2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

Probabilistic Verification and Reachability Analysis of Neural Networks via Semidefinite Programming

Mahyar Fazlyab, Manfred Morari, George J. Pappas

Abstract— **Quantifying the robustness of neural networks or verifying their safety properties against input uncertainties or adversarial attacks have become an important research area in learning-enabled systems. Most results concentrate around the worst-case scenario where the input of the neural network is perturbed within a norm-bounded uncertainty set. In this paper, we consider a probabilistic setting in which the uncertainty is random with known first two moments. In this context, we discuss two relevant problems: (i) probabilistic safety verification, in which the goal is to find an upper bound on the probability of violating a safety specification; and (ii) confidence ellipsoid estimation, in which given a confidence ellipsoid for the input of the neural network, our goal is to compute a confidence ellipsoid for the output. Due to the presence of nonlinear activation functions, these two problems are very difficult to solve exactly. To simplify the analysis, our main idea is to abstract the nonlinear activation functions by a combination of affine and quadratic constraints they impose on their input-output pairs. We then show that the safety of the abstracted network, which is sufficient for the safety of the original network, can be analyzed using semidefinite programming. We illustrate the performance of our approach with numerical experiments.**

I. Introduction

Neural Networks (NN) have been very successful in various applications such as end-to-end learning for self-driving cars [1], learning-based controllers in robotics [2], speech recognition, and image classifiers. Their vulnerability to input uncertainties and adversarial attacks, however, refutes the deployment of neural networks in safety critical applications. In the context of image classification, for example, it has been shown in several works [3]–[5] that even adding an imperceptible noise to the input of neural network-based classifiers can completely change their decision. In this context, verification refers to the process of checking whether the output of a trained NN satisfies certain desirable properties when its input is perturbed within an uncertainty model. More precisely, we would like to verify whether the neural network's prediction remains the same in a neighborhood of a test point $x^\star$. This neighborhood can represent, for example, the set of input examples that can be crafted by an adversary.

In *worst-case* safety verification, we assume that the input uncertainty is bounded and we verify a safety property for all possible perturbations within the uncertainty set. This approach has been pursued extensively in several works using various tools, such as mixed-integer linear programming [6]–[8], robust optimization and duality theory [9], [10], Satisfiability Modulo Theory (SMT) [11], dynamical systems [12], [13], Robust Control [14], Abstract Interpretation [15] and many others [16], [17].

In *probabilistic* verification, on the other hand, we assume that the input uncertainty is random but potentially unbounded. Random uncertainties can emerge as a result of, for example, data quantization, input preprocessing, and environmental background noises [18]. In contrast to the worst-case approach, there are only few works that have studied verification of neural networks in probabilistic settings [18]–[20]. In situations where we have random uncertainty models, we ask a related question: "Can we provide statistical guarantees on the output of neural networks when their input is perturbed with a random noise?" In this paper, we provide an affirmative answer by addressing two related problems:

- *Probabilistic Verification:* Given a safe region in the output space of the neural network, our goal is estimate the probability that the output of the neural network will be in the safe region when its input is perturbed by a random variable with a known mean and covariance.
- *Confidence propagation:* Given a confidence ellipsoid on the input of the neural network, we want to estimate the output confidence ellipsoid.

The rest of the paper is organized as follows. In Section II, we discuss safety verification of neural networks in both deterministic and probabilistic settings. In Section III, we provide an abstraction of neural networks using the formalism of quadratic constraints. In Section IV we develop a convex relaxation to the problem of confidence ellipsoid estimation. In Section V, we present the numerical experiments. Finally, we draw our conclusions in Section VI.

A. Notation and Preliminaries

We denote the set of real numbers by $\mathbb{R}$, the set of real n-dimensional vectors by $\mathbb{R}^n$, the set of $m \times n$-dimensional matrices by $\mathbb{R}^{m\times n}$, and the n-dimensional identity matrix by I_n. We denote by $\mathbb{S}^n$, $\mathbb{S}^n_+$, and $\mathbb{S}^n_{++}$ the sets of n-by-n symmetric, positive semidefinite, and positive definite matrices, respectively. We denote ellipsoids in $\mathbb{R}^n$ by

$$\mathcal{E}(x_c, P) = \{x \mid (x - x_c)^\top P^{-1}(x - x_c) \leq 1\},$$

where $x_c \in \mathbb{R}^n$ is the center of the ellipsoid and $P \in \mathbb{S}^n_{++}$ determines its orientation and volume. We denote the mean and covariance of a random variable $X \in \mathbb{R}^n$ by $\mathbf{E}[X] \in \mathbb{R}^n$ and $\mathbf{Cov}[X] \in \mathbb{S}^n_+$, respectively.

†Corresponding author: mahyarfa@seas.upenn.edu. This work was supported by DARPA Assured Autonomy and NSF CPS 1837210. The authors are with the Department of Electrical and Systems Engineering, University of Pennsylvania. Email: {mahyarfa, morari, pappasg}@seas.upenn.edu.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:25 UTC from IEEE Xplore. Restrictions apply.

II. Safety Verification of Neural Networks

A. Deterministic Safety Verification

Consider a multi-layer feed-forward fully-connected neural network described by the following equations,

$$\begin{aligned} x^0 &= x \\ x^{k+1} &= \phi(W^k x^k + b^k) \quad k = 0, \cdots, \ell - 1 \\ f(x) &= W^\ell x^\ell + b^\ell, \end{aligned} \tag{1}$$

where $x^0 = x$ is the input to the network, $W^k \in \mathbb{R}^{n_{k+1} \times n_k}$, $b^k \in \mathbb{R}^{n_{k+1}}$ are the weight matrix and bias vector of the k-th layer. The nonlinear activation function $\phi(\cdot)$ (Rectified Linear Unit (ReLU), sigmoid, tanh, leaky ReLU, etc.) is applied coordinate-wise to the pre-activation vectors, i.e., it is of the form

$$\phi(x) = [\varphi(x_1) \ \cdots \ \varphi(x_d)]^\top, \tag{2}$$

where φ is the activation function of each individual neuron. Although our framework is applicable to all activation functions, we focus our attention to ReLU activation functions, $\varphi(x) = \max(x, 0)$.

In deterministic safety verification, we are given a bounded set $\mathcal{X} \subset \mathbb{R}^{n_x}$ of possible inputs (the uncertainty set), which is mapped by the neural network to the output reachable set $f(\mathcal{X})$. The desirable properties that we would like to verify can often be described by a set $\mathcal{S} \subset \mathbb{R}^{n_y}$ in the output space of the neural network, which we call the safe region. In this context, the network is safe if $f(\mathcal{X}) \subseteq \mathcal{S}$.

B. Probabilistic Safety Verification

In a deterministic setting, reachability analysis and safety verification is a yes/no problem whose answer does not quantify the proportion of inputs for which the safety is violated. Furthermore, if the uncertainty is random and potentially unbounded, the output $f(x)$ would satisfy the safety constraint only with a certain probability. More precisely, given a safe region $\mathcal{S}$ in the output space of the neural network, we are interested in finding the probability that the neural network maps the random input X to the safe region,

$$\mathbf{Pr}(f(X) \in \mathcal{S}).$$

Since $f(x)$ is a nonlinear function, computing the distribution of $f(X)$ given the distribution of X is prohibitive, except for special cases. As a result, we settle for providing a *lower bound* $p \in (0, 1)$ on the desired probability,

$$\mathbf{Pr}(f(X) \in \mathcal{S}) \geq p.$$

To compute the lower bound, we adopt a geometrical approach, in which we verify whether the reachable set of a confidence region of the input lies entirely in the safe set $\mathcal{S}$. We first recall the definition of a confidence region.

Definition 1 (Confidence region) *The p-level ($p \in [0, 1]$) confidence region of a vector random variable $X \in \mathbb{R}^n$ is defined as any set $\mathcal{E}_p \subseteq \mathbb{R}^n$ for which $\mathbf{Pr}(X \in \mathcal{E}_p) \geq p$.*

Although confidence regions can have different representations, our particular focus in this paper is on ellipsoidal confidence regions. Due to their appealing geometric properties (e.g., invariance to affine subspace transformations), ellipsoids are widely used in robust control to compute reachable sets [21]–[23].

The next two lemmas characterize confidence ellipsoids for Gaussian random variables and random variables with known first two moments.

Lemma 1 *Let $X \sim \mathcal{N}(\mu, \Sigma)$ be an n-dimensional Gaussian random variable. Then the p-level confidence region of X is given by the ellipsoid*

$$\mathcal{E}_p = \{x \mid (x - \mu)^\top \Sigma^{-1} (x - \mu) \leq \chi_n^2(p)\}, \tag{3}$$

where $\chi_n^2(p)$ is the quantile function of the chi-squared distribution with n degrees of freedom.

For non-Gaussian random variables, we can use Chebyshev's inequality to characterize the confidence ellipsoids, if we know the first two moments.

Lemma 2 *Let X be an n-dimensional random variable with $\mathbf{E}[X] = \mu$ and $\mathbf{Cov}[X] = \Sigma$. Then the ellipsoid*

$$\mathcal{E}_p = \{x \mid (x - \mu)^\top \Sigma^{-1} (x - \mu) \leq \frac{n}{1 - p}\}, \tag{4}$$

is a p-level confidence region of X.

Lemma 3 *Let $\mathcal{E}_p$ be a confidence region of a random variable X. If $f(\mathcal{E}_p) \subseteq \mathcal{S}$, then $\mathcal{S}$ is a p-level confidence region for the random variable $f(X)$, i.e., $\mathbf{Pr}(f(X) \in \mathcal{S}) \geq p$.*

Proof: The inclusion $f(\mathcal{E}_p) \subseteq \mathcal{S}$ implies $\mathbf{Pr}(f(X) \in \mathcal{S}) \geq \mathbf{Pr}(f(X) \in f(\mathcal{E}_p))$. Since f is not necessarily a one-to-one mapping, we have $\mathbf{Pr}(f(X) \in f(\mathcal{E}_p)) \geq \mathbf{Pr}(X \in \mathcal{E}_p) \geq p$. Combining the last two inequalities yields the desired result. ■

According to Lemma 3, if we can certify that the output reachable set $f(\mathcal{E}_p)$ lies entirely in the safe set $\mathcal{S}$ for some $p \in (0, 1)$, then the network is safe with probability at least p. In particular, finding the best lower bound corresponds to the non-convex optimization problem,

$$\text{maximize} \quad p \quad \text{subject to } f(\mathcal{E}_p) \subseteq \mathcal{S}, \tag{5}$$

with decision variable $p \in [0, 1)$. By Lemma 3, the optimal solution $p^\star$ then satisfies

$$\mathbf{Pr}(f(X) \in \mathcal{S}) \geq p^\star. \tag{6}$$

C. Confidence Propagation

A closely related problem to probabilistic safety verification is confidence propagation. Explicitly, given a p-level confidence region $\mathcal{E}_p$ of the input of a neural network, our goal is to find a p-level confidence region for the output. To see the connection to the probabilistic verification problem, let $\mathcal{S}$ be any outer approximation of the output reachable set, i.e., $f(\mathcal{E}_p) \subseteq \mathcal{S}$. By lemma 3, $\mathcal{S}$ is a p-level confidence

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:25 UTC from IEEE Xplore. Restrictions apply.

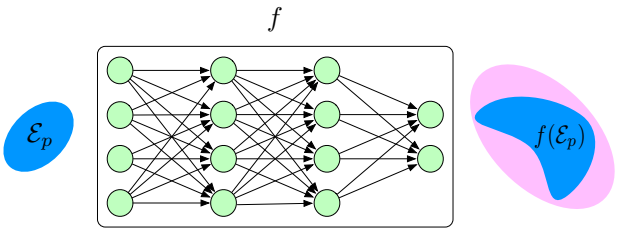


Fig. 1: p-level input confidence ellipsoid $\mathcal{E}_p$, its image $f(\mathcal{E}_p)$, and the estimated output confidence ellipsoid.

region for the output. Of course, there is an infinite number of such possible confidence regions. Our goal is find the "best" confidence region with respect to some metric. Using the volume of the ellipsoid as an optimization criterion, the best confidence region amounts to solving the problem

$$\text{minimize Volume}(\mathcal{S}) \quad \text{subject to } f(\mathcal{E}_p) \subseteq \mathcal{S}. \tag{7}$$

The solution to the above problem provides the p-level confidence region with the minimum volume. Figure 1 illustrates the procedure of confidence estimation. In the next section, we provide a convex relaxation of the optimization problem (7). The other problem in (5) is a straightforward extension of confidence estimation, and hence, we will not discuss the details.

III. Problem Relaxation via Quadratic Constraints

Due to the presence of nonlinear activation functions, checking the condition $f(\mathcal{E}_p) \subseteq \mathcal{S}$ in (5) or (7) is a non-convex feasibility problem and is NP-hard, in general. Our main idea is to abstract the original network f by another network $\tilde{f}$ in the sense that $\tilde{f}$ over-approximates the output of the original network for any input ellipsoid, i.e., $f(\mathcal{E}_p) \subseteq \tilde{f}(\mathcal{E}_p)$ for any $p \in [0, 1)$. Then it will be sufficient to verify the safety properties of the relaxed network, i.e., verify the inclusion $\tilde{f}(\mathcal{E}_p) \subseteq \mathcal{S}$. In the following, we use the framework of quadratic constraints to develop such an abstraction.

A. Relaxation of Nonlinearities by Quadratic Constraints

In this subsection, we show how we can abstract activation functions, and in particular the ReLU function, using quadratic constraints. We first provide a formal definition, introduced in [14].

Definition 2 *Let $\phi\colon \mathbb{R}^d \to \mathbb{R}^d$ be and suppose $\mathcal{Q} \subset \mathbb{S}^{2d+1}$ is the set of all symmetric and indefinite matrices Q such that the inequality*

$$\begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix}^\top Q \begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix} \geq 0, \tag{8}$$

holds for all $x \in \mathbb{R}^d$. Then we say ϕ satisfies the quadratic constraint defined by $\mathcal{Q}$.

Note that the matrix Q in Definition 2 is indefinite, or otherwise, the constraint trivially holds. Before deriving QCs for the ReLU function, we recall some definitions, which can be found in many references; for example [24], [25].

Definition 3 (Sector-bounded nonlinearity) *A nonlinear function $\varphi\colon \mathbb{R} \to \mathbb{R}$ is* sector-bounded *on the sector $[\alpha, \beta]$ ($0 \leq \alpha \leq \beta$) if the following condition holds for all x,*

$$(\varphi(x) - \alpha x)(\varphi(x) - \beta x) \leq 0. \tag{9}$$

Definition 4 (Slope-restricted nonlinearity) *A nonlinear function $\varphi(x)\colon \mathbb{R} \to \mathbb{R}$ is slope-restricted on $[\alpha, \beta]$ ($0 \leq \alpha \leq \beta$) if for any $(x, \varphi(x))$ and $(x^\star, \varphi(x^\star))$,*

$$(\varphi(x) - \varphi(x^\star) - \alpha(x - x^\star))(\varphi(x) - \varphi(x^\star) - \beta(x - x^\star)) \leq 0. \tag{10}$$

Repeated nonlinearities. Assuming that the same activation function is used in all neurons, we can exploit this structure to refine the QC abstraction of the nonlinearity. Explicitly, suppose $\varphi\colon \mathbb{R} \to \mathbb{R}$ is slope-restricted on $[\alpha, \beta]$ and let $\phi(x) = [\varphi(x_1) \ \cdots \varphi(x_d)]^\top$ be a vector-valued function constructed by component-wise repetition of φ. It is not hard to verify that ϕ is also slope-restricted in the same sector. However, this representation simply ignores the fact that all the nonlinearities that compose ϕ are the same. By taking advantage of this structure, we can refine the quadratic constraint that describes ϕ. To be specific, for an input-output pair $(x, \phi(x))$, $x \in \mathbb{R}^d$, we can write the slope-restriction condition

$$(\varphi(x_i) - \varphi(x_j) - \alpha(x_i - x_j))(\varphi(x_i) - \varphi(x_j) - \beta(x_i - x_j)) \leq 0, \tag{11}$$

for all distinct i, j. This particular QC can tighten the relaxation incurred by the QC abstraction of the nonlinearity.

There are several results in the literature about repeated nonlinearities. For instance, in [25], [26], the authors derive QCs for repeated and odd nonlinearities (e.g. tanh function).

B. QC for ReLU function

In this subsection, we derive quadratic constraints for the ReLU function, $\phi(x) = \max(0, x)$, $x \in \mathbb{R}^d$. Note that this function lies on the boundary of the sector $[0, 1]$. More precisely, we can describe the ReLU function by three quadratic and/or affine constraints:

$$y_i = \max(0, x_i) \Leftrightarrow y_i \geq x_i, \quad y_i \geq 0, \quad y_i^2 = x_i y_i. \tag{12}$$

On the other hand, for any two distinct indices $i \neq j$, we can write the constraint (11) with $\alpha = 0$, and $\beta = 1$,

$$(y_j - y_i)^2 \leq (y_j - y_i)(x_j - x_i). \tag{13}$$

By adding a weighted combination of all these constraints (positive weights for inequalities), we find that the ReLU function $y = \max(0, x)$ satisfies

$$\sum_{i=1}^{d} \lambda_i (y_i^2 - x_i y_i) + \nu_i (y_i - x_i) + \eta_i y_i - \sum_{i \neq j} \lambda_{ij} \left((y_j - y_i)^2 - (y_j - y_i)(x_j - x_i) \right) \geq 0, \tag{14}$$

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:25 UTC from IEEE Xplore. Restrictions apply.

for any multipliers $(\lambda_i, \nu_i, \eta_i, \lambda_{ij}) \in \mathbb{R} \times \mathbb{R}^3_+$ for $i, j \in \{1, \cdots, d\}$. This inequality can be written in the compact form (8), as stated in the following lemma.

Lemma 4 (QC for ReLU function) *The ReLU function, $\phi(x) = \max(0, x) \colon \mathbb{R}^d \to \mathbb{R}^d$, satisfies the QC defined by $\mathcal{Q}$ where*

$$\mathcal{Q} = \left\{ Q \mid Q = \begin{bmatrix} 0 & T & -\nu \\ T & -2T & \nu+\eta \\ -\nu^\top & \nu^\top+\eta^\top & 0 \end{bmatrix} \right\}. \tag{15}$$

Here $\eta, \nu \geq 0$ and $T \in \mathbb{S}^d_+$ is given by

$$T = \sum_{i=1}^{d} \lambda_i e_i e_i^\top + \sum_{i=1}^{d-1} \sum_{j>i}^{d} \lambda_{ij} (e_i - e_j)(e_i - e_j)^\top,$$

where e_i is the i-th basis vector in $\mathbb{R}^d$ and $\lambda_{ij} \geq 0$.

Proof: See [14]. ∎

Lemma 4 characterizes a family of valid QCs for the ReLU function. It is not hard to verify that the set $\mathcal{Q}$ of valid QCs is a convex cone. As we will see in the next section, the matrix Q in (15) appears as a decision variable in the optimization problem.

C. Tightening the Relaxation

In the previous subsection, we derived QCs that are valid for the whole space $\mathbb{R}^d$. When restricted to a region $\mathcal{R} \subseteq \mathbb{R}^d$, we can tighten the QC relaxation. Consider the relationship $\phi(x) = \max(0, x),\ x \in \mathcal{R} \subseteq \mathbb{R}^d$ and let $\mathcal{I}^+$, and $\mathcal{I}^-$ be the set of neurons that are always active or always inactive, i.e.,

$$\begin{aligned} \mathcal{I}^+ &= \{i \mid x_i \geq 0 \text{ for all } x \in \mathcal{R}\} \\ \mathcal{I}^- &= \{i \mid x_i < 0 \text{ for all } x \in \mathcal{R}\}. \end{aligned} \tag{16}$$

The constraint $y_i \geq x_i$ holds with equality for active neurons. Therefore, we can write

$$\nu_i \in \mathbb{R} \text{ if } i \in \mathcal{I}^+,\ \nu_i \geq 0 \text{ otherwise.}$$

Similarly, the constraint $y_i \geq 0$ holds with equality for inactive neurons. Therefore, we can write

$$\eta_i \in \mathbb{R} \text{ if } i \in \mathcal{I}^-,\ \eta_i \geq 0 \text{ otherwise.}$$

Finally, it can be verified that the cross-coupling constraint in (13) holds with equality for pairs of always active or always inactive neurons. Therefore, for any $1 \leq i < j \leq d$, we can write

$$\begin{aligned} &\lambda_{ij} \in \mathbb{R} \text{ if } (i,j) \in \mathcal{I}^+ \times \mathcal{I}^+ \text{ or } (i,j) \in \mathcal{I}^- \times \mathcal{I}^- \\ &\lambda_{ij} \geq 0 \text{ otherwise.} \end{aligned}$$

These additional degrees of freedom on the multipliers can tighten the relaxation incurred in (14). Note that the set of active or inactive neurons are not known *a priori*. However, we can partially find them using, for example, interval arithmetic.

IV. Analysis of the Relaxed Network via Semidefinite Programming

In this section, we use the QC abstraction developed in the previous section to analyze the safety of the relaxed network. In the next theorem, we state our main result for one-layer neural networks and will discuss the multi-layer case in Section IV-A.

Theorem 1 (Output covering ellipsoid) *Consider a one-layer neural network $f \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ described by the equation*

$$y = W^1 \phi(W^0 x + b^0) + b^1, \tag{17}$$

where $\phi \colon \mathbb{R}^{n_1} \to \mathbb{R}^{n_1}$ satisfies the quadratic constraint defined by $\mathcal{Q}$, i.e., for any $Q \in \mathcal{Q}$,

$$\begin{bmatrix} z \\ \phi(z) \\ 1 \end{bmatrix}^\top Q \begin{bmatrix} z \\ \phi(z) \\ 1 \end{bmatrix} \geq 0 \quad \text{for all } z. \tag{18}$$

Suppose $x \in \mathcal{E}(\mu_x, \Sigma_x)$. Consider the following matrix inequality

$$M_1 + M_2 + M_3 \preceq 0, \tag{19}$$

where

$$\begin{aligned} M_1 &= \begin{bmatrix} I_{n_x} & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} P(\tau) \begin{bmatrix} I_{n_x} & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}^\top \\ M_2 &= \begin{bmatrix} {W^0}^\top & 0 & 0 \\ 0 & I_{n_1} & 0 \\ {b^0}^\top & 0 & 1 \end{bmatrix} Q \begin{bmatrix} W^0 & 0 & b^0 \\ 0 & I_{n_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ M_3 &= \begin{bmatrix} 0 & 0 \\ {W^1}^\top & 0 \\ {b^1}^\top & 1 \end{bmatrix} S(A,b) \begin{bmatrix} 0 & W^1 & b^1 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

with

$$\begin{aligned} P(\tau) &= \tau \begin{bmatrix} -\Sigma_x^{-1} & \Sigma_x^{-1}\mu_x \\ \mu_x^\top \Sigma_x^{-1} & -\mu_x^\top \Sigma_x^{-1} \mu_x + 1 \end{bmatrix} \\ S(A,b) &= \begin{bmatrix} A^2 & Ab \\ b^\top A & b^\top b - 1 \end{bmatrix}. \end{aligned}$$

If (19) is feasible for some $(\tau, A, Q, b) \in \mathbb{R}_+ \times \mathbb{S}^{n_y} \times \mathcal{Q} \times \mathbb{R}^{n_y}$, then $y \in \mathcal{E}(\mu_y, \Sigma_y)$ with $\mu_y = -A^{-1}b$ and $\Sigma_y = A^{-2}$.

Proof: We first introduce the auxiliary variable z, and rewrite the equation of the neural network as

$$z = \phi(W^0 x + b^0) \quad y = W^1 z + b^1.$$

Since ϕ satisfies the QC defined by $\mathcal{Q}$, we can write the following QC from the identity $z = \phi(W^0 x + b^0)$:

$$\begin{bmatrix} W^0 x + b^0 \\ z \\ 1 \end{bmatrix}^\top Q \begin{bmatrix} W^0 x + b^0 \\ z \\ 1 \end{bmatrix} \geq 0, \text{ for all } Q \in \mathcal{Q}. \tag{21}$$

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:25 UTC from IEEE Xplore. Restrictions apply.

By substituting the identity

$$\begin{bmatrix} W^0x+b^0 \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} W^0 & 0 & b^0 \\ 0 & I_{n_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ z \\ 1 \end{bmatrix},$$

back into (21) and denoting $\mathbf{x} = [x^\top \ z^\top]^\top$, we can write the inequality

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top M_2 \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \geq 0, \tag{22}$$

for any $Q \in \mathcal{Q}$ and all $\mathbf{x}$. By definition, for all $x \in \mathcal{E}(\mu_x, \Sigma_x)$, we have $(x-\mu_x)^\top \Sigma_x^{-1}(x-\mu_x) \leq 1$, which is equivalent to

$$\tau \begin{bmatrix} x \\ 1 \end{bmatrix}^\top \begin{bmatrix} -\Sigma_x^{-1} & \Sigma_x^{-1}\mu_x \\ \mu_x^\top \Sigma_x^{-1} & -\mu_x^\top \Sigma_x^{-1}\mu_x + 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0.$$

By using the identity

$$\begin{bmatrix} x \\ 1 \end{bmatrix} = \begin{bmatrix} I_{n_x} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ z \\ 1 \end{bmatrix},$$

we conclude that for all $x \in \mathcal{E}(\mu_x, \Sigma_x),\ z = \phi(W^0x+b)$,

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top M_1 \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \geq 0. \tag{23}$$

Suppose (19) holds for some $(A, Q, b) \in \mathbb{S}^{n_y} \times \mathcal{Q} \times \mathbb{R}^{n_y}$. By left- and right- multiplying both sides of (18) by $[\mathbf{x}^\top \ 1]$ and $[\mathbf{x}^\top \ 1]^\top$, respectively, we obtain

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top M_1 \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top M_2 \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top M_3 \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \leq 0.$$

For any $x \in \mathcal{E}(\mu_x, \Sigma_x)$ the first two quadratic terms are nonnegative by (23) and (22), respectively. Therefore, the last term on the left-hand side must be nonpositive for all $x \in \mathcal{E}(\mu_x, \Sigma_x)$,

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top M_3 \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \leq 0.$$

But the preceding inequality, using the relation $y = W^1 z + b^1$, is equivalent to

$$\begin{bmatrix} y \\ 1 \end{bmatrix}^\top \begin{bmatrix} A^2 & Ab \\ b^\top A & b^\top b - 1 \end{bmatrix} \begin{bmatrix} y \\ 1 \end{bmatrix} \leq 0,$$

which is equivalent to $(y+A^{-1}b)^\top A^2 (y+A^{-1}b) \leq 1$. Using our notation for ellipsoids, this means for all $x \in \mathcal{E}(\mu_x, \Sigma_x)$, we must have $y \in \mathcal{E}(-A^{-1}b, A^{-2})$. ∎

In Theorem 1, we proposed a matrix inequality, in variables (Q, A, b), as a sufficient condition for enclosing the output of the neural network with the ellipsoid $\mathcal{E}(-A^{-1}b, A^{-2})$. We can now use this result to find the minimum-volume ellipsoid with this property. Note that the matrix inequality (19) is not linear in (A, b). Nevertheless, we can convexify it by using Schur Complements.

Lemma 5 *The matrix inequality in* (19) *is equivalent to the linear matrix inequality (LMI)*

$$M = \left[\begin{array}{ccc|c} & M_1 + M_2 - ee^\top & & \begin{matrix} 0_{n_x \times n_y} \\ {W^1}^\top A \\ {b^1}^\top A + b^\top \end{matrix} \\ \hline 0_{n_y \times n_x} & AW^1 & Ab^1+b & -I_{n_y} \end{array}\right] \succeq 0, \tag{24}$$

in (τ, A, Q, b), *where* $e = (0, \cdots, 0, 1) \in \mathbb{R}^{n_x+n_1+1}$.

Proof: It is not hard to verify that M_3 can be written as $M_3 = FF^\top - ee^\top$, where F, affine in (A, b), is given by

$$F(A, b) = \begin{bmatrix} 0_{n_x \times n_y} \\ {W^1}^\top A \\ {b^1}^\top A + b^\top \end{bmatrix}.$$

Using this definition, the matrix inequality in (19) reads $(M_1 + M_2 - ee^\top) + FF^\top \succeq 0$, which implies that the term in the parentheses must be non-negative, i.e., $M_1 + M_2 - ee^\top \succeq 0$. Using Schur Complements, the last two inequalities are equivalent to (24). ∎

Having established Lemma 5, we can now find the minimum-volume covering ellipsoid by solving the following semidefinite program (SDP),

$$\text{minimize} \quad -\log\det(A) \ \text{ subject to } \ (24). \tag{25}$$

where the decision variables are $(\tau, A, Q, b) \in \mathbb{R}_+ \times \mathbb{S}^{n_y} \times \mathcal{Q} \times \mathbb{R}^{n_y}$. Since $\mathcal{Q}$ is a convex cone, (25) is a convex program and can be solved via interior-point method solvers.

A. Multi-layer Case

For multi-layer neural networks, we can apply the result of Theorem 1 in a layer-by-layer fashion provided that the input confidence ellipsoid of each layer is non-degenerate. This assumption holds when for all $0 \leq k \leq \ell - 1$ we have $n_{k+1} \leq n_k$ (reduction in the width of layers), and the weight matrices $W^k \in \mathbb{R}^{n_{k+1} \times n_k}$ are full rank. To see this, we note that ellipsoids are invariant under affine subspace transformations such that

$$W^k \mathcal{E}(\mu^k, \Sigma^k) + b^k = \mathcal{E}(W^k\mu^k + b^k, W^k\Sigma^k {W^k}^\top).$$

This implies that $\Sigma_{k+1} := W^k\Sigma^k {W^k}^\top$ is positive definite whenever Σ^k is positive definite, implying that the ellipsoid $\mathcal{E}(\mu_{k+1}, \Sigma_{k+1})$ is non-degenerate. If the assumption $n_{k+1} \leq n_k$ is violated, we can use the compact representation of multi-layer neural networks elaborated in [14] to arrive at the multi-layer couterpart of the matrix inequality in (19).

V. Numerical Experiments

In this section, we consider a numerical experiment, in which we estimate the confidence ellipsoid of a one-layer neural network with $n_x = 2$ inputs, $n_1 \in \{10, 30, 50\}$ hidden neurons and $n_y = 2$ outputs. We assume the input is Gaussian with $\mu_x = (1, 1)$ and $\Sigma_x = \mathrm{diag}(1, 2)$. The weights and biases of the network are chosen randomly. We use MATLAB, CVX [27], and Mosek [28] to solve the

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:25 UTC from IEEE Xplore. Restrictions apply.

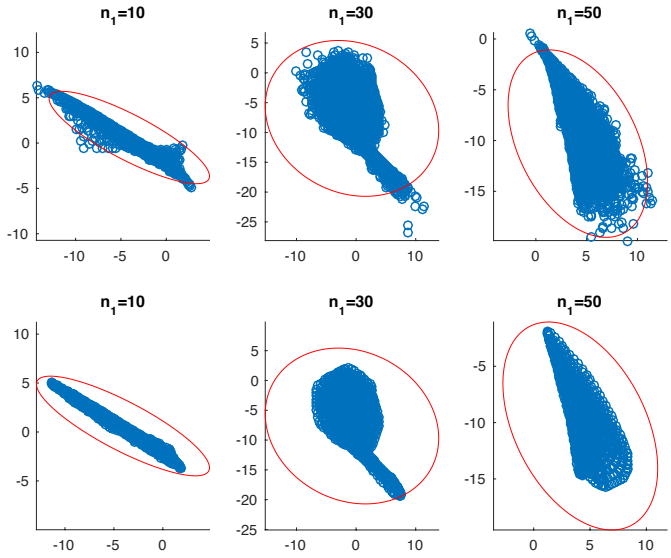


Fig. 2: Top: the estimated 95% confidence ellipsoid along with 10^4 samples of the output. Bottom: The image of the 95% input confidence ellipsoid ($f(\mathcal{E}_p)$ with $p = 0.95$) and its outer approximation (the output confidence ellipsoid).

corresponding SDP. In Figure 2, we plot the estimated 0.95-level output confidence ellipsoid along with 10^4 sample outputs. We also plot the image of 0.95-level input confidence ellipsoid under f along with the estimated 0.95-level output confidence ellipsoid.

VI. Conclusions

We studied probabilistic safety verification of neural networks when their inputs are subject to random noise with known first two moments. Instead of analyzing the network directly, we proposed to study the safety of an abstracted network instead, in which the nonlinear activation functions are relaxed by the quadratic constraints their input-output pairs satisfy. We then showed that we can analyze the safety properties of the abstracted network using semidefinite programming. It would be interesting to consider other related problems such as closed-loop statistical safety verification and reachability analysis.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[2] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," *arXiv preprint arXiv:1811.08027*, 2018.

[3] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 4480–4488, 2016.

[4] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *arXiv preprint*, 2017.

[5] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, 2019.

[6] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi, "Measuring neural net robustness with constraints," in *Advances in neural information processing systems*, pp. 2613–2621, 2016.

[7] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *arXiv preprint arXiv:1706.07351*, 2017.

[8] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," *arXiv preprint arXiv:1711.07356*, 2017.

[9] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, vol. 1, no. 2, p. 3, 2017.

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

[11] L. Pulina and A. Tacchella, "Challenging smt solvers to verify neural networks," *AI Communications*, vol. 25, no. 2, pp. 117–135, 2012.

[12] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: verifying safety properties of hybrid systems with neural network controllers," *arXiv preprint arXiv:1811.01828*, 2018.

[13] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE transactions on neural networks and learning systems*, no. 99, pp. 1–7, 2018.

[14] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *arXiv preprint arXiv:1903.01287*, 2019.

[15] M. Mirman, T. Gehr, and M. Vechev, "Differentiable abstract interpretation for provably robust neural networks," in *International Conference on Machine Learning*, pp. 3575–3583, 2018.

[16] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Advances in Neural Information Processing Systems*, pp. 2266–2276, 2017.

[17] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Efficient formal safety analysis of neural networks," in *Advances in Neural Information Processing Systems*, pp. 6369–6379, 2018.

[18] T.-W. Weng, P.-Y. Chen, L. M. Nguyen, M. S. Squillante, I. Oseledets, and L. Daniel, "Proven: Certifying robustness of neural networks with a probabilistic approach," *arXiv preprint arXiv:1812.08329*, 2018.

[19] K. Dvijotham, M. Garnelo, A. Fawzi, and P. Kohli, "Verification of deep probabilistic models," *arXiv preprint arXiv:1812.02795*, 2018.

[20] A. Bibi, M. Alfadly, and B. Ghanem, "Analytic expressions for probabilistic moments of pl-dnn with gaussian input," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9099–9107, 2018.

[21] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 7130–7135, IEEE, 2018.

[22] D. Van Hessem and O. Bosgra, "A conic reformulation of model predictive control including bounded and stochastic disturbances under state and input constraints," in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, vol. 4, pp. 4643–4648, IEEE, 2002.

[23] M. Cannon, B. Kouvaritakis, S. V. Rakovic, and Q. Cheng, "Stochastic tubes in model predictive control with probabilistic constraints," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 194–200, 2011.

[24] A. Megretski and A. Rantzer, "System analysis via integral quadratic constraints," *IEEE Transactions on Automatic Control*, vol. 42, no. 6, pp. 819–830, 1997.

[25] F. D'amato, M. A. Rotea, A. Megretski, and U. Jönsson, "New results for analysis of systems with repeated nonlinearities," *Automatica*, vol. 37, no. 5, pp. 739–747, 2001.

[26] V. V. Kulkarni and M. G. Safonov, "All multipliers for repeated monotone nonlinearities," *IEEE Transactions on Automatic Control*, vol. 47, no. 7, pp. 1209–1212, 2002.

[27] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming," 2008.

[28] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:25 UTC from IEEE Xplore. Restrictions apply.