Probabilistic Verification of ReLU Neural Networks via Characteristic Functions

Joshua Pilipovsky JPILIPOVSKY3@GATECH.EDU
Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology

Vignesh Sivaramakrishnan VIGSIV@UNM.EDU
Department of Electrical and Computer Engineering, University of New Mexico

Meeko M. K. Oishi OISHI@UNM.EDU
Department of Electrical and Computer Engineering, University of New Mexico

Panagiotis Tsiotras TSIOTRAS@GATECH.EDU
Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

Verifying the input-output relationships of a neural network to achieve desired performance specifications is a difficult, yet important, problem due to the growing ubiquity of neural nets in many engineering applications. We use ideas from probability theory in the frequency domain to provide probabilistic verification guarantees for ReLU neural networks. Specifically, we interpret a (deep) feedforward neural network as a discrete-time dynamical system over a finite horizon that shapes distributions of initial states, and use characteristic functions to propagate the distribution of the input data through the network. Using the inverse Fourier transform, we obtain the corresponding cumulative distribution function of the output set, which we use to check if the network is performing as expected given any random point from the input set. The proposed approach does not require distributions to have well-defined moments or moment generating functions. We demonstrate our proposed approach on two examples, and compare its performance to related approaches.

Keywords: Neural networks, ReLU, verification, characteristic functions, distributional control.

1. Introduction

Neural networks (NN) have become a powerful tool in recent years for a large class of applications, including image classification (Yang et al., 2018), speech recognition (Chiu et al., 2018), autonomous driving (Huang and Chen, 2020), drone acrobatics (Song et al., 2021), and many others. The formal verification of neural networks is crucial for their wider adoption in safety-critical scenarios. The main difficulty with the use of (deep) NN for safety-critical applications lies in the demonstrated sensitivity of DNNs to input uncertainties and/or adversarial attacks. For example, in the context of image classification, adding even a small amount of noise to the input set can greatly change the network output (Su et al., 2019; Moosavi-Dezfooli et al., 2017). For safety-critical applications, DNNs should be robust or insensitive to input uncertainties, a property that can be tested by verifying that the network prescribes to certain output specifications subject to various inputs.

Verification frameworks for DNNs can be classified as either *deterministic* or *probabilistic*. In deterministic verification, one maps an input set is to an output set; if any output falls outside the safety set, the verification fails. This is *worst-case* safety verification since the input set can be

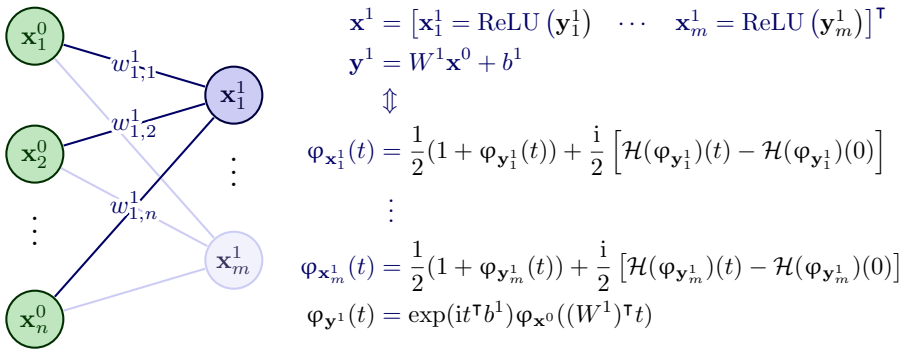


Figure 1: The characteristic function (CF) of the input data can be propagated through a ReLU network. This enables one to query the output CF of the network to answer questions such as being out-of-distribution. The use of CFs also applies where the underlying distributions do not have any moments or moment-generating functions (e.g., Cauchy distribution).

treated as an uncertainty set centered around some nominal input. Given some input x_0 and a neural network $f : x \mapsto y$, deterministic verification can be posed as a nonlinear program (NLP), with the objective function quantifying satisfaction of some safety rule $y \in \mathcal{S}$. In general, though, the resulting NLP is intractable using standard off-the-shelf solvers. Several works have used mixed-integer linear programming (MILP) (Lomuscio and Maganti, 2017; Cheng et al., 2017), Satisfiability Modulo Theories (SMT) (Katz et al., 2017; Scheibler et al., 2015), or semi-definite programming (SDP) (Brown et al., 2022; Fazlyab et al., 2022; Dvijotham et al., 2020; Dathathri et al., 2020; Wong and Kolter, 2018; Raghunathan et al., 2018), to recast and solve this NLP problem. In recent work, given an input or an output polytope, one can generate the respective output or input polytope through the ReLU neural network (Vincent and Schwager, 2021).

In probabilistic verification, the input set itself is uncertain and potentially unbounded. Random uncertainties naturally arise in practical applications, for example, from signal processing, environmental noise, and other exogenous disturbances. For example, impulse noise from Cauchy distributions arise in various sensing and imaging domains (Sciacchitano et al., 2015; Mei et al., 2018). In this context, the uncertainties are modeled in terms of probability distributions, and the verification problem considers the *probability* that the output is in a safety set given a random input from the input set. Given a random input vector $\mathbf{x}_0$ and a neural network f, the probability that the output random vector $\mathbf{y} = f(\mathbf{x}_0)$ lies in some safety set $\mathcal{S}$ is greater than some threshold $1 - p$ is given by the chance constraint,

$$\mathbb{P}(\mathbf{y} \in \mathcal{S}) \geq 1 - p. \tag{1}$$

Few works have studied the verification of DNNs in a probabilistic setting; most of the existing approaches involve under- or over-approximations. In Fazlyab et al. (2019), the authors approximate an output confidence ellipsoid via an SDP, and verify residing in a set via equivalence between confidence sets and chance constraints. PROVEN (Weng et al., 2019) accommodates bounded disturbances, using linear approximations of activation functions and concentration inequalities to generate bounds on (1). In Pautov et al. (2022), a similar approach is taken with Cramer-Chernoff concentration inequalities, and propagates samples through the network. Generative DNNs are considered in Berrada et al. (2021), which formulates an upper bound on the chance constraint via

duality. Lastly, a scenario optimization approach in Anderson and Sojoudi (2022) constructs a lower bound on (1) that depends upon the number of samples.

In this paper, we interpret a DNN as a dynamical system (Narendra and Parthasarathy, 1992, 1990; Weinan, 2017) that shapes distributions of data and view the verification problem as one of propagating a distribution through a linear stochastic system with nonlinearities to form an output distribution that needs to meet the safety constraints. We focus on DNNs with rectified linear units (ReLU) activation functions, as the piece-wise linear nonlinearity of ReLU have *known* integral operators which allow us to propagate a given input distribution. Specifically, as we denote in Figure 1, given an input distribution's characteristic function (CF), we recover the CF of the output of a ReLU DNN with, known error accuracy, from which we can verify the output chance constraint (1). Therefore, we can provide statistical guarantees for the performance of any given ReLU neural network for a input distribution.

The paper is organized as follows. Section 2 introduces the preliminaries and problem formulation. Section 3 presents the main properties of characteristic functions we use in our work and states the main result that allows us to propagate a characteristic function through a ReLU neural network. Section 4 presents the safety verification algorithm given the machinery developed in the previous section applied to output polytopes. Examples demonstrating the theory are given in Section 5, and we provide some concluding remarks and avenues for future work in Section 6.

2. Preliminaries and Problem Formulation

2.1. Notation

Real-valued vectors are denoted by lowercase letters, $u \in \mathbb{R}^m$, matrices are denoted by uppercase letters, $V \in \mathbb{R}^{n\times m}$, and random vectors are denoted by boldface, $\mathbf{w} \in \mathbb{R}^p$. We denote the j^{th} component of a vector by the subscript u_j, and the i^{th} row and j^{th} column of a matrix by $V_{i,j}$. The imaginary unit is denoted by $\mathrm{i} := \sqrt{-1}$. A random vector $\mathbf{w}$ is defined on the probability space $(\Omega, \mathscr{B}(\Omega), \mathbb{P}_{\mathbf{w}})$ (Billingsley, 2008, Sec. 2). We only consider continuous random vectors, i.e., those having probability measure $\mathbb{P}_{\mathbf{w}}(\{\mathbf{w} \in \mathcal{S}\}) = \int_{\mathcal{S}} \psi_{\mathbf{w}}(z)\,\mathrm{d}z$ for $\mathcal{S} \subseteq \mathscr{B}(\Omega)$, and PDF $\psi_{\mathbf{w}}$ that satisfies $\psi_{\mathbf{w}} \geq 0$ almost everywhere (a.e.) such that $\int_{\mathbb{R}} \psi_{\mathbf{w}}(z)\,\mathrm{d}z = 1$. For the random variable $\mathbf{y} = a^\intercal \mathbf{w},\ a \in \mathbb{R}^p$, we characterize the probability $\mathbb{P}\{a^\intercal \mathbf{w} \leq \alpha\}$ using the cumulative distribution function (CDF) $\Phi_{a^\intercal \mathbf{w}} : \mathbb{R} \to [0,1]$, that is, by $\mathbb{P}\{a^\intercal \mathbf{w} \leq \alpha\} = \Phi_{a^\intercal \mathbf{w}}(\alpha)$ (Billingsley, 2008, Sec. 14). We write $\mathbf{w} \sim \psi_{\mathbf{w}}$ to denote the fact that $\mathbf{w}$ is distributed according to the PDF $\psi_{\mathbf{w}}$. We denote a uniform distribution as $\mathrm{U}[a,b]$ where $a, b \in \mathbb{N},\ a < b$.

2.2. Problem Formulation

We consider an L-layer ReLU DNN with input $\mathbf{x}^0 \in \mathbb{R}^{h_0}$ and output $\mathbf{y} = f(\mathbf{x}^0) = \mathbf{x}^l \in \mathbb{R}^{h_L}$, with f being the composition of L layers, that is, $f = f_{L-1} \circ \cdots \circ f_0$. The kth layer of the ReLU network corresponds to a function $f_k : \mathbb{R}^{h_k} \to \mathbb{R}^{h_{k+1}}$ of the form

$$\mathbf{x}^{k+1} = f_k(\mathbf{x}^k) = \sigma(W^k \mathbf{x}^k + b^k), \tag{2}$$

where $W^k \in \mathbb{R}^{h_{k+1}\times h_k}$ is the weight matrix, $b^k \in \mathbb{R}^{h_{k+1}}$ is the bias, and $\sigma(x_j^k) := \max(0, x_j^k)$ is the component-wise ReLU function, where x_j^k is the jth component of $x^k \in \mathbb{R}^{h_k}$. We assume that the last layer is an affine transformation, that is, $\mathbf{x}^L = W^{L-1}\mathbf{x}^{L-1} + b^{L-1}$. Note that convolution layers can be captured by this framework, as they correspond to linear layers W^k endowed with a particular matrix structure.

Let the mapping $f : \mathcal{X} \mapsto \mathcal{Y}$ with $\mathcal{X}$ and $\mathcal{Y}$ subsets of Euclidean spaces of given dimensions, and let $\mathcal{S} \subset \mathcal{Y}$ denote the output safety set. We would like to answer the following questions:

- Given a random sample from the input set $x = \mathbf{x}(\omega) \in \mathcal{X}$, where $\omega \in \Omega$, what is the probability that the output $y = f(x) \in \mathcal{Y}$ lies in the output set $\mathcal{S}$? Equivalently, given some verification threshold $p \in (0, 1]$, is the chance constraint (1) satisfied for all $x \in \mathcal{X}$?
- Given the numerically computed output distribution $\hat{\psi}_{\mathbf{y}}$, what is the relative error in the probability of satisfaction of the output chance constraint compared to that of the true output distribution $\psi_{\mathbf{y}}$?

To answer the above questions, we use the machinery of characteristic functions (CF) to propagate a distribution through a ReLU network allowing us to perform the verification task.

3. Characteristic Functions

We assume that the input distribution ψ_0 over the input set $\mathcal{X}$ is given. The analog of the probability density function $\psi_{\mathbf{x}}$ in the spatial domain is the characteristic function $\varphi_{\mathbf{x}}$ in the frequency domain.

Definition 1 (Characteristic Function) *For a continuous random vector $\mathbf{w} \in \mathbb{R}^p$ such that $\mathbf{w} \sim \psi_{\mathbf{w}}$, the characteristic function (CF) is the Fourier transform $\mathcal{F}(\psi_{\mathbf{w}})(t)$ of the PDF $\psi_{\mathbf{w}} \in \mathcal{L}^2(\mathbb{R}^p)$ given by*

$$\varphi_{\mathbf{w}}(t) := \mathbb{E}_{\mathbf{w}}\left[e^{\mathrm{i}t^\intercal \mathbf{w}}\right] = \mathcal{F}(\psi_{\mathbf{w}})(t) = \int_{\mathbb{R}^p} e^{\mathrm{i}t^\intercal z}\psi_{\mathbf{w}}(z)\,\mathrm{d}z, \tag{3}$$

where $t \in \mathbb{R}^p$.

The CF has the following properties (Cramér, 1999; Lukacs, 1970):

P1: Let $\mathbf{w}_1, \mathbf{w}_2$ be random vectors of appropriate dimensions and let $\mathbf{z} = \mathbf{w}_1 + \mathbf{w}_2$. Then, $\psi_{\mathbf{z}}(z) = \big(\psi_{\mathbf{w}_1} * \psi_{\mathbf{w}_2}\big)(z)$ (i.e., convolution of their PDFs), and $\varphi_{\mathbf{z}}(t) = \varphi_{\mathbf{w}_1}(t)\varphi_{\mathbf{w}_2}(t)$.

P2: Given $\mathbf{z} = F\mathbf{w} + g$ for $F \in \mathbb{R}^{n\times p},\ g \in \mathbb{R}^n$, the CF is $\varphi_{\mathbf{z}}(t) = e^{\mathrm{i}t^\intercal g}\varphi_{\mathbf{w}}(F^\intercal t)$.

We note that the characteristic function of a distribution always exists, even when the probability density function or moment-generating function do not exist. We can recover the CDF of a distribution from its CF using the Hilbert transform.

Definition 2 *The Hilbert transform (HT) of a function f is defined as the linear integral operator*

$$\mathcal{H}(f)(t) := \frac{1}{\pi}\,\text{p.v.}\int_{\mathbb{R}} \frac{f(\tau)}{t-\tau}\,\mathrm{d}\tau, \tag{4}$$

where p.v. *denotes the Cauchy principal value, that is,*

$$\text{p.v.}\int_{\mathbb{R}} f(t)\,\mathrm{d}t = \lim_{\epsilon\downarrow 0, a\uparrow\infty}\left[\int_{\epsilon}^{a} f(t)\,\mathrm{d}t + \int_{-a}^{-\epsilon} f(t)\,\mathrm{d}t\right]. \tag{5}$$

If not stated otherwise, all integrals in this paper are understood in the principal value sense. Using the change of variables $\tau \to -\tau$, one may equivalently express the HT as

$$\mathcal{H}(f)(t) = \frac{1}{\pi} \int_0^\infty [f(t-\tau) - f(t+\tau)] \, \frac{\mathrm{d}\tau}{\tau}. \tag{6}$$

Note that the Hilbert transform is bounded on $\mathcal{L}^2(\mathbb{R})$ (Pereyra and Ward, 2012). To numerically compute the Hilbert transform of a continuous function, we use a finite expansion of the sinc functions, which have known error bounds in the literature for some grid resolution h and number of terms N (Stenger, 2012). Appendix 1 gives a heuristic to compute the optimal values of h, N and provides error bounds in the resulting HT approximation (Pilipovsky et al., 2023). In addition, Feng and Lin (2013) show that one can recover the CDF of a distribution via a Hilbert transform, yielding similar error bounds.

Theorem 1 (Gil-Pelaez Inversion Theorem, (Feng and Lin, 2013; Gil-Pelaez, 1951)) *Given a random variable* $\mathbf{x}$ *with CF* $\varphi_{\mathbf{x}}$*, the CDF of* $\mathbf{x}$*,* $\Phi_{\mathbf{x}}(\cdot)$*, at each point of continuity* x*, can be evaluated by*

$$\Phi_{\mathbf{x}}(x) = \frac{1}{2} - \frac{\mathrm{i}}{2} \mathcal{H}(e^{-\mathrm{i}tx} \, \varphi_{\mathbf{x}}(t))(0), \tag{7}$$

where $x, t \in \mathbb{R}$.

3.1. Propagation of a Characteristic Function through a ReLU Network

Given an initial characteristic function φ^0 that represents the input distribution, we compute the output characteristic function φ^L. At an arbitrary layer k this propagation can be split into a two-step process: (i) propagate the CF through the affine layer to obtain $\varphi_{\mathbf{y}^k}$, where $\mathbf{y}^k = W^k \mathbf{x}^k + b^k$, and (ii) propagate the intermediate CF through the ReLU layer to obtain the output $\varphi_{\mathbf{x}^{k+1}}$. Using Property P2 of CFs, it is straightforward to compute

$$\varphi_{\mathbf{y}^k}(t) = \exp(\mathrm{i}t^\intercal b^k) \varphi_{\mathbf{x}^k}((W^k)^\intercal t). \tag{8}$$

Given the intermediate CF $\varphi_{\mathbf{y}^k}$, we can compute the component-wise CF after the ReLU, based on the work of Pinelis (2015), which is summarized below.

Corollary 1 *The characteristic function of the random variable* $\mathbf{x}_+ := \max(0, \mathbf{x})$ *is given by*

$$\varphi_{\mathbf{x}_+}(t) := \mathbb{E}[e^{\mathrm{i}t\mathbf{x}_+}] = \frac{1}{2} \left[1 + \varphi_{\mathbf{x}}(t)\right] + \frac{\mathrm{i}}{2} \left[\mathcal{H}(\varphi_{\mathbf{x}})(t) - \mathcal{H}(\varphi_{\mathbf{x}})(0)\right]. \tag{9}$$

Proof See Appendix 2 (Pilipovsky et al., 2023). ■

Applying the CF update (9) to each neuron j for each layer k results in the following update

$$\varphi_{\mathbf{x}_j^{k+1}}(t_j) = \frac{1}{2}(1 + \varphi_{\mathbf{y}_j^k}(t_j)) + \frac{\mathrm{i}}{2} \left[\mathcal{H}(\varphi_{\mathbf{y}_j^k})(t_j) - \mathcal{H}(\varphi_{\mathbf{y}_j^k})(0)\right], \tag{10}$$

where $t_j = e_{j,h_{k+1}}^\intercal t$ isolates the jth element of the frequency variable t.

3.2. Complexity of Propagation

Given the machinery of how to propagate CFs through a ReLU network via (8) and (10), we would like to know how many computations are actually being done per layer. The accuracy of our method is dependent on the refinement of the frequency grid for the CF evaluations and the parameters used to numerically compute the HT.

3.2.1. FREQUENCY DOMAIN GRIDDING

In order to numerically propagate the CF through the ReLU DNN, one needs to properly setup the bounds for computing the CF. Since the CF is defined for all points $t \in \mathbb{R}$, we need to setup a grid $\{t_m\}_{m=1}^N$ with some cutoffs $-\infty < d_- < d_+ < \infty$ and evaluate the CF at these grid points. As we shall see, our method is only linearly complex in the total number of grid points used to compute the CF. Given that the CF reduces down to zero at its tails, we can heuristically find the cutoff points with a convergence-type condition of the form $d_- := \text{argmax}_t |\varphi(t) - \varphi(t-\epsilon)| \leq \epsilon$ and $d_+ := \text{argmin}_t |\varphi(t+\epsilon) - \varphi(t)| \leq \epsilon$.

3.2.2. AFFINE LAYER PROPAGATION

We can break up the computations between the affine and max layers to analyze the computation complexity. The propagation of the joint CF is given in (8), however in practice, we propagate each component $\varphi_k^{(j)}$ individually, then parallelize over each marginal CF. In the spacial domain, breaking up the affine layer propagation into components yields

$$\mathbf{y}_j^k = \sum_{\ell=1}^{h_k} W_{j,\ell}^k \mathbf{x}_\ell^k + b_\ell^k, \quad j \in \{1, \ldots, h_{k+1}\}. \tag{11}$$

Thus, we use property P1 and P2 of CFs to get,

$$\varphi_{\mathbf{y}_\ell^k}(t_m) = \exp(\mathrm{i} t_m b_\ell^k) \prod_{\ell=1}^{h_k} \varphi_{\mathbf{x}_\ell^k}(W_{j,\ell}^k t_m), \tag{12}$$

where $t_m \in [d_-, d_+]$ is a grid point in the frequency domain. From (12), there are $(h_k + 1)$ terms in the product for each grid point and each component, which results in a complexity of $\mathcal{O}(h_k h_{k+1} N)$. Since the number of grid points $N \gg h_k$ for all layers k, this essentially becomes $\mathcal{O}(N)$, which is *linear* in the resolution of the grid. As a result, this implies we can construct a very fine grid without major losses in computational speed.

3.2.3. MAX LAYER PROPAGATION

The propagation of the CF through the max layer requires the computation of two Hilbert transforms, as per (10), for each grid point and neuron. From Appendix A, the discrete HT requires $2M + 1$ terms in the sum, which implies a complexity of $\mathcal{O}(h_{k+1} M N) \sim \mathcal{O}(MN)$ across all neurons for one layer (Pilipovsky et al., 2023).

4. Probabilistic DNN Verification

With the developed CF machinery outlined in Section 3.1, we can verify ReLU networks to a prescribed degree of accuracy. For example, if $p = 0.05$, then a NN passes verification if *at least* 95% of the input samples belong in the desired output set $\mathcal{S}$. We presume that the output set $\mathcal{Y} \subseteq \mathbb{R}^{h_L}$ can be represented by a convex polytope, that is, an intersection of halfspaces. For notational simplicity, we consider an output set that can be written as,

$$\mathcal{S} = \{y \in \mathbb{R}^{h_l} \mid c^\intercal y \leq d\}. \tag{13}$$

and note extension to convex polytopes follows by analyzing each half-space independently.

Algorithm 1 ReLU Network Verification

Input : $\varphi_{\mathbf{x}}$, $\{c, d\}$, N, M, h, p
Output: $\hat{p}$, pass/fail

1 $\varphi_{\mathbf{x}_j^0} \leftarrow$ Compute initial CF components on grid
2 $\varphi_{\mathbf{x}_j^L} \leftarrow$ Propagate through ReLU network using (12) and (10)
3 $\varphi_{\mathbf{y}} \leftarrow$ Compute CF of output r.v. $\mathbf{y} := c^\intercal \mathbf{x}_L$
4 $\Phi_{\mathbf{y}} \leftarrow$ Compute CDF using (7)
5 $\hat{p} := \mathbb{P}(\mathbf{y} \in \mathcal{S}) = \Phi_{\mathbf{y}}(d)$
6 **if** $\hat{p} < 1 - p$ **then**
7 | Fail verification **else** Pass verification;
8 **end**

The first three parameters the algorithm accepts are the characteristic function of the input, $\varphi_{\mathbf{x}}$, and the parameters that define the half-space, c, d. The last two design choices are the HT resolution, specified by N, h, M, and the cutoff probability for verification, p. The initial CF is then propagated through the network, which yields the final CF. Since the output set is a half-space, the probability for the output $\mathbf{x}^L$ to be in the half-space is given by

$$\mathbb{P}(\mathbf{x}^L \in \mathcal{S}) = \Phi_{\mathbf{y}}(d) = \frac{1}{2} - \frac{\mathrm{i}}{2}\mathcal{H}(e^{-\mathrm{i}td}\varphi_{\mathbf{y}}(t))(0), \tag{14}$$

where $\mathbf{y} := c^\intercal \mathbf{x}^L$ and $\varphi_{\mathbf{y}}(t) = \prod_j \varphi_{\mathbf{x}_j^L}(c_j t)$. Thus, Steps 3-4 in Algorithm 1 compute the associated CF and CDF of the constraint (13). The CDF evaluated at $x = d$ represents the probability of the event $\{c^\intercal \mathbf{x} \leq d\}$; if this value is less than $1 - p$, this is below the cutoff for verification. As an example, if $\Phi_{c^\intercal \mathbf{x}^L}(d) = 0.7$ but $p = 0.1$, then only 70% of samples from the output set lie in the safety set, which is less than the cutoff of 90%; hence the verification test fails in this case.

5. Examples

We provide two examples that illustrate the proposed verification algorithm. Both examples use ReLU feedfoward neural networks from the verification literature. All simulations were run on a 32 GB Intel i7-10750H @ 2.60 GHz computer. For computations and memory storage, we use python with JAX (Bradbury et al., 2018). JAX was run on CPU-only mode but can be run on GPUs or TPUs. All trials of the verification algorithm were compared to an empirical truth computed by brute-force propagation of 10^4 samples through the ReLU networks for each example.

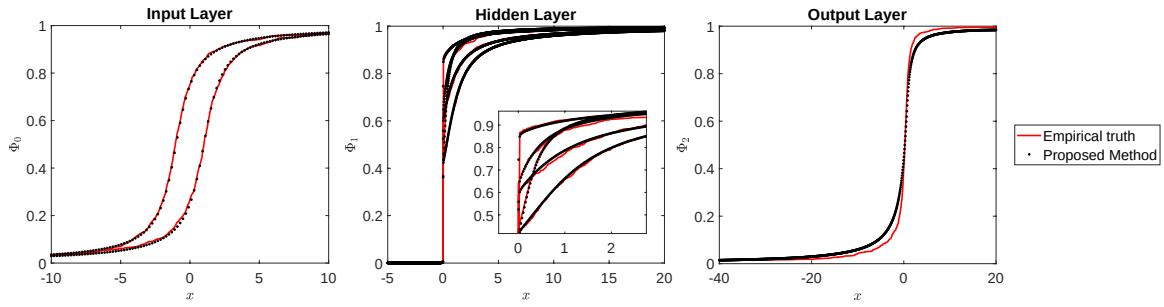


Figure 2: The characteristic function and CDF for each layer in the ReLU network. The CDF computed using the proposed method (black dot) using (7) closely resembles the empirical CDF computed from brute-force propagation of 10^4 input samples (red line).

5.1. Example 1

We present our verification method and its advantages via the following scenario, adapted from Brown et al. (2022), and compare against the bounds from a scenario-based approach in Anderson and Sojoudi (2022). We run a set of 1000 trials, where we sample from $\mathrm{U}[-1, 1]$ for the weights and biases of the ReLU network. The network architecture has two inputs, one output, and one hidden layer with 10 neurons. The output safety set is $\mathcal{S} = \{x_L : x_L \geq 0\}$. The maximum probability of lying outside the safety set is $p = 0.05$. Lastly, we also generated (an approximation of) the true output set $\mathcal{Y}$ by propagating 1 million samples from the input set through the network. We model the inputs of the network as a Cauchy distribution,

$$\varphi_0(t) = \exp(x_0 \mathrm{i} t - \gamma|t|), \tag{15}$$

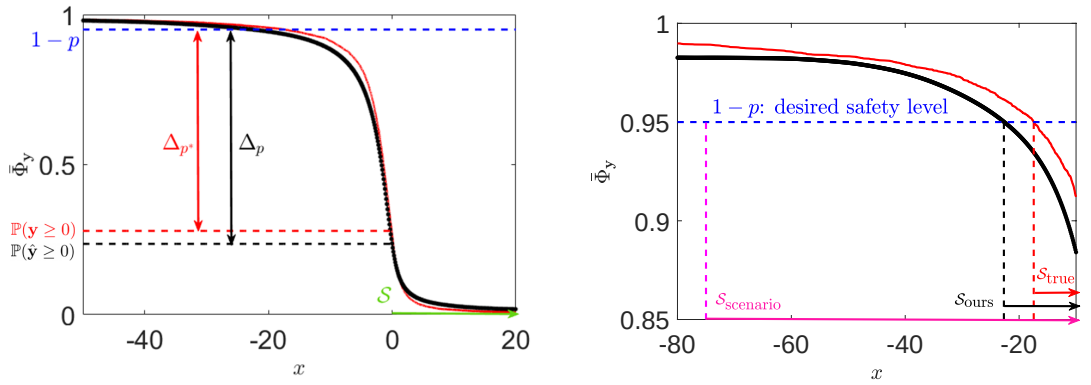
with locations $x_0^{(1)} = 1, x_0^{(2)} = -1$ and scale $\gamma^{(1)} = \gamma^{(2)} = 1$. The CF of a distribution allows one to easily compute its moments from the derivatives of the CF via the following expression

$$\mathbb{E}[\mathbf{x}^k] = \mathrm{i}^{-k} \varphi_{\mathbf{x}}^{(k)}(0), \tag{16}$$

where $f^{(k)}$ denotes the kth derivative of the function f. Since the derivatives of the Cauchy CF do *not* exist at zero, this distribution has undefined moments and moment generating function. As a result, the methods proposed in Pautov et al. (2022); Fazlyab et al. (2019); Weng et al. (2019) would not work in this case.

To show how the distribution of the inputs propagates throughout the ReLU network, we take a snapshot of the CFs and CDFs for a few random trials. The plots in Figures 2-3 correspond to the parameters $\{N, h, M, d\} = \{10000, 0.05, 5000, 50\}$, namely, 10001 terms in the HT for each of the 10^4 grid points in the domain $\mathcal{D} = \{t : t \in [-50, 50]\}$. Figure 2 shows the CDF at each layer in the network, as computed from the CF through the HT. It closely resembles the *ground-truth* CDF computed via sampling.

The accuracy of this propagation depends on the grid resolution in the frequency domain and the numerical accuracy of the HT used to propagate the CF through the max layer. A finer grid in the frequency domain with a large number of terms in the HT summation yields better results than a coarser grid with fewer terms in the summation. To illustrate this, the fourth column in Table 1 in



(*a*) For this random trial, our method accurately determines the violation of the output safety set. Our results (black) are very close ($|\delta\Delta| = 0.049$) to the empirically obtained CDF and likelihood (red).

(*b*) Our estimated safety set at the desired probability threshold (black) is much closer to the empirically determined safety set (red) as compared to other SoTA methods (magenta).

Figure 3: Comparison of ReLU network safety verification.

Appendix D computes the average error in probability across all trials for various values of the grid resolution and HT parameters (Pilipovsky et al., 2023). The parameter Δ represents the difference in the computed probability of success with the given probability threshold, i.e.,

$$\Delta := \bar{\Phi}_{\mathbf{y}}(0) - (1-p), \tag{17}$$

where $\bar{\Phi}_{\mathbf{y}}(0) := \mathbb{P}(\mathbf{x}^L > 0) = 1 - \Phi_{\mathbf{x}^L}(0)$ is known as the *complementary* CDF. See Figure 3 for a visual representation of these differences. Thus, the difference in these deltas is a metric for how accurate the CF propagation is — if the numerics were exact ($M, L \to \infty$), then $\Delta_{p^*} = \Delta_p$. Note that the trial for Figure 3 fails verification because $\Delta < 0$, which implies that the probability of being in the safety set is *less* than $1 - p = 0.95$.

For the trial in Figure 3(a), the estimated probability of being in the safety set is approximately 23.6%, whereas the true probability is 27.9%, giving $|\delta\Delta_p| = 4.3\%$. In comparison, (Anderson and Sojoudi, 2022, Appendix D), uses scenario optimization to solve the reverse problem; namely, that of finding the maximal safety set $\mathcal{X} = \bar{r}(p) = \sup_r\{r \in \mathbb{R} : \mathbb{P}(\mathbf{y} > r) \geq 1 - p\}$. Running the method by choosing samples according to $N \geq \frac{2}{\epsilon}(\log(\frac{1}{\delta}) + 1)$ where δ is a confidence parameter such that $\mathbb{P}_{\tilde{r}}\{\mathbb{P}(\mathbf{y} > r) \geq 1-p\} \geq 1-\delta$. With $\delta = 10^{-5}$, we require 501 samples. Over 500 trials, we generated 501 samples and propagated them through the network. The best $\tilde{r} = -74.99$ with the average over 500 trials is $\mathbb{E}[\tilde{r}] = -3297.92$, whereas the *true* 95% quantile occurs at $x^* = -17.43$ while our estimated quantile is at $x = -22.67$. We mark the quantile values with vertical lines on Figure 3(b). The Cauchy distribution has a longer tail than the normal distribution, thus sampling from it produces more outliers. This does not bode well for sampling-based verification methods, potentially causing large over-approximations of the safety set.

Table 1 in Appendix D shows the average time it takes to complete verification for one trial for various parameters (Pilipovsky et al., 2023). For a grid resolution of 10^4 points and 10^3 HT computations per grid point, we can get verification results in approximately 3 seconds for a two-layer network. Naturally, the accuracy of the propagation degrades with lower values for the parameters, but the computation time decreases, so there is a trade-off between accuracy and speed.

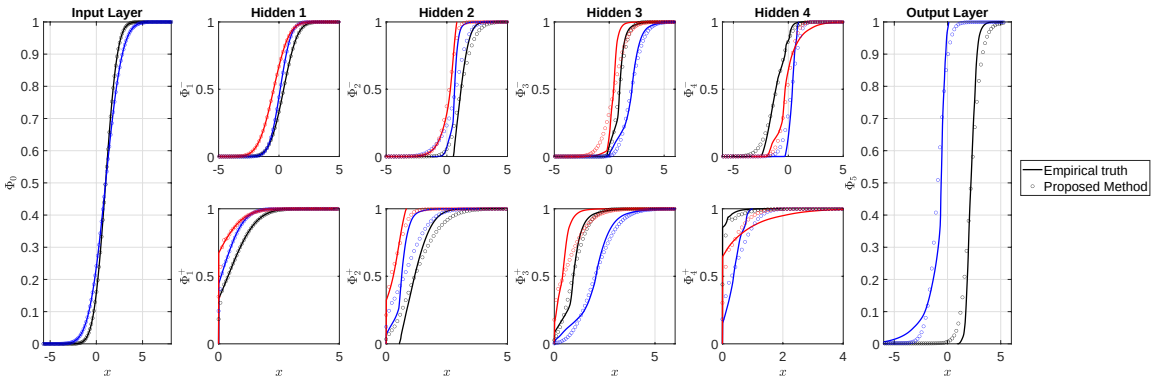


Figure 4: Comparison of empirical truth and estimated CDFs for each layer of the ReLU network where only the CDF of the first three neurons are plotted before activation, φ^-, and after activation, φ^+. The CDF we compute from the CF via (7) (circles) closely matches the empirically calculated CDF from the propagation of 10^4 samples (solid lines) for a 50 neuron hidden layer deep network.

5.2. Example 2

We also consider a more complex network based on (Fazlyab et al., 2019). In this example, we have 2 inputs, 5 hidden layers, 50 neurons in each of the 5 hidden layers, and 2 outputs. The inputs are normally distributed with mean $\mu_0 = [1, 1]^\intercal$ and covariance $\Sigma = \text{diag}(1, 2)$. We randomly choose weights and biases from $\text{U}[-1, 1]$. The propagation uses $d = 20$ for the frequency cutoffs, $N = 10^4$ grid points for the frequency resolution, and $h = 0.5$ and $M = 5000$ for the HT computations.

The verification algorithm takes approximately 15.28 sec to complete for the given propagation parameters. Figure 4 shows the evolution of the CDFs of each marginal distribution along the network. The labels $\varphi^{+/-}$ denote the CDF before and after the ReLU activation layer. We see that the CF propagation produces a relatively accurate CDF throughout the whole network given the resolution in the CF and HT. The inaccuracies result from the evaluation of the CDF at $x = 0$ as can be first seen in Φ_1^+. The sinc method that we use to compute the HT and CDF does not perform very well at discontinuity points and these errors propagate after each max layer (Feng and Lin, 2013).

6. Conclusion and Future Work

We have presented a probabilistic verification scheme for ReLU neural networks using the machinery of characteristic functions. We show that our method has a clear representation of distribution propagation through a ReLU feedforward (deep) neural network and verification becomes a evaluation of the CDF from the network's output characteristic function. One extension of this work could be to optimize the risk level by minimizing p such that $\mathbb{P}(f(\mathbf{x}^0) \in \mathcal{S}) \geq 1 - p$, for some input distribution $\mathbf{x}^0 \sim \psi^0$. Moreover, we can consider the reverse problem of finding the *largest* input set $\mathcal{X}$ such that a network is probabilistically safe for a given risk level p (Anderson and Sojoudi, 2022; Weng et al., 2019). Lastly, it might be possible to extend this framework to other activation functions, as long as one can analytically propagate the CF through that activation function.

Acknowledgments

We thank Brendon G. Anderson for providing us with the code of (Anderson and Sojoudi, 2022). This work has been supported in part by the National Science Foundation under award CNS-1836900 and by NASA under the University Leadership Initiative award 80NSSC20M0163. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or any NASA entity.

References

Brendon G. Anderson and Somayeh Sojoudi. Data-driven certification of neural networks with random input noise. *IEEE Transactions on Control of Network Systems*, pages 1–12, 2022. doi: 10.1109/TCNS.2022.3199148.

Leonard Berrada, Sumanth Dathathri, Krishnamurthy Dvijotham, Robert Stanforth, Rudy R Bunel, Jonathan Uesato, Sven Gowal, and M. Pawan Kumar. Make sure you're unsure: A framework for verifying probabilistic specifications. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 11136–11147. Curran Associates, Inc., 2021.

Patrick Billingsley. *Probability and Measure*. Wiley, 2008.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs. 2018.

Robin A. Brown, Edward Schmerling, Navid Azizan, and Marco Pavone. A unified view of SDP-based neural network verification through completely positive programming. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 9334–9355. PMLR, 28–30 Mar 2022.

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis*, pages 251–268, Cham, 2017. Springer International Publishing.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, and et al. State-of-the-art speech recognition with sequence-to-sequence models. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778, 2018.

Harald Cramér. *Mathematical Methods of Statistics*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, Princeton, 1999.

Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy R Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy S Liang, and Pushmeet Kohli. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In *Advances in Neural Information Processing Systems*, volume 33, pages 5318–5331. Curran Associates, Inc., 2020.

Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Chongli Qin, Soham De, and Pushmeet Kohli. Efficient neural network verification with exactness characterization. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 497–507, 22–25 Jul 2020.

Mahyar Fazlyab, Manfred Morari, and George J. Pappas. Probabilistic verification and reachability analysis of neural networks via semidefinite programming. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 2726–2731, 2019. doi: 10.1109/CDC40024.2019.9029310.

Mahyar Fazlyab, Manfred Morari, and George J. Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 67(1):1–15, 2022. doi: 10.1109/TAC.2020.3046193.

Liming Feng and Xiong Lin. Inverting analytic characteristic functions and financial applications. *SIAM Journal on Financial Mathematics*, 4(1):372–398, 2013.

J. Gil-Pelaez. Note on the inversion theorem. *Biometrika*, 38(3-4):481–482, 1951. doi: 10.1093/biomet/38.3-4.481.

Yu Huang and Yue Chen. Survey of state-of-art autonomous driving technologies with deep learning. In *IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 221–228, 2020. doi: 10.1109/QRS-C51114.2020.00045.

Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification*, pages 97–117. Springer, 2017.

Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward relu neural networks, 2017.

Eugene Lukacs. *Characteristic Functions*. Griffin, London, 2nd ed. edition, 1970.

Jin-Jin Mei, Yiqiu Dong, Ting-Zhu Huang, and Wotao Yin. Cauchy Noise Removal by Nonconvex ADMM with Convergence Guarantees. *Journal of Scientific Computing*, 74(2):743–766, February 2018.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94, 2017. doi: 10.1109/CVPR.2017.17.

Kumpati S. Narendra and Kannan Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990. doi: 10.1109/72.80202.

Kumpati S. Narendra and Kannan Parthasarathy. Neural networks and dynamical systems. *International Journal of Approximate Reasoning*, 6(2):109–131, 1992. ISSN 0888-613X. doi: 10.1016/0888-613X(92)90014-Q.

Mikhail Pautov, Nurislam Tursynbek, Marina Munkhoeva, Nikita Muravev, Aleksandr Petiushko, and Ivan Oseledets. Cc-cert: A probabilistic approach to certify general robustness of neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):7975–7983, Jun. 2022. doi: 10.1609/aaai.v36i7.20768.

María Cristina Pereyra and Lesley A. Ward. *Harmonic analysis*, volume 63 of *Student Mathematical Library*. American Mathematical Society, Providence, RI; Institute for Advanced Study (IAS), Princeton, NJ, 2012. doi: 10.1090/stml/063. From Fourier to wavelets, IAS/Park City Mathematical Subseries.

Joshua Pilipovsky, Vignesh Sivaramakrishnan, Meeko M. K. Oishi, and Panagiotis Tsiotras. Probabilistic verification of relu neural networks via characteristic functions. 2023. doi: 10.48550/arXiv.2212.01544.

Iosif Pinelis. Characteristic function of the positive part of a random variable and related results, with applications. *Statistics & Probability Letters*, 106:281–286, 2015. ISSN 0167-7152. doi: https://doi.org/10.1016/j.spl.2015.07.031.

Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Karsten Scheibler, Leonore Winterer, Ralf Wimmer, and Bernd Becker. Towards verification of artificial neural networks. In *MBMV*, 2015.

Federica Sciacchitano, Yiqiu Dong, and Tieyong Zeng. Variational approach for restoring blurred images with cauchy noise. *SIAM Journal on Imaging Sciences*, 8(3):1894–1922, 2015.

Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing with deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1205–1212, 2021. doi: 10.1109/IROS51168.2021.9636053.

Frank Stenger. *Numerical methods based on sinc and analytic functions*, volume 20. Springer Science & Business Media, 2012.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation,*, 23(5):828–841, 2019.

Joseph A. Vincent and Mac Schwager. Reachable polyhedral marching (rpm): A safety verification algorithm for robotic systems with deep neural network components. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9029–9035, 2021. doi: 10.1109/ICRA48506.2021.9561956.

E Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 3 2017. doi: 10.1007/s40304-017-0103-z. Dedicated to Professor Chi-Wang Shu on the occasion of his 60th birthday.

Lily Weng, Pin-Yu Chen, Lam Nguyen, Mark Squillante, Akhilan Boopathy, Ivan Oseledets, and Luca Daniel. PROVEN: Verifying robustness of neural networks with a probabilistic approach. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6727–6736, 09–15 Jun 2019.

Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th International Conference on Machine Learning, (ICML)*, volume 80, pages 5283–5292, Stockholm, Sweden, July 10-15, 2018.

Xiaofei Yang, Yunming Ye, Xutao Li, Raymond Y. K. Lau, Xiaofeng Zhang, and Xiaohui Huang. Hyperspectral image classification with deep learning models. *IEEE Transactions on Geoscience and Remote Sensing*, 56(9):5408–5423, 2018. doi: 10.1109/TGRS.2018.2815613.