainties may exist, albeit with an extremely low probability. This is typical in real-world applications such as aviation [17]. In fact, the International Organization for Standardization (ISO) asserts in their guide on safety aspects that there is never absolute safety, and therefore the goal is to achieve what they define to be *tolerable risk* [18], [19]. Not only are random uncertainties pervasive and realistic, they have been shown to pose a legitimate threat—small uniform noise causes misclassification rates of well over 10% on MNIST and CIFAR-10 networks, and for Bernoulli noise the misclassification rates become drastically worse, sometimes reaching 100% [20].

The aforementioned motivations have led to an influx of recent works considering robustness against random inputs, which we review in Section I-A. Many of them make stringent assumptions on the structure of the network or input distribution, or the formal certification guarantees are relaxed or eliminated in order to enhance computational tractability. Since neural networks are more sensitive to adversarial inputs than to random input noise [1], worst-case sensitivity analyses are too conservative for random input noise when the goal at hand is to achieve a tolerable risk level, whereas high-probability robustness certificates may completely fail in the presence of adversaries; the two settings are disjoint and should be studied using distinct methods. Consequently, our study is intended to certify robustness against random input noise with minimal conservatism, and is not intended to assess adversarial robustness.

A. Related Works

In this section, we review the state-of-the-art methods for assessing robustness to random inputs, highlight their usages, and address their limitations. For instance, [14] defines robustness as the network output being Lipschitz continuous with high probability when two inputs are chosen randomly. Their proposed method is limited to neural networks composed of conditional affine transformations, e.g., ReLU networks. On the other hand, [20] analytically bounds the probability that a classifier's margin function exceeds a given value. Although this probabilistic method applies to general neural network models, it assumes that the random input noise is constrained to an ℓ_p-norm ball and is either Gaussian or has independent coordinates. Furthermore, their bounding technique relies on worst-case analysis methods, making their resulting certificates relatively loose (see Section VI).

B. G. Anderson is with the Department of Mechanical Engineering, University of California, Berkeley (email: bganderson@berkeley.edu).

S. Sojoudi is with the Department of Electrical Engineering and Computers Sciences and the Department of Mechanical Engineering, University of California, Berkeley (email: sojoudi@berkeley.edu).

In [13], robustness is measured by the probability that random input noise results in misclassification. The authors propose a sampling-based approximation of the robustness level. However, no theoretical guarantees are given to certify the network's robustness. Contrarily, [21] formally bounds the probability that a random input maps to an unsafe output. However, the bounding function is nonconvex, and therefore obtaining tight bounds amounts to nonconvex optimization. Alternatively, [1] bounds the size of a random input perturbation that causes a classifier prediction error with high probability. Their bounds provide elegant theoretical guarantees, but they depend on the network's worst-case robustness level, which is generally NP-hard to compute without approximation errors or additional assumptions [10], [22].

The works [23] and [24] provide methods to guarantee that network outputs do not significantly deviate from the nominal output when the input is subject to random uncertainty. This corresponds to the problem of localizing the network outputs within the output space. The work [23] uses the output localization to issue high-probability guarantees for the network's robustness. However, their method requires solving a semidefinite program, and their results are demonstrated on small, single-layer networks, so it is not clear whether their method scales to realistic applications. The concentration bounds presented in [24] can be applied to deep networks, but their localization results do not immediately translate into a meaningful certificate of robustness.

The authors of [25] and [26] use input-output samples to learn how input noise is propagated to the output space through a method called scenario optimization. This approach naturally embeds the stochastic nature of the input noise into the assessment procedure. The work [25] provides a method to estimate a network's set of possible outputs, but this localization may fail to determine the safety of the output since the underlying optimization does not directly consider any safety specifications, e.g., classification boundaries. We demonstrate this phenomenon in Appendix I-A. Other scenario-based output set estimation techniques for general nonlinear maps, such as [27]–[29], suffer from the same limitation in the context of neural network certification. Contrarily, [26] directly considers output safety in their scenario approach. However, their method makes use of an affine approximation to the network's nonlinear margin function, a worst-case analysis technique from the adversarial robustness literature. In our experiments, we show that this worst-case technique yields loose robustness bounds as the size of the input noise increases.

Finally, [17] also uses sampled data to bound the probability of failure. Their guarantees take the probably approximately correct (PAC) form in terms of probability levels $\epsilon, \delta \in [0,1]$, and they improve the sample complexity from $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ of the naive Chernoff bound to $O(\frac{1}{\epsilon}\log\frac{1}{\delta})$. This is achieved by imposing a Bayesian framework and assuming that the failure probability follows a uniform prior distribution. As the authors remark, this is a "very conservative choice." In contrast, the method to be proposed in this paper solves for the optimal (least conservative) robustness bound of this form with the same $O(\frac{1}{\epsilon}\log\frac{1}{\delta})$ sample complexity.

B. Contributions

In this paper, we develop a data-driven framework for certifying neural network robustness against random input noise using scenario optimization. Our direct approach avoids worst-case analysis techniques, such as those found in the adversarial robustness literature, e.g., [20], and also avoids the need for selecting a Bayesian prior distribution governing the network's failure probability, as in [17]. We show in Section III that a general class of sets, termed ϵ-covers, suffice for localizing and certifying network outputs. In Section IV, we develop sufficient conditions on this class to ensure that the procedure amounts to a convex optimization problem, and we develop formal guarantees that the resulting robustness certificate holds with overwhelming probability upon using sufficiently many samples in the scenario optimization. These results draw on the areas of output set estimation, parametric and set-valued optimization, and scenario optimization to provide a unified data-driven assessment procedure that is novel and state-of-the-art in the robustness certification literature. Although the method is applicable to all neural networks and all input noise distributions, we show in Section V how to exploit the structure of networks with affinely bounded activation functions in order to reduce sample complexity.

Our numerical experiments demonstrate that the proposed optimization is capable of issuing robustness certificates in cases where the two-step process of optimally localizing the outputs (e.g., using [25]) and then certifying them cannot, providing a novel perspective that output set estimation techniques do not necessarily work well for certification. Furthermore, we show on both synthetic networks and large MNIST and CIFAR-10 networks that our robustness bounds are much tighter than those obtained by the state-of-the-art method [20], particularly for large noise levels.

Due to space restrictions, proofs, supplementary experiments, and additional technical details are moved to the technical report [30].

C. Notations

The ceiling of $x \in \mathbb{R}$ is written $\lceil x \rceil$. For $x, y \in \mathbb{R}^n$, we define $[x,y] = \{z \in \mathbb{R}^n : x \leq z \leq y\}$, where the inequalities are interpreted element-wise. Given a set $\mathcal{X}$, we denote its power set by $\mathcal{P}(\mathcal{X})$. The Minkowski sum of sets $\mathcal{X}$ and $\mathcal{Y}$ is defined as $\mathcal{X} + \mathcal{Y} = \{x + y : x \in \mathcal{X},\ y \in \mathcal{Y}\}$. We define $\mathbb{R}_{++} = \{x \in \mathbb{R} : x > 0\}$. For a function $f \colon \mathbb{R}^m \to \mathbb{R}^n$, we write the image of $\mathcal{X} \subseteq \mathbb{R}^m$ under f as $f(\mathcal{X}) = \{f(x) \in \mathbb{R}^n : x \in \mathcal{X}\}$. If $g \colon \mathbb{R}^n \to \mathbb{R}^p$ is another function, we define the composition $g \circ f \colon \mathbb{R}^m \to \mathbb{R}^p$ by $g \circ f(x) = g(f(x))$. If $X \colon \Omega \to \mathbb{R}^n$ is a random variable on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, $g \colon \mathbb{R}^n \to \mathbb{R}$ is a Borel measurable function, and $c \in \mathbb{R}$, we use the notation $\mathbb{P}(g(X) \geq c)$ to mean $\mathbb{P}(\{\omega \in \Omega : g(X(\omega)) \geq c\})$. Similarly, if $\mathcal{S} \subseteq \mathbb{R}^m$ is a Borel set and $h \colon \mathbb{R}^n \to \mathbb{R}^m$ is a Borel measurable function, we write $\mathbb{P}(h(X) \in \mathcal{S})$ to mean $\mathbb{P}(\{\omega \in \Omega : h(X(\omega)) \in \mathcal{S}\})$. For a norm $\|\cdot\|$ on $\mathbb{R}^n$ we denote its dual norm by $\|\cdot\|_*$, where $\|y\|_* = \sup\{x^\top y : \|x\| \leq 1\}$. We assume throughout that optimization problems are attained by a solution.

II. Problem Statement

A. Network Description, Safe Set, and Safety Level

In this paper, we consider a Borel measurable neural network $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ with arbitrary structure and parameters.[1] We assume that the input to the network is a random variable $X\colon \Omega_X \to \mathbb{R}^{n_x}$ on a fixed probability space $(\Omega_X, \mathcal{F}_X, \mathbb{P}_X)$.[2] We do not assume that the distribution $\mathbb{P}_X$ is exactly known—we only assume that we are able to sample from $\mathbb{P}_X$. The support of the probability measure $\mathbb{P}_X$ is called the *input set*, which is denoted by $\mathcal{X} \subseteq \mathbb{R}^{n_x}$. The *output set* of the network is defined to be $\mathcal{Y} = f(\mathcal{X}) \subseteq \mathbb{R}^{n_y}$.

Next, consider a given convex polyhedral *safe set* $\mathcal{S} = \{y \in \mathbb{R}^{n_y} : Ay + b \geq 0\}$, where $A \in \mathbb{R}^{n_s \times n_y}$ and $b \in \mathbb{R}^{n_s}$. Without loss of generality, we assume that $n_s = 1$, henceforth setting $A = a^\top \in \mathbb{R}^{1\times n_y}$ and $b \in \mathbb{R}$.[3] The results of this paper can be immediately generalized to the case where $n_s > 1$. See the technical report [30] for a detailed explanation.

The elements of the set $\mathcal{S}$ are considered to be *safe*. For a point y in the output space $\mathbb{R}^{n_y}$, the value $s(y) = a^\top y + b$ is called the *safety level* of y. The point y is safe if and only if its safety level is nonnegative. Broadly speaking, the overall goal of this paper is to certify that the random output $Y = f(X)$ is safe. When this holds for all or many of the possible outputs in $\mathcal{Y}$, we obtain a natural certificate for the robustness of the network against the random noise.

Example 1. When f is an n_y-class classifier and $\bar{x} \in \mathbb{R}^{n_x}$ is a deterministic nominal input with class $i^* \in \arg\max_{i\in\{1,2,\dots,n_y\}} f_i(\bar{x})$, a common goal is to certify that additive random noise δ on $\bar{x}$ does not cause misclassification [20]. This problem falls within our framework by defining the safety level of $f(X)$ to be the *margin function* value $g_i(X) := f_{i^*}(X) - f_i(X)$ for $X = \bar{x} + \delta$.

B. Various Notions of Robustness

We now use the safety level of outputs to introduce three meaningful notions of robustness against random input noise, and discuss how they are related to one another.

1) Deterministic Robustness Level: The *deterministic robustness level* of the network is defined as

$$r^* = \inf_{y\in\mathcal{Y}} a^\top y + b. \tag{1}$$

If $r^* \geq 0$, then $\mathcal{Y} \subseteq \mathcal{S}$, implying that the random output $Y = f(X)$ is safe with probability one. This notion of robustness coincides with that used when considering adversarial inputs [9], [11], [12], but the resulting worst-case safety level is often much lower than the safety levels of random outputs in practice [13]. Consequently, using r^* to assess robustness may falsely indicate that the network is sensitive to the input noise.

2) Approximate Robustness Level: Although r^* can issue strong guarantees about the safety of the network output, (1) amounts to an intractable nonconvex optimization problem, since $\mathcal{Y}$ is generally a nonconvex set. Instead of computing r^*, we can consider approximating it by

$$\hat{r}(\hat{\mathcal{Y}}) = \inf_{y\in\hat{\mathcal{Y}}} a^\top y + b, \tag{2}$$

where $\hat{\mathcal{Y}} \subseteq \mathbb{R}^{n_y}$, termed the *surrogate output set*, is more tractable than $\mathcal{Y}$, and preferably convex. We call (2) the *approximate robustness level* of the network. If $\mathcal{Y} \subseteq \hat{\mathcal{Y}}$, then $\hat{r}(\hat{\mathcal{Y}}) \leq r^*$. In this case, if $\hat{r}(\hat{\mathcal{Y}}) \geq 0$, then the random output $Y = f(X)$ is safe with probability one. In general, choosing $\hat{\mathcal{Y}}$ to cover $\mathcal{Y}$ makes $\hat{r}(\hat{\mathcal{Y}})$ an over-conservative measure of robustness for the same reasons r^* is.

3) Probabilistic Robustness Level: The notion of deterministic robustness is too strong for applications involving random input noise, as many input distributions have unbounded support or have their worst-case inputs in regions of low probability measure [13]. Furthermore, (1) and (2) neglect the distributional information given for X. This conservatism means that the robustness levels (1) and (2), with $\mathcal{Y} \subseteq \hat{\mathcal{Y}}$, are generally unable to certify that Y is safe, even when Y concentrates around safe outputs. Consequently, for a prescribed probability level $\epsilon \in [0,1]$, we define the more natural *probabilistic robustness level* of the network to be

$$\bar{r}(\epsilon) = \sup\{r \in \mathbb{R} : \mathbb{P}_X(a^\top f(X) + b \geq r) \geq 1-\epsilon\}. \tag{3}$$

Intuitively, the random output $f(X)$ has a safety level at least $\bar{r}(\epsilon)$ with high probability. In the case that $\bar{r}(\epsilon) \geq 0$, we certify that the random output $Y = f(X)$ is safe with probability $1-\epsilon$, and we say that the network is *probabilistically robust*.[4] Another interpretation of probabilistic robustness is that the majority of possible outputs (with respect to the distribution $\mathbb{P}_X$) are safe. The probabilistic robustness level is catered towards our setting of random input noise, and, compared to the worst-case alternatives, reduces conservatism by considering the actual likelihood of the possible inputs. This is precisely the notion of robustness we adopt in this paper. We remark that this definition of probabilistic robustness coincides with that used in [13] and [20], albeit our subsequent analysis and guarantees vastly differ from these works.

Example 2. Consider again the classification network in Example 1. In this setting, if $r^* \geq 0$, then the probability of misclassification is zero, in which case the noise δ is not important. On the other hand, if there exists a perturbed input $\bar{x} + \delta$ in the input set $\mathcal{X}$ that is misclassified, then $r^* < 0$. It may be the case, however, that this misclassification occurs with a sufficiently low probability with respect to the tolerance ϵ. This is precisely what we seek to certify:

[1]Borel measurability of the network is almost always satisfied in practice. Indeed, every continuous function is Borel measurable.

[2]It is easily verified that all probabilistic expressions in this paper are well-defined from a measure-theoretic perspective. We leave out these measurability verifications for the sake of exposition.

[3]The polyhedral safe set assumption is without loss of generality. Suppose that the safe set is $\mathcal{S} = \{y \in \mathbb{R}^{n_y} : a^\top g(y) + b \geq 0\}$ for some nonlinear Borel measurable $g\colon \mathbb{R}^{n_y} \to \mathbb{R}^{n_z}$ and some $a \in \mathbb{R}^{n_z}, b \in \mathbb{R}$. Then we can reduce the problem to our assumed form by considering $f' := g \circ f$ to be the (Borel measurable) neural network and $\mathcal{S}' := \{z \in \mathbb{R}^{n_z} : a^\top z + b \geq 0\}$ to be the (polyhedral) safe set, since then $f(x) \in \mathcal{S}$ if and only if $f'(x) \in \mathcal{S}'$, for all $x \in \mathcal{X}$. The architecture-dependent results of Section V must be applied to f' with care, since g must now be considered as another layer in the network, making Assumptions 1 and 2 to follow slightly more stringent.

[4]Note the distinction: "safety" is a property of outputs, whereas "robustness" is a property of the neural network (with respect to the noisy input distribution $\mathbb{P}_X$). We are always careful when using these terminologies.

that the probability of misclassification is less than ϵ, which mathematically amounts to showing that $\bar{r}(\epsilon) \geq 0$.

In this paper, we aim to certify the safety of $Y = f(X)$ by lower-bounding the probabilistic robustness level $\bar{r}(\epsilon)$. A trivial lower bound is easily verified: $r^* \leq \bar{r}(\epsilon)$ for all $\epsilon \in [0,1]$, and $r^* = \bar{r}(0)$. However, as we will see in Section VI, this approach of considering *worst-case* inputs instead of *likely* inputs, often results in a very loose lower bound, sometimes failing to issue a robustness guarantee at all. In the next section, we show that by using a special type of surrogate output set in $\hat{r}(\hat{\mathcal{Y}})$, we can optimize a lower bound on $\bar{r}(\epsilon)$ and obtain an estimate of the output set $\mathcal{Y}$ as a natural byproduct.

III. Formulating the Certificate

A. Bounding the Probabilistic Robustness Level

As we have seen, $\hat{r}(\hat{\mathcal{Y}}) \approx r^* \leq \bar{r}(\epsilon)$. Two natural questions arise. 1) Can one use $\hat{r}(\hat{\mathcal{Y}})$ to certifiably lower-bound the quantity $\bar{r}(\epsilon)$ of interest? 2) If so, how can $\hat{\mathcal{Y}}$ be chosen to optimize the bound? In this section, we study the first question. As it turns out, such a lower bound holds so long as $\hat{\mathcal{Y}}$ has high enough coverage over $\mathcal{Y}$. Before proving this claim, we formally define this notion of coverage.

Definition 1. Let $\hat{\mathcal{Y}}$ be a subset of $\mathbb{R}^{n_y}$. For $\epsilon \in [0,1]$, the set $\hat{\mathcal{Y}}$ is said to be an *ϵ-cover* of $\mathcal{Y} = f(\mathcal{X})$ if $\hat{\mathcal{Y}}$ is Borel measurable and $\mathbb{P}_X(f(X) \in \hat{\mathcal{Y}}) \geq 1 - \epsilon$.

Intuitively, an ϵ-cover of $\mathcal{Y}$ is a set that contains $Y = f(X)$ with high probability. If we can compute an ϵ-cover of $\mathcal{Y}$, then we will have localized the output with high confidence. By restricting $\hat{\mathcal{Y}}$ in (2) to be an ϵ-cover of $\mathcal{Y}$, we ensure that the approximate robustness level takes into account the *likely* inputs X, but not necessarily the worst-case inputs. Consequently, this permits $\hat{r}(\hat{\mathcal{Y}})$ to be greater than r^*, reducing the conservatism in our measure of robustness caused by unlikely worst-case inputs. We now show that this special type of surrogate output set is a good enough estimate of $\mathcal{Y}$ to maintain the lower bound on $\bar{r}(\epsilon)$ that we seek.

Proposition 1. *Let $\hat{\mathcal{Y}}$ be an arbitrary subset of $\mathbb{R}^{n_y}$. If $\hat{\mathcal{Y}}$ is an ϵ-cover of $\mathcal{Y} = f(\mathcal{X})$, then*

$$\hat{r}(\hat{\mathcal{Y}}) \leq \bar{r}(\epsilon). \tag{4}$$

B. Optimizing the Bound

From Proposition 1, we know that ϵ-covers constitute good choices of the surrogate output set $\hat{\mathcal{Y}}$ used to compute the approximate robustness level. This is because the random output $Y = f(X)$ is guaranteed to have safety level at least $\hat{r}(\hat{\mathcal{Y}})$ with high probability. However, it is entirely possible that the choice of ϵ-cover results in $\hat{r}(\hat{\mathcal{Y}}) < 0$, even when the network is probabilistically robust. In this case, $\hat{r}(\hat{\mathcal{Y}})$ fails to issue a high-probability certificate for the safety of the random output $Y = f(X)$, despite $\hat{\mathcal{Y}}$ being able to localize it.

To overcome the above problem, we turn to studying our second inquiry from earlier, namely, how to optimize the lower bound (4). This amounts to finding an ϵ-cover of $\mathcal{Y}$ that maximizes $\hat{r}(\hat{\mathcal{Y}})$. Since optimizing over all subsets of $\mathbb{R}^{n_y}$ is intractable, we restrict our search to sets within a class $\mathcal{H} = \{h(\theta) : \theta \in \Theta\}$ parameterized by a parameter set $\Theta \subseteq \mathbb{R}^p$ and a set-valued function $h\colon \mathbb{R}^p \to \mathcal{P}(\mathbb{R}^{n_y})$. We assume throughout that the class $\mathcal{H}$ is chosen such that $h(\theta)$ is a Borel set for all parameters $\theta \in \Theta$. A concrete example of one such class is given below.

Example 3. Let $\|\cdot\|$ be a fixed norm on $\mathbb{R}^{n_y}$ and $\Theta = \mathbb{R}^{n_y} \times \mathbb{R}_{++}$. Defining $p = n_y + 1$, let $h\colon \mathbb{R}^p \to \mathcal{P}(\mathbb{R}^{n_y})$ be defined by $h(\bar{y}, r) = \{y \in \mathbb{R}^{n_y} : \|y - \bar{y}\| \leq r\}$. Then, Θ and $h(\cdot)$ define the class of $\|\cdot\|$-norm balls:

$$\mathcal{H} = \big\{\{y \in \mathbb{R}^{n_y} : \|y - \bar{y}\| \leq r\} : r > 0,\ \bar{y} \in \mathbb{R}^{n_y}\big\}.$$

The problem of choosing $h(\cdot)$ and Θ (and therefore also $\mathcal{H}$) is discussed in detail in Section IV. By restricting our search for ϵ-covers to within the class $\mathcal{H}$, our search reduces to maximizing the approximate robustness level over the parameter set Θ. By slightly abusing notation, we denote the dependence of the approximate robustness level on the parameter θ explicitly as $\hat{r}(\theta) = \inf\{a^\top y + b : y \in h(\theta)\}$, and we formulate the following optimization problem:

$$\begin{aligned} \underset{\theta\in\Theta}{\text{maximize}} \quad & \hat{r}(\theta) - \lambda v(\theta) \\ \text{subject to} \quad & \mathbb{P}_X(f(X) \in h(\theta)) \geq 1 - \epsilon, \end{aligned} \tag{5}$$

where $\lambda \geq 0$ and $v\colon \mathbb{R}^p \to \mathbb{R}$ is taken to be a nonnegative convex function on Θ that increases with the volume of $h(\theta)$. The objective $\hat{r}(\theta)$ in (5) is the approximate robustness level computed using the set $h(\theta)$ as the surrogate output set. The constraint $\mathbb{P}_X(f(X) \in h(\theta)) \geq 1 - \epsilon$ enforces that we only consider parameters θ such that $h(\theta)$ is an ϵ-cover of $\mathcal{Y}$. The regularization term $-\lambda v(\theta)$ penalizes the size of $h(\theta)$. This makes the set $h(\theta)$ as small as possible while maintaining its ϵ-coverage, thereby yielding the tightest localization of the output Y. The regularization is done at the expense of a slightly suboptimal bound (4), and can be eliminated by setting $\lambda = 0$, if no localization of the output Y is desired. On the other hand, increasing λ amounts to putting more assessment effort into localizing Y, making the ϵ-cover $h(\theta)$ a better estimate of $\mathcal{Y}$. This certification-localization tradeoff is experimentally explored in Section VI-A.

IV. Data-Driven Reformulation

Even when the set $h(\theta)$ is convex for all $\theta \in \Theta$, the probabilistic constraint in (5) is in general nonconvex [31]. Constraints of this form are referred to as *chance constraints*, and there exist various approaches to reformulating and relaxing them into convex constraints. Since the problem at hand considers neural networks whose models are usually complicated to analyze, but whose input-output samples are easily obtained, we seek a data-driven approach to approximately enforcing the chance constraint in (5), without losing the robustness certificate provided by the solution. The *scenario approach* is a popular method within the stochastic optimization and robust control communities that replaces the chance constraint with hard constraints on a number of random samples [31]–[34]. The scenario approach has been studied for general problems, even those with nonconvex objectives and those whose resulting hard constraints are nonconvex [35]. However, the most powerful use of scenario optimization arises when the resultant scenario problem is convex, as then *a priori* probabilistic guarantees can be made about the solution's feasibility for the original chance constraint. As we will soon see, this sampling-based method fits nicely into the framework of our problem, and maintains a lower bound on $\bar{r}(\epsilon)$ with high probability, provided that a sufficiently large number of samples is used and the scenario problem is convex.

To implement the scenario approach, suppose that $\{x_j : j \in \{1,2,\dots,N\}\} \subseteq \mathcal{X}$ is a set of N independent and identically distributed samples drawn from $\mathbb{P}_X$. For each input x_j, we compute its corresponding output $y_j = f(x_j)$. Then, replacing the chance constraint in (5) with N hard constraints on the samples y_j yields the following scenario optimization:

$$\begin{aligned} \underset{\theta\in\Theta}{\text{maximize}} \quad & \hat{r}(\theta) - \lambda v(\theta) \\ \text{subject to} \quad & y_j \in h(\theta) \text{ for all } j \in \{1,2,\dots,N\}. \end{aligned} \tag{6}$$

Note that, because the data y_j is random, solutions θ^* to (6) are random. We assume throughout the paper that (6) is attained by a solution θ^*, and we denote the probability space on which it is defined by $(\Omega_{\theta^*}, \mathcal{F}_{\theta^*}, \mathbb{P}_{\theta^*})$.

Remark 1. The above assumption of independent and identically distributed samples is critical for relating the solution θ^* back to the original chance-constrained problem (5). In particular, it is a key assumption on which the forthcoming high-probability robustness certificate in Theorem 2 rests. Despite these assumptions holding in many practical models, the independence may be violated in certain applications with inherent time-correlation between samples, and the assumption prevents the use of selective sampling to improve the efficiency of the scenario approach.

The identical distribution assumption is also critical, and it may be violated in two main ways. First, the underlying distribution of the data used in (6) may change from sample to sample, and second, the underlying noise distribution of the actual input may be different in practice from the samples used in the robustness certification procedure. Despite these sources of modeling error, our scenario-based approach can be modified into a distributionally robust variant to still give high-probability robustness certificates in the case that the distribution of the input is contained in a finite set of possible distributions.

As mentioned in Section I-A, the scenario approach was used recently in reachable set estimation for dynamical systems [25]. We remark that (6) recovers the scenario optimization of [25] in the special case that the objective is re-scaled to $\frac{1}{\lambda}\hat{r}(\theta) - v(\theta)$ and $\lambda \to \infty$, the regularizer $v(\theta)$ equals the volume of the set $h(\theta)$, and $\mathcal{H}$ is the norm ball class. This reduction amounts to finding the tightest norm ball ϵ-cover of $\mathcal{Y}$, without regard to optimizing the lower bound (4) of interest. In Appendix I-A, we demonstrate the necessity for the more general formulation (6) by giving an example where reducing to the special case of [25] causes robustness certification to fail, despite finding the tightest ϵ-cover of $\mathcal{Y}$.

Although the scenario approach has eliminated the chance constraint from (5), there remain two problems to consider. First, it is not immediately clear whether (6) is convex or computationally tractable, as it has an inherent max-min optimization structure. However, it is important to ensure the problem's convexity, since no *a priori* guarantees can be made regarding the feasibility of θ^* for the original chance constraint in the general case of nonconvex scenario optimization [35]. In Section IV-A, we leverage results from parametric optimization to develop conditions on our choice of Θ and $h(\cdot)$ to ensure that (6) is convex. Second, the solution of (6) gives a random approximation to the solution of (5), which optimizes the bound (4) on $\bar{r}(\epsilon)$. In Section IV-B we develop formal guarantees showing that the solution of (6) maintains a lower bound on $\bar{r}(\epsilon)$ with high probability, provided that N is sufficiently large.

A. Conditions for Convex Optimization

In this section, we consider the choices of the parameter set Θ and the set-valued function $h(\cdot)$ on lower-bounding $\bar{r}(\epsilon)$, and on the tractability of the resulting scenario problem (6). A key insight is this: an ϵ-cover of $\mathcal{Y}$ may in general be much larger than $\mathcal{Y}$ itself. This is because regions of an ϵ-cover that do not intersect with $\mathcal{Y}$ also do not count towards the coverage proportion $1-\epsilon$. Therefore, if the class $\mathcal{H}$ from which we choose an ϵ-cover does not have high enough complexity, then the ϵ-covers within $\mathcal{H}$ may need to be exceedingly large in order to achieve ϵ-coverage.

The problem with unnecessarily large ϵ-covers is that the feasible set in the optimization defining $\hat{r}(\theta)$ includes many vectors y that may not be actual outputs in $\mathcal{Y}$. In this case, $\hat{r}(\theta)$ is small, even though $\bar{r}(\epsilon)$ may be large. To avoid this problem, our choice of Θ and $h(\cdot)$ should ensure that the class $\mathcal{H}$ has high enough complexity. However, our choices should also yield a scenario problem (6) that is convex. Indeed, Theorem 1 gives sufficient conditions for the convexity of the scenario problem. Before presenting these conditions, let us recall a fundamental definition for set-valued functions.

Definition 2. A set-valued function $h\colon \mathbb{R}^p \to \mathcal{P}(\mathbb{R}^{n_y})$ is said to be *convex* on a convex set $\Theta \subseteq \mathbb{R}^p$ if

$$\big(\lambda h(\theta_1) + (1-\lambda)h(\theta_2)\big) \subseteq h(\lambda\theta_1 + (1-\lambda)\theta_2)$$

for all $\theta_1, \theta_2 \in \Theta$ and all $\lambda \in [0,1]$. The function $h(\cdot)$ is said to be *concave* on Θ if

$$h(\lambda\theta_1 + (1-\lambda)\theta_2) \subseteq \big(\lambda h(\theta_1) + (1-\lambda)h(\theta_2)\big)$$

for all $\theta_1, \theta_2 \in \Theta$ and all $\lambda \in [0,1]$. Finally, the function $h(\cdot)$ on Θ is said to be *affine* if it is both convex and concave.

Example 4. Consider the norm ball class $\mathcal{H}$ given in Example 3. It is easily verified by Definition 2 that the set-valued function $h(\cdot)$ defining the class $\mathcal{H}$ is affine on $\Theta = \mathbb{R}^{n_y} \times \mathbb{R}_{++}$.

With tools for defining and proving convexity of set-valued functions now in place, we can present conditions under which the scenario optimization (6) is convex, and therefore easily solvable. In Theorem 2, we will also rely on this convexity to guarantee with high probability that $h(\theta^*)$ is an ϵ-cover of $\mathcal{Y}$ and that our desired lower bound $\hat{r}(\theta^*) \le \bar{r}(\epsilon)$ holds. Generating such guarantees is in general not possible for nonconvex scenario optimization [35], further illustrating the importance of Theorem 1 below.

Theorem 1. *Consider the scenario problem* (6). *Suppose that* Θ *takes the form*

$$\Theta = \{\theta \in \mathbb{R}^p : g_i(\theta) \leq 0 \text{ for all } i \in \{1, 2, \dots, m\}\},$$

where every $g_i\colon \mathbb{R}^p \to \mathbb{R}$ *is convex. Furthermore, suppose that* $h(\cdot)$ *is a concave set-valued function that takes the form*

$$h(\theta) = \{y \in \mathbb{R}^{n_y} : h_i(y, \theta) \leq 0 \text{ for all } i \in \{1, 2, \dots, n\}\},$$

where $h_i\colon \mathbb{R}^{n_y} \times \mathbb{R}^p \to \mathbb{R}$ *and* $h_i(y, \cdot)$ *is convex for all* $y \in \mathbb{R}^{n_y}$. *Then,* (6) *is a convex optimization problem.*

Theorem 1 precisely answers our earlier inquiry: the class $\mathcal{H}$ should be complex enough to contain tight ϵ-covers of the output set $\mathcal{Y}$, but at the same time Θ should be defined by convex constraints and $h(\cdot)$ should be taken as a concave set-valued function also defined by convex constraints. Note that these conditions on $h(\cdot)$ are not as restrictive as they may seem. In particular, Example 4 shows for the norm ball class that $h(\cdot)$ is affine (and therefore concave) and defined by convex constraints, and that this holds *for all norms on* $\mathbb{R}^{n_y}$, even though norm functions themselves are not affine. Therefore, Theorem 1 guarantees that the scenario optimization (6) using the norm ball class is a convex problem. We verify this fact in the following example.

Example 5. Recall the norm ball class of Examples 3 and 4. We show that (6) using this class is convex. Indeed, the approximate robustness level is

$$\hat{r}(\bar{y}, r) = \inf_{\|y-\bar{y}\|\leq r} a^\top y + b = a^\top \bar{y} - r\|a\|_* + b,$$

which is affine in the optimization variable $\theta = (\bar{y}, r)$. Hence, the scenario problem reduces to

$$\begin{aligned} \underset{(\bar{y},r)\in\mathbb{R}^{n_y}\times\mathbb{R}_{++}}{\text{maximize}} \quad & b + a^\top \bar{y} - r\|a\|_* - \lambda v(\bar{y}, r) \\ \text{subject to} \quad & \|y_j - \bar{y}\| \leq r \text{ for all } j \in \{1, 2, \dots, N\}, \end{aligned} \tag{7}$$

which is a convex problem since $v(\cdot)$ is convex.

B. High-Probability Guarantees

We now turn to consider the randomness of the scenario problem's optimal value. In particular, we ask the following question: Does the random solution to (6) maintain a certified lower bound on $\bar{r}(\epsilon)$? In Theorem 2, we show that the answer is affirmative with high probability, provided that the problem is convex and a large enough number of samples is used.

Theorem 2. *Let* $\epsilon, \delta \in [0, 1]$. *Assume that the scenario optimization* (6) *is convex and is attained by a solution* $\theta^* \in \mathbb{R}^p$. *If* $N \geq \frac{2}{\epsilon}\left(\log\frac{1}{\delta} + p\right)$, *then the following inequalities hold:*

1) $\mathbb{P}_{\theta^*}(\mathbb{P}_X(f(X) \in h(\theta^*)) \geq 1 - \epsilon) \geq 1 - \delta$;
2) $\mathbb{P}_{\theta^*}(\hat{r}(\theta^*) \leq \bar{r}(\epsilon)) \geq 1 - \delta$.

The conclusions of Theorem 2 assert that, with overwhelming probability, $h(\theta^*)$ is an ϵ-cover of $\mathcal{Y}$ and that the probabilistic robustness level is lower-bounded as $\hat{r}(\theta^*) \leq \bar{r}(\epsilon)$. This gives high-probability guarantees for the simultaneous localization and safety certification of the output $Y = f(X)$.

In Theorem 2, randomness of a solution θ^* to (6) is taken care of by the $1-\delta$ probability bound. In particular, $h(\theta^*)$ may not actually be an ϵ-cover, albeit with probability at most δ. For this reason, we slightly abuse terminology and call $h(\theta^*)$ the optimal ϵ-cover. The additional layer of uncertainty embedded into the parameter δ is precisely the price paid for replacing the intractable chance-constrained problem (5) with the tractable scenario problem (6). However, Theorem 2 shows that the additional randomness is not an issue, since the requirement on N scales as $\log\frac{1}{\delta}$. Therefore, we can select a small value for δ while maintaining a reasonable sample size N.

Remark 2. The guarantees in Theorem 2 are of the probably approximately correct (PAC) form. In the language of PAC learning, the surrogate output set $h(\theta^*)$ is the hypothesis of the learner, which is selected from the concept class $\mathcal{H} = \{h(\theta) : \theta \in \Theta\}$. Theorem 2 asserts that the hypothesis is probably approximately correct, where *approximately correct* means the hypothesis (which is a set) contains the random output $Y = f(X)$ with probability at least $1 - \epsilon$, and where *probably* means the hypothesis (which is selected based on the specific instances $x_1, x_2, \dots, x_N$) is approximately correct (for general X) with probability at least $1 - \delta$. Since this PAC guarantee holds whenever the scenario problem is convex, Theorem 1 gives sufficient conditions for the concept class $\mathcal{H}$ to be PAC learnable, and our proposed method can be viewed as learning robustness using the framework of PAC learning.

V. Exploiting Network Structure

In this section, we show how to exploit the structure of deep neural networks to reduce the time complexity of our method. The basic idea is to utilize adversarial bounds on the deep layers to replace f with a shallower neural network, in effect developing a hybrid adversarial-probabilistic certification scheme. We assume that the network takes the form

$$f = \sigma^{(K)} \circ \mathcal{A}^{(K-1)} \cdots \circ \sigma^{(1)} \circ \mathcal{A}^{(0)},$$

where $\sigma^{(k)}\colon \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ is the kth layer's activation function and $\mathcal{A}^{(k)}\colon \mathbb{R}^{n_k} \to \mathbb{R}^{n_{k+1}}$ is the affine map given by $\mathcal{A}^{(k)}(z) = W^{(k)}z + b^{(k)}$. Note that $n_0 = n_x$ and $n_K = n_y$.

Now, suppose that $f_L, f_U\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ are two functions satisfying $f_L(x) \leq f(x) \leq f_U(x)$ for all $x \in \mathcal{X}$, which are to be determined. Then, define the function $f'\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ by

$$f'_i(x) = \begin{cases} (f_L(x))_i & \text{if } a_i \geq 0, \\ (f_U(x))_i & \text{if } a_i < 0, \end{cases}$$

for all $i \in \{1, 2, \dots, n_y\}$ and all $x \in \mathbb{R}^{n_x}$. It is immediately clear that $a^\top f'(x) + b \leq a^\top f(x) + b$ for all $x \in \mathcal{X}$, so $f(x) \in \mathcal{S}$ for all $x \in \mathcal{X}$ such that $f'(x) \in \mathcal{S}$. This shows that $\mathbb{P}_X(f'(X) \in \mathcal{S}) \leq \mathbb{P}_X(f(X) \in \mathcal{S})$. Therefore, to certify the probabilistic robustness of f, it suffices to apply our certification procedure to the function f'. By bounding the deep layers' activations in f by affine functions, we will reduce the problem to analyzing a simpler and shallower network f' that allows for faster sampling of the outputs y_j. For notational simplicity, we let $\phi^{(k)} = \sigma^{(k)} \circ \mathcal{A}^{(k-1)} \circ \cdots \circ \sigma^{(1)} \circ \mathcal{A}^{(0)}$ for all

$k \in \{1,2,\dots,K\}$, so that $\phi^{(k)}(x)$ is the activation at layer k corresponding to the input x. Let $\phi^{(0)}$ be the identity map on $\mathbb{R}^{n_x}$. We now recall the notion of preactivation bounds, and make two assumptions.

Definition 3. A vector $l^{(k)} \in \mathbb{R}^{n_k}$ satisfying $l^{(k)} \leq \mathcal{A}^{(k-1)} \circ \phi^{(k-1)}(x)$ for all $x \in \mathcal{X}$ is called a *k^{th} layer preactivation lower bound.* A vector $u^{(k)} \in \mathbb{R}^{n_k}$ satisfying $\mathcal{A}^{(k-1)} \circ \phi^{(k-1)}(x) \leq u^{(k)}$ for all $x \in \mathcal{X}$ is called a *k^{th} layer preactivation upper bound.*

Assumption 1. For all $k \in \{1,2,\dots,K\}$, there exist k^{th} layer preactivation lower and upper bounds $l^{(k)}$ and $u^{(k)}$, respectively.

Assumption 2. For all $k \in \{1,2,\dots,K\}$, there exist functions $\mathcal{L}^{(k)},\mathcal{U}^{(k)} \colon \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ given by $\mathcal{L}^{(k)}(z) = W_L^{(k)} z + b_L^{(k)}$ and $\mathcal{U}^{(k)}(z) = W_U^{(k)} z + b_U^{(k)}$, that satisfy $\mathcal{L}^{(k)}(z) \leq \sigma^{(k)}(z) \leq \mathcal{U}^{(k)}(z)$ for all $z \in [l^{(k)}, u^{(k)}]$.

Definition 3 and Assumptions 1 and 2 are standard in the adversarial robustness literature. Notice that in many common architectures, $n_1 > n_0$ and the rank of $\mathcal{A}^{(0)}$ is n_0, and in this case Assumption 1 requires the input set $\mathcal{X}$ to be bounded. For most common activation functions and input sets, there exist a variety of methods for computing the above preactivation bounds and affine bounding functions—see, e.g., [36].

The following lemma transforms our affine bounds on each activation function $\sigma^{(k)}$ into affine bounds relating the activation of one layer to the activation of the next layer.

Lemma 1. *Suppose that Assumptions 1 and 2 hold. For all* $k \in \{1,2,\dots,K\}$, *it holds for all* $x \in \mathcal{X}$ *that* $\tilde{W}_L^{(k)} \phi^{(k-1)}(x) + \tilde{b}_L^{(k)} \leq \phi^{(k)}(x) \leq \tilde{W}_U^{(k)} \phi^{(k-1)}(x) + \tilde{b}_U^{(k)}$, *where*

$$\begin{aligned} \tilde{W}_L^{(k)} &= W_L^{(k)} W^{(k-1)}, \quad \tilde{b}_L^{(k)} = W_L^{(k)} b^{(k-1)} + b_L^{(k)}, \\ \tilde{W}_U^{(k)} &= W_U^{(k)} W^{(k-1)}, \quad \tilde{b}_U^{(k)} = W_U^{(k)} b^{(k-1)} + b_U^{(k)}. \end{aligned} \tag{8}$$

Next, we use the affine bounds between each neighboring layer in Lemma 1 to develop one overall affine bound relating the activation at some layer k^* to the output $\phi^{(K)}(x)$ of the neural network. The proof technique follows the idea developed in [10], [36], albeit allows for more general activation functions and allows us to "start" the affine bounding within the interior of the neural network architecture.

Proposition 2. *Suppose that Assumptions 1 and 2 hold, and assume that* $K \geq 3$. *Let* $k^* \in \{1,2,\dots,K-2\}$ *and define* $M = K - k^*$. *Consider the matrices* $\tilde{W}_L^{(k)}, \tilde{W}_U^{(k)}$ *and vectors* $\tilde{b}_L^{(k)}, \tilde{b}_U^{(k)}$ *defined in* (8). *Define* $E_1 = \tilde{W}_L^{(k^*+1)}$, $F_1 = \tilde{b}_L^{(k^*+1)}$, $G_1 = \tilde{W}_U^{(k^*+1)}$, *and* $H_1 = \tilde{b}_U^{(k^*+1)}$. *Also, for* $n \in \{2,3,\dots,M\}$, *define*

$$\begin{aligned} E_n &= \min\{0, \tilde{W}_L^{(k^*+n)}\} G_{n-1} + \max\{0, \tilde{W}_L^{(k^*+n)}\} E_{n-1}, \\ F_n &= \min\{0, \tilde{W}_L^{(k^*+n)}\} H_{n-1} \\ &\quad + \max\{0, \tilde{W}_L^{(k^*+n)}\} F_{n-1} + \tilde{b}_L^{(k^*+n)}, \\ G_n &= \max\{0, \tilde{W}_U^{(k^*+n)}\} G_{n-1} + \min\{0, \tilde{W}_U^{(k^*+n)}\} E_{n-1}, \\ H_n &= \max\{0, \tilde{W}_U^{(k^*+n)}\} H_{n-1} \\ &\quad + \min\{0, \tilde{W}_U^{(k^*+n)}\} F_{n-1} + \tilde{b}_U^{(k^*+n)}. \end{aligned}$$

Then, for all $x \in \mathcal{X}$, *it holds that*

$$E_M \phi^{(k^*)}(x) + F_M \leq \phi^{(K)}(x) \leq G_M \phi^{(k^*)}(x) + H_M.$$

Since $\phi^{(K)}(x) = f(x)$, Proposition 2 shows that we may take the functions f_L, f_U to be $f_L = \mathcal{A}_L \circ \phi^{(k^*)}$ and $f_U = \mathcal{A}_U \circ \phi^{(k^*)}$, where $\mathcal{A}_L(z) = E_M z + F_M$ and $\mathcal{A}_U(z) = G_M z + H_M$. In this case, our function f' becomes

$$f_i'(x) = \begin{cases} \left(\mathcal{A}_L \circ \phi^{(k^*)}(x)\right)_i & \text{if } a_i \geq 0, \\ \left(\mathcal{A}_U \circ \phi^{(k^*)}(x)\right)_i & \text{if } a_i < 0. \end{cases}$$

This function f' is a new neural network with the same first $k^* < K$ nonlinear layers as f, and with one final affine transformation. Thus, a lower bound on the probabilistic robustness level of this shallow surrogate network f' is also a lower bound on the probabilistic robustness level of the deep original network f.

When k^* is chosen to be small, the depth of this surrogate network is reduced, making it more efficient to sample outputs from it. As k^* increases, our method incorporates more of the underlying nonlinear nature of the network f into the samples that we use to assess f's robustness, meaning that the robustness certificate becomes tighter, but at the expense of increased sampling time. Specifically, in the common setting where every activation $\sigma^{(k)}$ is an element-wise operator with the time complexity $O(n_k)$, the time complexity of the sampling procedure for f is $O(N(n_0 n_1 + n_1 n_2 + \cdots + n_{K-1} n_K))$, whereas the time complexity for f' is $O(N(n_0 n_1 + n_1 n_2 + \cdots + n_{k^*-1} n_{k^*} + n_{k^*} n_K))$. If, for example, every number n_k is of order $O(n)$, then f would have the sampling time complexity $O(NKn^2)$, whereas f' would be of order $O(Nk^* n^2)$, giving a factor of k^*/K reduction in time complexity. As we will see in Appendix I-B, this reduced time complexity is particularly helpful in deep neural network settings.

VI. Numerical Experiments

A. Illustrative Example

We consider the distributed linear system $x(t+1) = Ax(t) + Bu(t)$ for times $t \in \{0,1,\dots,T\}$, $T = 20$, as constructed in [4]. The system has $n = 10$ nodes, with a single state and input associated with every node; $x(t), u(t) \in \mathbb{R}^n$. The system and control matrices A, B respect the underlying graph topology of the system, encoded by the support matrix S—see [4].

The control law is defined by a graph neural network: $u(t) = \Phi(x(t), S) \coloneqq \sum_{k=0}^{K-1} h_{k+1}^{(2)} S^k \sigma\left(\sum_{j=0}^{J-1} h_{j+1}^{(1)} S^j x(t)\right)$, with $\sigma(\cdot) = \mathrm{ReLU}(\cdot)$, $K = J = 3$, $h^{(1)} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, and $h^{(2)} = (1,1,1)$. This neural network controller, defined in terms of S, respects the distributed nature of the system [4]. In this experiment, we consider the case where the graph support of the control law may be randomly perturbed, so that $u(t) = \Phi(x(t), S')$ for some $S' \in \mathbb{R}^{n \times n}$ with $S'_{ij} = X S_{ij}$, where X is a Bernoulli random variable equal to 1 with probability 0.8; the controller loses an edge in its support graph with probability 0.2. We fix a (normal random) initial condition $x(0) \in \mathbb{R}^n$, and we consider the map $f \colon \mathbb{R}^{n \times n} \to \mathbb{R}^2$ given by $f(S') = (x_1(T), x_2(T))$, where $x(T)$ is the terminal state of the system under the control law given by $u(t) = \Phi(x(t), S')$.

The safe set is defined by $\mathcal{S}_1 = \{y \in \mathbb{R}^2 : a^\top y + b \geq 0\}$, where $a = (1, 0)$ and $b = 0.05$. We seek to certify that the first two elements of the (random) terminal state are safe even under the perturbed control support S', i.e., that $f(S') \in \mathcal{S}_1$.

The norm ball class $\mathcal{H}$ of Examples 3, 4, and 5 is employed with $\|\cdot\|$ being the ℓ_2-norm, and with probability levels $\epsilon = 0.05$ and $\delta = 10^{-5}$. We choose the regularizer for the scenario problem (7) to be the square of the norm ball radius, i.e., $v(\bar{y}, r) = r^2$. The optimization problem is convex as guaranteed by Theorem 1. We solve the scenario problem first without regularization, and then with two different levels of regularization: $\lambda_1 = 1$ and $\lambda_2 = 100$. The respective solutions are denoted by θ^*, $\theta^*_{\lambda_1}$, and $\theta^*_{\lambda_2}$. Each instance takes approximately 15 seconds to solve using CVX in MATLAB on a standard laptop with a 2.6 GHz dual-core i5 processor. The resulting approximate robustness levels are $\hat{r}(\theta^*) = 0.0058$, $\hat{r}(\theta^*_{\lambda_1}) = 0.0054$, and $\hat{r}(\theta^*_{\lambda_2}) = -0.0061$. In the instances without regularization and with regularization level λ_1, Theorem 2 guarantees that the perturbed terminal state $(x_1(T), x_2(T))$ has a safety level of 0.005 with our prescribed high probability, granting the probablistic robustness certificate we seek. On the other hand, since $\hat{r}(\theta^*_{\lambda_2}) < 0$, the scenario problem using regularization level λ_2 is not able to certify the safety of the terminal state. This is due to the inherent tradeoff between localization and certification, which we now discuss.

The optimal ϵ-covers $h(\theta^*)$, $h(\theta^*_{\lambda_1})$, and $h(\theta^*_{\lambda_2})$ are shown in Figure 1, which is placed in Appendix I due to space constraints. The unregularized set $h(\theta^*)$ is massively over-conservative due to the choice $\lambda = 0$, which corresponds to pure robustness certification. Indeed, $h(\theta^*)$ is the ϵ-cover from our class of sets that is furthest from the boundary of the safe set, making $\hat{r}(\theta^*)$ the tightest lower bound on $\bar{r}(\epsilon)$. On the other hand, the optimal ϵ-covers using $\lambda = \lambda_1$ and $\lambda = \lambda_2$ are seen to give tighter localizations of the terminal state $(x_1(T), x_2(T))$. The approximate robustness level using regularization λ_1 is only slightly lower than the unregularized value, but the regularization λ_2 is large enough to cause the approximate robustness level $\hat{r}(\theta^*_{\lambda_2})$ to become negative at the expense of localization. This shows how overemphasizing localization may harm the certification aspect of robustness assessment, and empirically demonstrates why output set estimation methods may not be adequate for issuing robustness certificates. This is explored further in Appendix I-A.

We repeat the experiment with the more complicated safe set $\mathcal{S}_2 = \{y \in \mathbb{R}^2 : Ay + b \geq 0\}$, where $A = \left[\begin{smallmatrix} 1 & 0 \\ -1 & 0 \end{smallmatrix}\right]$ and $b = (0.05, 0)$, applying our method to each row of $\mathcal{S}_2$ individually. To do so, we set $\epsilon' = \epsilon/2$, $\delta' = \delta/2$, and $N' = \lceil \frac{2}{\epsilon'} \left(\log \frac{1}{\delta'} + p\right) \rceil = 1217$. For each of the two half-spaces defining $\mathcal{S}_2$, we solve the scenario problem using N' independent and identically distributed samples, and then intersect the two resulting ϵ'-covers. Doing so, we obtain an ϵ-cover with probability at least $1 - \delta$. We repeat this process again using regularization levels $\lambda_1 = 1$ and $\lambda_2 = 100$, and we find that each scenario problem takes approximately 30 seconds to solve. As seen in Figure 2 in Appendix I, some samples may reside outside the resulting intersection ϵ-covers—this is valid, and the robustness certificates still hold.

Again, we find robustness certificates for $\lambda = 0$ and $\lambda = \lambda_1$. However, for $\lambda = \lambda_2$, the optimal ϵ-covers corresponding to both half-spaces are found to intersect the unsafe region of the state space, due to the increased emphasis on localization. Interestingly, the overall localization after intersecting the two ϵ'-covers for $\lambda = \lambda_2$ is in a sense looser than that of the case $\lambda = \lambda_1$, indicating that moderate regularization levels, like λ_1 in this experiment, may simultaneously perform best for both localization and certification in the case of safe sets defined by more than one half-space. Optimizing λ in general poses an interesting problem for future research.

B. Comparison to PROVEN

In this experiment, we compare our approach using the half-space class $\mathcal{H} = \left\{\{y \in \mathbb{R}^{n_y} : c^\top y + d \geq 0\} : (c, d) \in \mathbb{R}^{n_y} \times \mathbb{R}\right\}$, for which we solve the scenario problem using its closed-form solution (see our technical report [30]), to the state-of-the-art algorithm, PROVEN [20], for assessing robustness against random input noise. Throughout, we use open-source neural network models provided in [20]. The underlying framework of PROVEN relies on bounding a classifier's margin function by affine functions. PROVEN uses the affine functions to give closed-form bounds on the misclassification probability. We remark that, since PROVEN does not rely on sampling, their lower bound on $\bar{r}(\epsilon)$ is deterministic, whereas our bound holds with probability $1 - \delta$, which is taken to be $1 - 10^{-5} = 0.99999$ in this experiment. The results in this section are computed using TensorFlow in Python on a standard laptop with a 2.6 GHz dual-core i5 processor.

We first consider a variety of pre-trained MNIST digit classification networks with ReLU activation functions [37]. A network model with m hidden layers, each having n neurons, is denoted by $m \times [n]$. We model the noisy input X as being distributed uniformly on $\mathcal{X} = \{x \in \mathbb{R}^{n_x} : \|x - \bar{x}\|_\infty \leq \epsilon_x\}$. For 10 randomly selected nominal inputs $\bar{x}$, we compute a lower bound $\hat{r}(\theta^*)$ on the probabilistic robustness level $\bar{r}(\epsilon)$. The robustness level of a network (for a particular pair (ϵ, ϵ_x)) is evaluated by computing the average robustness level lower bound across the 10 inputs.[5] This is done for probability levels $\epsilon \in \{0.001, 0.1, 0.25\}$ (with corresponding sample sizes $N \in \{25026, 251, 101\}$) and for a variety of noise levels ϵ_x. We include the certified adversarial radius computed using [36], which is a lower bound on the smallest radius such that $\mathcal{X}$ contains an input that yields an unsafe output. The targeted class i, which defines the margin function g_i relative to the nominal input's true class i^*, is randomly chosen for each input tested. See Examples 1 and 2 for more information on this application. The average lower bound values computed using our approach (denoted Ours) and PROVEN's (denoted PRVN) are shown in Table I, which is placed in Appendix I due to space constraints.

[5]Despite $\bar{r}(\epsilon)$ being an input-specific quantity, we follow the literature's standard practice and average our robustness metric over a collection of test inputs. This standard was popularized in [5], where model robustness is evaluated using average certified input set radii. Our average robustness level lower bound immediately gives an average certified input set radius when the bound is nonnegative. In the probabilistic setting, it can be more natural to evaluate models in terms of misclassification probability, like our bounds do, instead of in terms of certified input set radii, see, e.g., [13], [17], [23], [24].

As seen in Table I, our method is able to certify larger input sets than PROVEN for every network tested. Although PROVEN's lower bound is tighter for small radii, at large radii our bound is significantly tighter than PROVEN's, particularly for the larger networks in Tables Ic and Id. This indicates that our method is especially powerful for certifying deep neural networks. The end-to-end affine bounding scheme in PROVEN tends to become looser as the network becomes deeper and as the input set becomes larger [20]. The technique comes from the adversarial robustness literature, and therefore it being embedded into PROVEN is likely the reason why PROVEN fails for radii larger than the certified adversarial radius. Our method bypasses this preliminary bound altogether. We also remark that our method certifies much larger input set radii (sometimes up to 20 times larger) compared to the certified adversarial radii (italicized) computed using the state-of-the-art worst-case analysis [36]. The exact minimum adversarial radii (averaged across the 10 inputs) for the $2\times[20]$ and $3\times[20]$ ReLU networks are efficiently computed to be around 0.07 using mixed-integer linear programming [38]. With the tolerance $\epsilon = 0.001$, our method certifies radii over 0.1 for these networks. This evidences the claim that worst-case approaches, including exact ones, are over-conservative when applied to settings where a small amount of risk may be tolerable, in effect justifying our data-driven framework. We repeat the experiment using the sample-based certification method of [17], which we recall is only able to issue binary certificates for whether or not $\bar{r}(\epsilon)$ is lower bounded by 0. Our method gives much less conservative lower bounds—see Table II in Appendix I for the $2\times[20]$ ReLU network and our technical report [30] for the other models.

In Table III of Appendix I, we repeat the experiment using three variants in the neural network model. Model 1 is an MNIST classifier with $\tanh(\cdot)$ activation functions of size $4\times[1024]$. On the other hand, Model 2 is a CIFAR-10 network with ReLU activations of size $5\times[2048]$. We see that both Model 1 and Model 2 exhibit the same behavior as before; for small input set radii, the lower bounds provided by PROVEN and our method are similar and both yield high-probability robustness certificates. For larger radii, our lower bound significantly outperforms PROVEN's. Since the affine bounds in PROVEN are relatively tight for small input sets radii, we suspect the PROVEN bound to closer match our bound for large input set radii in the special case of linear classification networks.

Model 3 is a linear classifier, i.e., of the form $f(x)=Wx+b$, with 50 inputs, 10 outputs, and weights, biases, and nominal input all chosen randomly with elements uniform on $[0,1]$. We computed lower bounds on $\bar{r}(\epsilon)$ for 100 such models and averaged the results. Table III shows that indeed the PROVEN bound closely matches our bound for every radius tested in this special case, and that the two methods succeed and fail to issue robustness certificates simultaneously. These results show that the worst-case bounding techniques used in the adversarial robustness literature may work satisfactorily for simple models with random inputs, such as linear classifiers, but that these bounds are too loose for general nonlinear networks.

VII. Conclusions

In this paper, we propose a data-driven method for certifying the robustness of neural networks against random input noise. Sufficient conditions are developed for the convexity of the resulting optimization, as well as on the number of samples to issue a high-probability guarantee for the safety of the output. The method applies to general neural networks and general input noise distributions. In cases where the activation functions can be affinely bounded, we show how to exploit the network structure to reduce sample complexity. The unified framework allows the user to balance the strength of the robustness bound with the tightness of the resulting output set estimate. Our numerical experiments show that the proposed method gives less conservative robustness bounds than the prior state-of-the-art techniques, as it is capable of certifying larger input uncertainty regions on synthetic, MNIST, and CIFAR-10 networks. In situations where neural network failure modes may exist but are unlikely and hence robustness amounts to achieving tolerable risk, these results suggest that re-tooling worst-case analysis techniques from the adversarial robustness literature results in overly conservative bounds. We conclude that taking a data-driven approach to generate probabilistic robustness guarantees, as developed in this paper, is the better option in these contexts.

Acknowledgment

This work was supported by grants from AFOSR, ONR, and NSF. The authors would like to thank Lily Weng for their insightful discussions on the implementation of PROVEN.

References

[1] J.-Y. Franceschi, A. Fawzi, and O. Fawzi, "Robustness of classifiers to uniform ℓ_p and Gaussian noise," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, 2018, pp. 1280–1288.

[2] M. Jin, J. Lavaei, S. Sojoudi, and R. Baldick, "Boundary defense against cyber threat for power system state estimation," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1752–1767, 2020.

[3] D. Z. Simon Geisler and S. Günnemann, "Reliable graph neural networks via robust aggregation," in *Advances in Neural Information Processing Systems*, 2020.

[4] F. Gama and S. Sojoudi, "Graph neural networks for distributed linear-quadratic control," in *Learning for Dynamics and Control*. PMLR, 2021, pp. 111–124.

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

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

[7] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[8] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2017.

[9] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5286–5295.

[10] L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for ReLU networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5276–5285.

[11] A. Raghunathan, J. Steinhardt, and P. S. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 877–10 887.

[12] B. G. Anderson, Z. Ma, J. Li, and S. Sojoudi, "Tightened convex relaxations for neural network robustness certification," in *Proceedings of the 59th IEEE Conference on Decision and Control*, 2020.

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

[14] R. Mangal, A. V. Nori, and A. Orso, "Robustness of neural networks: A probabilistic and practical approach," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2019, pp. 93–96.

[15] K. J. Åström, *Introduction to Stochastic Control Theory*. Courier Corporation, 2012.

[16] H. Föllmer and A. Schied, *Stochastic Finance*. de Gruyter, 2016.

[17] R. R. Zakrzewski, "Randomized approach to verification of neural networks," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, vol. 4. IEEE, 2004, pp. 2819–2824.

[18] C. Huang, Z. Hu, X. Huang, and K. Pei, "Statistical certification of acceptable robustness for neural networks," in *International Conference on Artificial Neural Networks*. Springer, 2021, pp. 79–90.

[19] ISO/IEC, "Guide 51: Safety aspects—guidelines for their inclusion in standards," 1999.

[20] L. Weng, P.-Y. Chen, L. Nguyen, M. Squillante, A. Boopathy, I. Oseledets, and L. Daniel, "PROVEN: Verifying robustness of neural networks with a probabilistic approach," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6727–6736.

[21] K. Dvijotham, M. Garnelo, A. Fawzi, and P. Kohli, "Verification of deep probabilistic models," in *Advances in Neural Information Processing Systems, SecML workshop*, 2018.

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

[23] M. Fazlyab, M. Morari, and G. J. Pappas, "Probabilistic verification and reachability analysis of neural networks via semidefinite programming," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 2726–2731.

[24] N. Couellan, "Probabilistic robustness estimates for feed-forward neural networks," *Neural Networks*, vol. 142, pp. 138–147, 2021.

[25] A. Devonport and M. Arcak, "Estimating reachable sets with scenario optimization," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 75–84.

[26] R. Li, P. Yang, C.-C. Huang, B. Xue, and L. Zhang, "Probabilistic robustness analysis for DNNs based on PAC learning," *arXiv preprint arXiv:2101.10102*, 2021.

[27] Y. Yang, J. Zhang, K.-q. Cai, and M. Prandini, "A stochastic reachability analysis approach to aircraft conflict detection and resolution," in *2014 IEEE Conference on Control Applications (CCA)*. IEEE, 2014, pp. 2089–2094.

[28] M. L. Fravolini, T. Yucelen, A. Ficola, and M. R. Napolitano, "Probabilistic estimation of the reachable set of model reference adaptive controllers using the scenario approach," *International Journal of Control*, vol. 90, no. 2, pp. 307–321, 2017.

[29] H. Sartipizadeh, A. P. Vinod, B. Açikmeşe, and M. Oishi, "Voronoi partition-based scenario reduction for fast sampling-based stochastic reachability computation of linear systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 37–44.

[30] B. G. Anderson and S. Sojoudi, "Data-driven certification of neural networks with random input noise," 2022. [Online]. Available: https://brendon-anderson.github.io/files/publications/anderson2022data-long.pdf

[31] A. Nemirovski and A. Shapiro, "Convex approximations of chance constrained programs," *SIAM Journal on Optimization*, vol. 17, no. 4, pp. 969–996, 2007.

[32] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Springer Science & Business Media, 2012.

[33] M. C. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design," *Annual Reviews in Control*, vol. 33, no. 2, pp. 149–157, 2009.

[34] J. Luedtke and S. Ahmed, "A sample approximation approach for optimization with probabilistic constraints," *SIAM Journal on Optimization*, vol. 19, no. 2, pp. 674–699, 2008.

[35] M. C. Campi, S. Garatti, and F. A. Ramponi, "A general scenario theory for nonconvex optimization and decision making," *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4067–4078, 2018.

[36] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in Neural Information Processing Systems*, 2018, pp. 4939–4948.

[37] Y. LeCun, "The MNIST database of handwritten digits," *http://yann.lecun. com/exdb/mnist/*, 1998.

[38] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *International Conference on Learning Representations*, 2019.

Brendon G. Anderson is a Ph.D. student in the Department of Mechanical Engineering at the University of California, Berkeley, where he obtained his M.S. in 2020. He obtained his B.S. in Mechnical Engineering at the University of California, Los Angeles in 2018. His research interests revolve around optimization, machine learning, and control theory.

Somayeh Sojoudi is an Assistant Professor in the Departments of Electrical Engineering & Computer Sciences and Mechanical Engineering at the University of California, Berkeley. She works on interdisciplinary problems in optimization theory, machine learning, and control theory. She is a recipient of multiple awards, including NSF CAREER Award, and ONR Young Investigator Award.

Appendix I
Figures, Tables, and Additional Experiments

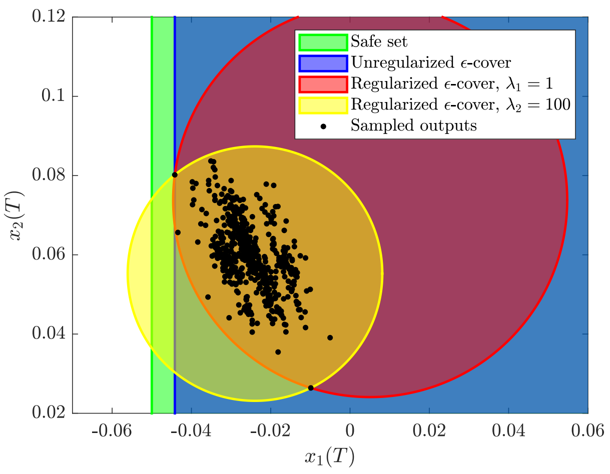

Fig. 1: Optimal ℓ_2-norm ball ϵ-covers for safe set $\mathcal{S}_1$.

TABLE I: Average probabilistic robustness level lower bounds $\hat{r}(\theta^*)$ for MNIST ReLU networks subject to uniform noise over ℓ_∞-norm ball. All values are averaged over 10 nominal inputs with randomly chosen target classes i. Lower bounds giving certified robustness (on average) are bolded, and the average certified adversarial radii computed using [36] are italicized.

(a) $2 \times [20]$ network.

Radius	$\epsilon = 0.001$		$\epsilon = 0.1$		$\epsilon = 0.25$	
	PRVN	Ours	PRVN	Ours	PRVN	Ours
0.01	**24.51**	**14.11**	**24.79**	**14.26**	**24.88**	**14.28**
	1.491 s	0.605 s	1.405 s	0.004 s	1.370 s	0.003 s
0.027	**14.71**	**13.36**	**15.45**	**13.77**	**15.68**	**13.85**
	1.434 s	0.599 s	1.540 s	0.004 s	1.427 s	0.003 s
0.05	−1.33	**12.34**	**0.02**	**13.11**	**0.44**	**13.26**
	1.511 s	0.597 s	1.468 s	0.004 s	1.423 s	0.003 s
0.1	−42.09	**10.25**	−39.43	**11.66**	−38.61	**12.00**
	1.485 s	0.623 s	1.437 s	0.004 s	1.437 s	0.002 s
0.5	−404.42	−7.21	−391.05	−0.05	−386.96	**1.71**
	1.525 s	0.645 s	1.472 s	0.004 s	1.432 s	0.002 s

(b) $3 \times [20]$ network.

Radius	$\epsilon = 0.001$		$\epsilon = 0.1$		$\epsilon = 0.25$	
	PRVN	Ours	PRVN	Ours	PRVN	Ours
0.01	**29.24**	**17.28**	**29.59**	**17.45**	**29.70**	**17.49**
	1.416 s	0.380 s	1.345 s	0.003 s	1.388 s	0.001 s
0.022	**18.80**	**16.65**	**19.49**	**17.02**	**19.71**	**17.10**
	1.382 s	0.362 s	1.345 s	0.003 s	1.364 s	0.001 s
0.05	−22.31	**15.19**	−20.67	**16.00**	−20.17	**16.19**
	1.377 s	0.345 s	1.325 s	0.003 s	1.374 s	0.001 s
0.1	−114.83	**12.57**	−111.59	**14.19**	−110.60	**14.58**
	1.351 s	0.372 s	1.343 s	0.003 s	1.340 s	0.002 s
0.5	−866.28	−9.36	−857.82	−0.55	−855.22	**0.36**
	1.385 s	0.368 s	1.351 s	0.003 s	1.336 s	0.003 s

(c) $2 \times [1024]$ network.

Radius	$\epsilon = 0.001$		$\epsilon = 0.1$		$\epsilon = 0.25$	
	PRVN	Ours	PRVN	Ours	PRVN	Ours
0.01	**51.07**	**27.73**	**51.40**	**27.87**	**51.50**	**27.93**
	0.899 s	1.102 s	0.877 s	0.008 s	0.874 s	0.004 s
0.032	**30.34**	**26.69**	**31.37**	**27.13**	**31.69**	**27.32**
	0.869 s	1.202 s	0.858 s	0.008 s	0.846 s	0.004 s
0.05	**6.95**	**25.83**	**8.59**	**26.53**	**9.09**	**26.82**
	0.846 s	1.125 s	0.851 s	0.008 s	0.844 s	0.004 s
0.1	−77.53	**23.46**	−74.13	**24.84**	−73.09	**25.42**
	0.861 s	1.137 s	0.854 s	0.008 s	0.881 s	0.004 s
0.5	−914.36	**4.83**	−900.22	**11.79**	−895.89	**14.45**
	0.869 s	1.159 s	0.883 s	0.008 s	0.868 s	0.004 s

(d) $3 \times [1024]$ network.

Radius	$\epsilon = 0.001$		$\epsilon = 0.1$		$\epsilon = 0.25$	
	PRVN	Ours	PRVN	Ours	PRVN	Ours
0.01	**68.87**	**36.86**	**69.28**	**37.06**	**69.41**	**37.12**
	2.535 s	1.782 s	2.382 s	0.015 s	2.464 s	0.009 s
0.024	**44.14**	**35.97**	**45.18**	**36.44**	**45.50**	**36.58**
	2.434 s	2.026 s	2.448 s	0.013 s	2.510 s	0.008 s
0.05	−111.09	**34.32**	−108.25	**35.32**	−107.39	**35.59**
	2.739 s	2.258 s	2.671 s	0.013 s	2.761 s	0.007 s
0.1	−729.24	**31.10**	−723.45	**33.10**	−721.68	**33.69**
	3.081 s	2.325 s	2.916 s	0.014 s	2.912 s	0.007 s
0.5	−6872.3	**6.89**	−6849.5	**15.85**	−6842.5	**18.56**
	2.877 s	1.955 s	2.996 s	0.014 s	3.012 s	0.007 s

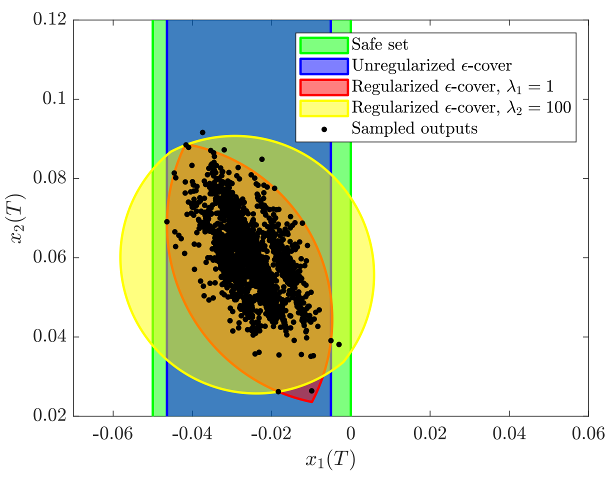

Fig. 2: Optimal ϵ-covers amongst intersections of two ℓ_2-norm ball $\frac{\epsilon}{2}$-covers for safe set $\mathcal{S}_2$.

A. Comparison to Output Set Estimation

In this example, we compare our proposed method to an alternate approach. In the second approach, we first estimate the output set of the neural network using the scenario-based reachability analysis in [25]. We then use the resulting output set estimate to assess robustness. Recall that our proposed scenario optimization (6) generalizes the reachability analysis of [25]. In addition to localizing the network outputs, our approach directly takes the goal of robustness certification into account, whereas the estimation technique of [25] does not.

To illustrate our comparison, consider a simple ReLU neural network given by $f\colon \mathbb{R}^2 \to \mathbb{R}^2$, where $f_i(x) = \max\{0, x_i\}$ for $i \in \{1, 2\}$. The noisy input X is distributed uniformly on the input set $\mathcal{X} = \{x \in \mathbb{R}^2 : \|x - \bar{x}\|_1 \leq 1\}$, where $\bar{x} = (1, 0)$. The safe set is given as $\mathcal{S} = \{y \in \mathbb{R}^2 : a^\top y + b \geq 0\}$, where $a = (0, 1)$ and $b = 0.5$. It is straightforward to show that the output set is the top-half of the input set, namely, $\mathcal{Y} = \mathcal{X} \cap \{y \in \mathbb{R}^2 : y_2 \geq 0\}$. Hence, if $y \in \mathcal{Y}$ then $a^\top y + b = y_2 + b \geq b \geq 0$. Therefore, $\mathcal{Y} \subseteq \mathcal{S}$, and so the random output $Y = f(X)$ is safe with probability one.

We now perform the two assessments at hand, computing our proposed solution first. We choose the ℓ_2-norm ball class for our candidate ϵ-covers and draw sufficiently many output samples $\{y_j\}_{j=1}^N$ according to Theorem 2 with $\epsilon = 0.1$ and $\delta = 10^{-5}$. Next, we choose the regularizer $v(\bar{y}, r) = r^2$ with $\lambda = 0.1$ and solve the scenario problem (7) for the ℓ_2-norm ball class. The solution correctly certifies that network outputs are safe with high probability; see the blue set in Figure 3.

We now turn to the alternative method. We use the same ℓ_2-norm ball class as above and solve for the minimum volume ϵ-cover using the same N sampled outputs. The estimated

TABLE II: Experiment of Section VI-B and Table Ia using [17] with percentages of inputs that each method is able to certify.

Radius	$\epsilon = 0.001$		$\epsilon = 0.1$		$\epsilon = 0.25$	
	[17]	Ours	[17]	Ours	[17]	Ours
0.01	**0.00**	**14.11**	**0.00**	**14.26**	**0.00**	**14.28**
	100%	100%	100%	100%	100%	100%
0.027	**0.00**	**13.36**	**0.00**	**13.77**	**0.00**	**13.85**
	100%	100%	100%	100%	100%	100%
0.05	**0.00**	**12.34**	**0.00**	**13.11**	**0.00**	**13.26**
	100%	100%	100%	100%	100%	100%
0.1	**0.00**	**10.25**	**0.00**	**11.66**	**0.00**	**12.00**
	100%	100%	100%	100%	100%	100%
0.5	N/A	−7.21	N/A	−0.05	N/A	**1.71**
	20%	20%	40%	40%	50%	50%

TABLE III: Average probabilistic robustness level lower bounds $\hat{r}(\theta^*)$ for various other models. Values for Models 1 and 2 are averaged over 10 inputs, and for Model 3 they are averaged over 100 network realizations. Lower bounds giving certified robustness (on average) are bolded, and the average certified adversarial radii computed using [36] are italicized.

	Model 1			Model 2			Model 3	
Radius	PRVN	Ours	Radius	PRVN	Ours	Radius	PRVN	Ours
0.005	**7.81**	**19.01**	0.001	**48.80**	**33.26**	0.01	**1.96**	**1.97**
0.0068	**2.20**	**18.96**	*0.0023*	**10.90**	**33.17**	0.05	**1.71**	**1.79**
0.01	−26.31	**18.88**	0.003	−91.20	**33.12**	0.1	**1.40**	**1.56**
0.05	−1769.97	**17.83**	0.005	−1056.85	**32.98**	0.5	−1.06	−0.29
0.1	−4493.02	**16.48**	0.01	−8717.05	**32.62**	1.0	−4.13	−2.60

output set is shown in red in Figure 3. Despite being a tighter localization, a substantial portion of the estimated output set exits the safe set, meaning that this approach cannot certify the robustness of the network, even though the random output is truly safe with probability one. This comparison shows that a good estimate of the output set may not be the most informative set to use for assessing output safety. This observation endorses our proposed method, which simultaneously encodes both goals of certification and localization.

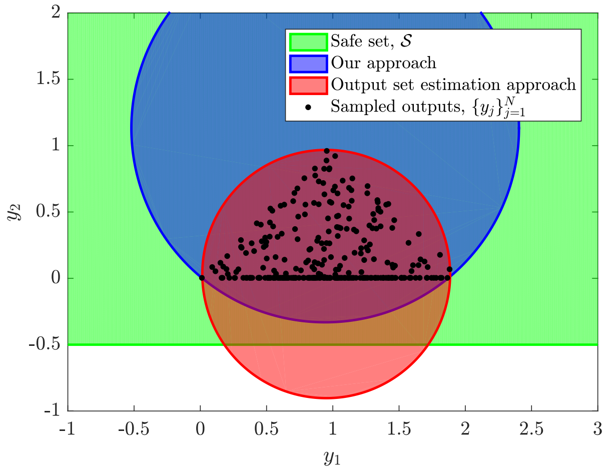

Fig. 3: The tightest ϵ-cover of the output set (red) does not correctly certify robustness. Our approach (blue) correctly certifies robustness and maintains reasonable localization.

B. Exploiting Network Structure

In this experiment, we implement the complexity-reducing method of Section V. We consider networks with 10 inputs, 10 outputs, and 250 neurons in every hidden layer. The number of layers K varies from 3 to 25. The weights and biases for every architecture are chosen randomly (with Gaussian elements, then normalized). Every activation function $\sigma^{(k)}$ is chosen to be ReLU, with preactivation and affine bounds derived according to [10]. We consider (randomly chosen Gaussian) clean inputs $\bar{x}$ with uniform additive random noise on the ℓ_∞-norm ball with radius $\epsilon_x = 0.1$, so that the noisy inputs X are distributed uniformly on $\{x \in \mathbb{R}^{n_x} : \|x - \bar{x}\|_\infty \leq \epsilon_x\}$.

For every architecture, we lower-bound the probabilistic robustness level for 50 different realizations of the weights, biases, and inputs, where for each realization we solve the scenario optimization problem using the class $\mathcal{H} = \{\{y \in \mathbb{R}^{n_y} : c^\top y + d \geq 0\} : (c,d) \in \mathbb{R}^{n_y} \times \mathbb{R}\}$ of half-spaces with $N = 1000$ sampled inputs. This is done both using our baseline methodology, maintaining the full nonlinearity of each deep network, as well as using the shallow surrogate networks proposed in Section V. Figure 4 displays the ratio $T_{f'}/T_f$ between the sampling time $T_{f'}$ (averaged over all realizations of a given depth) for the shallow surrogate network f' and the sampling time T_f (again, averaged) for the deep network f. We see that, when $k^* = O(K)$, meaning that the majority of nonlinearity is maintained in f', the sampling times remain roughly the same. On the other hand, when $k^* = O(1)$, meaning the majority of nonlinearity is replaced by affine bounds, the sampling time is reduced by nearly two orders of magnitude, and the reduction follows the expected rate of $k^*/K = O(1/K)$. For in-between surrogate architectures using $k^* = O(\log K)$ and $k^* = O(\sqrt{K})$, we find respectable time complexity reductions, nearing an order of magnitude decrease in sampling time. The decreases in the lower bound on the probabilistic robustness level are also shown in Figure 4. The average lower bound r_f without exploiting structure is 0.1. Therefore, the degradation of the bound incurred by using the shallow surrogate networks is relatively constant and minimal. The experiment results in the same conclusions when using tanh activation functions, and when using smaller and larger input set radii ϵ_x.

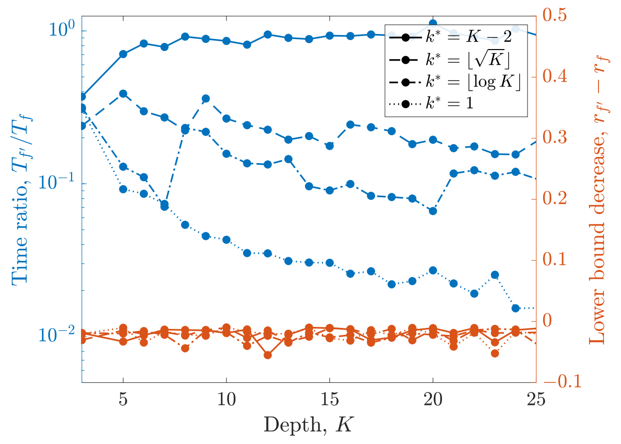

Fig. 4: Ratio between the average sampling time of the shallow surrogate network f' and that of the deep original network f, and the corresponding decrease in the lower bound on the probabilistic robustness level.