Efficient Verification and Falsification of ReLU Neural Barrier Certificates

Dejin Ren[1, 2*], **Yiling Xue**[1, 2, 3*], **Taoran Wu**[1, 2], **and Bai Xue**[1, 2, 3†]

[1] Key laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Sciences, Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] School of Advanced Interdisciplinary Sciences, University of Chinese Academy of Sciences, Beijing, China
{rendj,xueyl,wutr,xuebai}@ios.ac.cn

Abstract

Barrier certificates play an important role in verifying the safety of continuous-time systems, including autonomous driving, robotic manipulators and other critical applications. Recently, ReLU neural barrier certificates—barrier certificates represented by the ReLU neural networks—have attracted significant attention in the safe control community due to their promising performance. However, because of the approximate nature of neural networks, rigorous verification methods are required to ensure the correctness of these certificates. This paper presents a necessary and sufficient condition for verifying the correctness of ReLU neural barrier certificates. The proposed condition can be encoded as either a Satisfiability Modulo Theories (SMT) or optimization problem, enabling both verification and falsification. To the best of our knowledge, this is the first approach capable of falsifying ReLU neural barrier certificates. Numerical experiments demonstrate the validity and effectiveness of the proposed method in both verifying and falsifying such certificates.

Code — https://github.com/YilingXue/evf-rnbc
Extended version — https://arxiv.org/abs/2511.10015

1 Introduction

Safety is a crucial property for continuous-time systems, including autonomous driving, robotic manipulators and other vital applications. Formally, a system is safe if every trajectory starting from the initial set never enters the unsafe set. In practice, a *barrier certificate* offers a theoretical guarantee of safety. The 0-superlevel (or sublevel) set of a barrier certificate defines a positive invariant set — meaning that trajectories starting within it remain there indefinitely. If this positive invariant set contains the initial set and does not intersect the unsafe set, the safety of the system is ensured.

In recent years, the sum-of-squares (SOS) technique has been widely used to synthesize polynomial barrier certificates for certifying positive invariance (Ames et al. 2019; Clark 2021). However, SOS methods are restricted to polynomial systems and limit the expressive power of the resulting certificates. To address these limitations, barrier certificates defined by neural networks—known as neural barrier certificates—have been introduced (Dawson, Gao, and Fan 2023; Zhao et al. 2020; Qin et al. 2021; Abate et al. 2021; Zhao et al. 2021a; Liu, Liu, and Dolan 2023). Leveraging their universal approximation capability, neural barrier certificates have demonstrated promising performance in applications such as robot control (Dawson et al. 2022; Xiao et al. 2023). Nonetheless, due to the approximate nature of neural networks, they may fail to guarantee positive invariance. Therefore, verification methods are essential to certify the correctness of learned neural barrier certificates.

This paper focuses on the verification and falsification of neural barrier certificates using Rectified Linear Unit (ReLU) activation functions, due to their widespread use in the safe control community (Dawson, Gao, and Fan 2023; Zhao et al. 2021b; Mathiesen, Calvert, and Laurenti 2022). However, ReLU neural barrier certificates are not differentiable, making traditional methods that rely on Lie derivative conditions inapplicable (Dai et al. 2017; Ames et al. 2019). Under the assumption that the derivative of the ReLU activation function is the Heaviside step function (an assumption that lacks mathematical rigor for verifying positive invariance; see the appendix for details), some works (Zhao et al. 2022; Hu et al. 2024) over-approximate the possible values of the Lie derivative and then verify the Lie derivative condition over these over-approximations, leveraging either mixed integer programming (Zhao et al. 2022) or symbolic bound propagation (Hu et al. 2024).

Recently, (Zhang et al. 2023) proposed a necessary and sufficient condition for verifying positive invariance based on the Bouligand tangent cone, which was further used to synthesize ReLU neural barrier certificates in (Zhang et al. 2024). Their condition avoids assuming that the derivative of the ReLU activation is the Heaviside step function. However, the most computationally intensive step in their method is enumerating all possible intersection combinations of linear regions intersecting the boundary of the 0-superlevel set of the ReLU neural barrier certificate. This enumeration procedure has exponential complexity with respect to the number of linear regions containing boundary points (see Remark 2). Additionally, the Bouligand tangent cone condition cannot be encoded exactly as optimization problems due to the presence of strict inequalities (see Remark 4). As a result, the condition must be relaxed to a sufficient one to enable its

*These authors contributed equally.
†Corresponding author

incorporation into optimization formulations.

In summary, all existing methods for verifying ReLU neural barrier certificates rely on derivative assumption or sufficient conditions, which can be overly conservative and lead to false negatives in practical applications. Moreover, none of these methods can be used for falsification, leaving a critical gap in real-world deployment. In this paper, we propose a novel necessary and sufficient condition for certifying the positive invariance of 0-superlevel sets of continuous piecewise linear functions (CPLFs) under continuous-time systems. This condition is applicable not only to ReLU neural barrier certificates but also to other networks with piecewise linear activation functions, such as leaky ReLU (Maas et al. 2013) and PReLU (He et al. 2015), since these networks are inherently CPLFs. Our proposed condition states that the 0-superlevel set of a CPLF is positively invariant if and only if, in each valid linear region (see Definition 5), the inner product between the region's linear coefficient vector and the vector field is non-negative. Compared to the condition in (Zhang et al. 2023), our condition requires verification in significantly fewer regions, as it avoids enumerating all possible intersection combinations of linear regions that intersect boundary. This reduction makes the proposed condition more efficient and practical for real-world applications.

We propose a verification algorithm based on our necessary and sufficient condition. The algorithm begins by identifying an initial valid linear region using the Interval Bound Propagation (IBP) technique. Once such a region is found, a boundary propagation algorithm is employed to enumerate all neighboring valid linear regions. By iteratively applying this propagation step, the algorithm ensures that all valid linear regions are covered. For each valid linear region, we can translate the proposed condition into Satisfiability Modulo Theories (SMT) and optimization problems for verifying and falsifying the ReLU neural barrier certificate. Finally, numerical experiments demonstrate the validity and effectiveness of our method in both verification and falsification. The main contributions of this paper are summarized as follows.

- We propose a necessary and sufficient condition for certifying the positive invariance of 0-superlevel sets of CPLFs under continuous-time systems. Unlike (Zhao et al. 2022; Hu et al. 2024), our condition does not rely on the assumption that the derivative of the ReLU activation is the Heaviside step function. Besides, compared to the condition in (Zhang et al. 2023), it significantly reduces the computational complexity of implementation.
- The proposed condition can be encoded as SMT and optimization problems, enabling both verification and falsification of ReLU neural barrier certificates. To the best of our knowledge, this is the first method capable of falsifying such certificates.

2 Preliminaries

2.1 Notation

$\mathbb{R}^n$ represents n-dimensional real space; $\mathbb{R}^{m\times n}$ represents space of $m \times n$ real matrices; $\mathbb{N}_{[m,n]}$ represents the non-negative integers in $[m, n]$. Vectors and matrices are denoted as boldface lowercase and uppercase respectively. For a vector $\boldsymbol{x}$, $\boldsymbol{x}(i)$ represents its i-th entry; $\|\boldsymbol{x}\|$ represents its norm. $\boldsymbol{x} \cdot \boldsymbol{y}$ represents inner product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. For a matrix $\boldsymbol{M}$, $\boldsymbol{M}(i)$ represents its i-th row vector; $\mathrm{rows}(\boldsymbol{M})$ and $\mathrm{cols}(\boldsymbol{M})$ denote the number of its rows and columns respectively; $\mathrm{rank}(\boldsymbol{M})$ represents the rank of $\boldsymbol{M}$; $[\boldsymbol{M}; \boldsymbol{x}^\top]$ represents adding $\boldsymbol{x}$ to the last row of $\boldsymbol{M}$. $\mathbf{0}$ (or $\mathbf{1}$) represents the vector (or matrix) whose entries are all zero (or one) with appropriate dimensions in the context. For a set $\mathcal{S}$, its complement, interior, closure, boundary, cardinality and power set are denoted by $\mathcal{S}^c, \mathrm{Int}\,\mathcal{S}, \overline{\mathcal{S}}, \partial\mathcal{S}, |\mathcal{S}|$ and $2^{\mathcal{S}}$, respectively. For two sets $\mathcal{S}_1, \mathcal{S}_2$, $\mathcal{S}_1 \setminus \mathcal{S}_2$ represents the set $\{s : s \in \mathcal{S}_1 \wedge s \notin \mathcal{S}_2\}$. $B(\boldsymbol{x}, \delta) = \{\boldsymbol{x}' \in \mathbb{R}^n : \|\boldsymbol{x}' - \boldsymbol{x}\| \le \delta\}$ represents the closed δ-ball around the vector $\boldsymbol{x}$.

2.2 ReLU Neural Network

We introduce notations to describe a neural network (NN) with L hidden layers, where the i-th layer contains M_i neurons. Let $\boldsymbol{x} \in \mathbb{R}^n$ denote the input to the network, z_{ij} the output of the j-th neuron in the i-th layer, and y the one-dimensional network output. We use $\boldsymbol{z}_i$ to represent the vector of neuron outputs in the i-th layer. The outputs are computed as

$$z_{ij} = \begin{cases} \sigma\left(\boldsymbol{w}_{ij}^\top \boldsymbol{x} + b_{ij}\right), & i = 1 \\ \sigma\left(\boldsymbol{w}_{ij}^\top \boldsymbol{z}_{i-1} + b_{ij}\right), & 2 \le i \le L \end{cases} \quad y = \boldsymbol{\omega}^\top \boldsymbol{z}_L + \phi$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation function. The input to σ is the pre-activation value to the neuron: for the j-th neuron in the first layer, this value is given by $\boldsymbol{w}_{1j}^\top \boldsymbol{x} + b_{1j}$; for the j-th neuron in the i-th hidden layer ($i > 1$), it is given by $\boldsymbol{w}_{ij}^\top \boldsymbol{z}_{i-1} + b_{ij}$. Here, $\boldsymbol{w}_{ij} \in \mathbb{R}^n$ when $i = 1$, and $\boldsymbol{w}_{ij} \in \mathbb{R}^{M_{i-1}}$ when $i > 1$. Throughout this paper, we assume σ is the ReLU function $\sigma(z) = \max\{0, z\}$. The final output of the network is given by $y = \boldsymbol{\omega}^\top \boldsymbol{z}_L + \phi$, where $\boldsymbol{\omega} \in \mathbb{R}^{M_L}$ and $\phi \in \mathbb{R}$. A neuron is said to be *activated* by an input $\boldsymbol{x}$ if its pre-activation value is non-negative, and *inactivated* if it is non-positive. If the pre-activation value is exactly zero, the neuron is considered both activated and inactivated.

An *activation indicator* is an L-tuple $\mathscr{C} = \langle \boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_L \rangle$, where each $\boldsymbol{s}_i = (s_{i1}, s_{i2}, \ldots, s_{iM_i})^\top$ is a binary vector of length M_i. Each entry $s_{ij} \in \{0, 1\}$ indicates whether the j-th neuron in the i-th layer is inactivated (0) or activated (1).

For a given activation indicator $\mathscr{C}$, if an input $\boldsymbol{x}$ activates $\mathscr{C}$, the pre-activation values of all neurons, as well as the overall network output, are affine functions of $\boldsymbol{x}$. The corresponding affine mapping is determined by $\mathscr{C}$ as follows. For the first layer, we define:

$$\bar{\boldsymbol{w}}_{1j}(\mathscr{C}) = \begin{cases} \boldsymbol{w}_{1j}, & s_{1j} = 1 \\ 0, & s_{1j} = 0 \end{cases} \quad \bar{b}_{1j}(\mathscr{C}) = \begin{cases} b_{1j}, & s_{1j} = 1 \\ 0, & s_{1j} = 0 \end{cases}$$

Then, the output of the j-th neuron in the first layer is given by $\bar{\boldsymbol{w}}_{1j}(\mathscr{C})^\top x + \bar{b}_{1j}(\mathscr{C})$. We recursively define $\bar{\boldsymbol{w}}_{ij}(\mathscr{C})$ and $\bar{b}_{ij}(\mathscr{C})$ for $i > 1$ by letting $\overline{\boldsymbol{W}}_i(\mathscr{C})$ be the matrix whose

columns are $\bar{\boldsymbol{w}}_{i1}(\mathscr{C}), \ldots, \bar{\boldsymbol{w}}_{iM_i}(\mathscr{C})$, and setting:

$$\bar{\boldsymbol{w}}_{ij}(\mathscr{C}) = \begin{cases} \overline{\boldsymbol{W}}_{i-1}(\mathscr{C})\boldsymbol{w}_{ij}, & s_{ij} = 1 \\ 0, & s_{ij} = 0 \end{cases}$$

$$\bar{b}_{ij}(\mathscr{C}) = \begin{cases} \boldsymbol{w}_{ij}^\top \bar{\boldsymbol{b}}_{i-1}(\mathscr{C}) + b_{ij}, & s_{ij} = 1 \\ 0, & s_{ij} = 0 \end{cases}$$

where $\bar{\boldsymbol{b}}_i(\mathscr{C})$ is the vector of bias terms $\bar{b}_{ij}(\mathscr{C})$ for $j = 1, \ldots, M_i$. We define $\boldsymbol{w}(\mathscr{C}) = \overline{\boldsymbol{W}}_L(\mathscr{C})\boldsymbol{\omega}$ and $b(\mathscr{C}) = \boldsymbol{\omega}^\top \bar{\boldsymbol{b}}_L(\mathscr{C}) + \phi$. Based on these notations, if the input $\boldsymbol{x}$ activates the activation indicator $\mathscr{C}$, the output of each neuron and the final network output are given by: $z_{ij} = \bar{\boldsymbol{w}}_{ij}(\mathscr{C})^\top \boldsymbol{x} + \bar{b}_{ij}(\mathscr{C})$ and $y = \boldsymbol{w}(\mathscr{C})^\top \boldsymbol{x} + b(\mathscr{C})$. Next, we characterize the activated region corresponding to a given activation indicator $\mathscr{C}$.

Lemma 1 ((Zhang et al. 2023)). *Let $\mathcal{X}(\mathscr{C})$ denote the set of inputs that activate a particular set of neurons represented by the activation indicator $\mathscr{C}$. For notational consistency, we define $\overline{\boldsymbol{W}}_0(\mathscr{C})$ as the identity matrix and $\bar{\boldsymbol{b}}_0(\mathscr{C})$ as the zero vector. Then*

$$\mathcal{X}(\mathscr{C}) = \bigcap_{i=1}^{L} \left(\bigcap_{j=1}^{M_i} \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{w}_{ij}^\top \left(\overline{\boldsymbol{W}}_{i-1}(\mathscr{C})^\top \boldsymbol{x} + \bar{\boldsymbol{b}}_{i-1}\right) + b_{ij} \geq 0, \boldsymbol{s}_i(j) = 1\} \cap \bigcap_{j=1}^{M_i} \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{w}_{ij}^\top \left(\overline{\boldsymbol{W}}_{i-1}(\mathscr{C})^\top \boldsymbol{x} + \bar{\boldsymbol{b}}_{i-1}\right) + b_{ij} \leq 0, \boldsymbol{s}_i(j) = 0\} \right).$$

The activated region $\mathcal{X}(\mathscr{C})$ forms a polyhedron. Let $\mathcal{I}$ denote the set of all possible activation indicators. With the above notations, the ReLU neural network can be expressed as a **continuous piecewise linear function (CPLF)**:

$$y = \boldsymbol{w}(\mathscr{C})^\top \boldsymbol{x} + b(\mathscr{C}), \boldsymbol{x} \in \mathcal{X}(\mathscr{C}), \mathscr{C} \in \mathcal{I}. \tag{1}$$

Note that $\mathcal{I}$ is a finite set, as the number of activated regions is no more than $2^{\sum_{i=1}^{L} M_i}$ (Montufar et al. 2014).

2.3 Positive Invariance and Barrier Certificate

In this paper we consider the continuous-time system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \tag{2}$$

with $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ locally Lipschitz. For any initial condition $\boldsymbol{x}_0 \in \mathbb{R}^n$, there exists a maximal time interval of existence $I(\boldsymbol{x}_0) = [0, \tau_{\max})$ such that $\boldsymbol{\phi}_{\boldsymbol{x}_0} : I(\boldsymbol{x}_0) \to \mathbb{R}^n$ is the unique solution to system (2), where $\boldsymbol{\phi}_{\boldsymbol{x}_0}(0) = \boldsymbol{x}_0$ and $\tau_{\max}$ is the explosion time with $\lim_{t \to \tau_{\max}} \|\boldsymbol{\phi}_{\boldsymbol{x}_0}(t)\| = +\infty$.

Definition 1 (Positive invariance). *A set $\mathcal{C} \subseteq \mathbb{R}^n$ is positively invariant for system (2) if for all $\boldsymbol{x}_0 \in \mathcal{C}$ and all $t \in I(\boldsymbol{x}_0)$, the corresponding trajectory satisfies $\boldsymbol{\phi}_{\boldsymbol{x}_0}(t) \in \mathcal{C}$.*

Definition 2 (Barrier certificate). *Given the system (2), let the initial set be $\mathcal{S}_I = \{\boldsymbol{x} \in \mathbb{R}^n : h_I(\boldsymbol{x}) > 0\}$ and the unsafe set be $\mathcal{S}_U = \{\boldsymbol{x} \in \mathbb{R}^n : h_U(\boldsymbol{x}) > 0\}$, where h_I, h_U are continuous functions, and both $\mathcal{S}_I, \mathcal{S}_U$ are nonempty and connected. A barrier certificate for system (2) is a continuous function $h : \mathbb{R}^n \to \mathbb{R}$ whose 0-superlevel set $\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) \geq 0\}$ satisfies the following conditions:*

1. ***Initial set condition**: $\mathcal{S}_I \subset \mathcal{C}$.*
2. ***Unsafe set condition**: $\mathcal{S}_U \cap \mathcal{C} = \emptyset$.*
3. ***Positively invariant condition**: $\mathcal{C}$ is positively invariant under system (2).*

Once a barrier certificate is found, it guarantees that all trajectories starting from the initial set will never enter the unsafe set. A **ReLU neural barrier certificate** refers to a barrier certificate represented by a neural network with ReLU activation functions. The objective of this paper is to verify or falsify a given ReLU neural barrier certificate.

Remark 1. *In fact, it suffices for a single connected component of $\mathcal{C}$ to satisfy the three conditions to ensure that all trajectories starting from the initial set will never enter the unsafe set. In the verification algorithm proposed in Sect.4, the boundary propagation algorithm is employed to identify the complete boundary of such a connected component of $\mathcal{C}$.*

3 Tangent Cones and Invariance Conditions

Since a ReLU neural network is essentially a CPLF, this section investigates the necessary and sufficient conditions for the positively invariant condition to the 0-superlevel set defined by a CPLF. We begin by reviewing some significant theorems about positive invariance and related foundational concepts.

Definition 3 (Distance). *Given a vector space X with norm $\|\cdot\|$, the distance between two points $\boldsymbol{x}_1, \boldsymbol{x}_2 \in X$ is $d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$; the distance between a set $\mathcal{S} \subset X$ and a point $\boldsymbol{x} \in X$ is $d(\boldsymbol{x}, \mathcal{S}) = \inf_{\boldsymbol{y} \in \mathcal{S}} \|\boldsymbol{x} - \boldsymbol{y}\|$.*

Definition 4 (Tangent cones (Clarke et al. 2008; Aubin and Frankowska 2009)). *Let $\mathcal{S}$ be a closed subset of the Banach space X.*

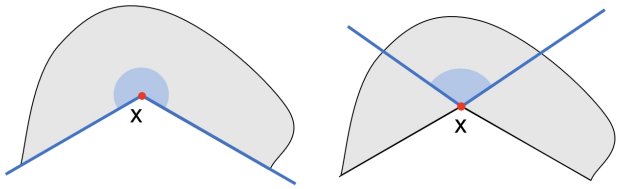
1. *The Bouligand tangent cone or contingent cone to $\mathcal{S}$ at $\boldsymbol{x}$, denoted $T_{\mathcal{S}}^B(\boldsymbol{x})$, is defined as follows:*

$$T_{\mathcal{S}}^B(\boldsymbol{x}) \triangleq \left\{ \boldsymbol{v} \in X \;\middle|\; \liminf_{t \to 0^+} \frac{d(\boldsymbol{x} + t\boldsymbol{v}, \mathcal{S})}{t} = 0 \right\}.$$

2. *The Clarke tangent cone or circatangent cone to $\mathcal{S}$ at $\boldsymbol{x}$, denoted $T_{\mathcal{S}}^C(\boldsymbol{x})$, is defined as follows:*

$$T_{\mathcal{S}}^C(\boldsymbol{x}) \triangleq \left\{ \boldsymbol{v} \in X \;\middle|\; \lim_{t \to 0^+, \boldsymbol{x}' \xrightarrow{\mathcal{S}} \boldsymbol{x}} \frac{d(\boldsymbol{x}' + t\boldsymbol{v}, \mathcal{S})}{t} = 0 \right\},$$

where $\boldsymbol{x}' \xrightarrow{\mathcal{S}} \boldsymbol{x}$ means the convergence is in $\mathcal{S}$.



(a) Bouligand tangent cone (b) Clarke tangent cone

Figure 1: Illustration of tangent cones at a nonsmooth boundary point.

Both tangent cones in Definition 4 are closed cones. If $\boldsymbol{x} \in \operatorname{Int} \mathcal{S}$, then $T_{\mathcal{S}}^B(\boldsymbol{x}) = T_{\mathcal{S}}^C(\boldsymbol{x}) = X$. If $\boldsymbol{x} \in \mathcal{S}^c$, then $T_{\mathcal{S}}^B(\boldsymbol{x}) = T_{\mathcal{S}}^C(\boldsymbol{x}) = \emptyset$. Therefore, tangent cones are nontrivial only on the boundary $\partial\mathcal{S}$.

The following theorem is quoted from (Clarke et al. 2008) with some modifications for the context of ordinary differential equations instead of differential inclusions.

Theorem 1 ((Clarke et al. 2008, Theorem 3.8 in Chapter 4))**.** *Consider the system* (2) *and let $\mathcal{S} \subset \mathbb{R}^n$ be a closed set, then the following assertions are equivalent:*

a. $\mathcal{S}$ is positively invariant for the system (2)*;*

b. for all $\boldsymbol{x} \in \partial\mathcal{S}$, $\boldsymbol{f}(\boldsymbol{x}) \in T_{\mathcal{S}}^B(\boldsymbol{x})$;

c. for all $\boldsymbol{x} \in \partial\mathcal{S}$, $\boldsymbol{f}(\boldsymbol{x}) \in T_{\mathcal{S}}^C(\boldsymbol{x})$.

For a CPLF $h : \mathbb{R}^n \to \mathbb{R}$, its expression can be written as

$$h(\boldsymbol{x}) = \boldsymbol{w}_i^\top \boldsymbol{x} + b_i, \boldsymbol{x} \in \mathcal{X}_i, i = 1, \cdots, N, \tag{3}$$

where $\mathcal{X}_i = \{\boldsymbol{x} : \boldsymbol{A}_i\boldsymbol{x} \leq \boldsymbol{d}_i\}$ is a polyhedron (the background of polyhedron is provided in the appendix) of full dimension, i.e., $\dim(\mathcal{X}_i) = n$, referred to as a *linear region*. The collection $\{\mathcal{X}_i\}_{i=1}^N$ partitions the entire input space $\mathbb{R}^n$, i.e., $\bigcup_{i=1}^N \mathcal{X}_i = \mathbb{R}^n$, and for all $i \neq j$, $\dim(\mathcal{X}_i \cap \mathcal{X}_j) \leq n-1$. We say $\mathcal{X}_i$ and $\mathcal{X}_j$ are adjacent if $\dim(\mathcal{X}_i \cap \mathcal{X}_j) = n-1$. In such a case, there exists a hyperplane $\mathcal{H}$ such that $\mathcal{X}_i \cap \mathcal{X}_j \subseteq \mathcal{H}$, and $\mathcal{H} \cap \mathcal{X}_i$ and $\mathcal{H} \cap \mathcal{X}_j$ are facets of $\mathcal{X}_i$ and $\mathcal{X}_j$, respectively.

In this paper, we adopt the following assumption, which is also emphasized in (Ames et al. 2016, 2019).

Assumption 1. *For a candidate barrier certificate h, its 0-superlevel set $\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) \geq 0\}$ satisfies: $\partial\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) = 0\}$, $\operatorname{Int}\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) > 0\}$; $\mathcal{C}$ contains interior points and is a regular closed set, i.e., $\operatorname{Int}\mathcal{C} \neq \emptyset, \overline{\operatorname{Int}\mathcal{C}} = \mathcal{C}$.*

For ease of presentation, we define valid linear regions as follows.

Definition 5 (Valid linear region)**.** *A linear region $\mathcal{X}_i$ is said to be **valid** if it is n-dimensional and, for the associated hyperplane $\mathcal{H}_i = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{w}_i^\top \boldsymbol{x} + b_i = 0\}$, either $\mathcal{H}_i \cap \operatorname{Int}\mathcal{X}_i \neq \emptyset$ or $\mathcal{H}_i \cap \mathcal{X}_i$ is a facet of $\mathcal{X}_i$.*

The following two propositions respectively characterize the Bouligand tangent cone and the Clarke tangent cone to the set $\mathcal{C}$.

Proposition 1. *Given a CPLF h as defined in* (3)*, consider its 0-superlevel set $\mathcal{C} = \{\boldsymbol{z} \in \mathbb{R}^n : h(\boldsymbol{z}) \geq 0\}$. Under Assumption 1, the following assertions hold for any $\boldsymbol{x} \in \partial\mathcal{C}$:*

1. if $\boldsymbol{x} \in \operatorname{Int}\mathcal{X}_i, i = 1, \cdots, N$, then

$$T_{\mathcal{C}}^B(\boldsymbol{x}) = \{\boldsymbol{v} \in \mathbb{R}^n : \boldsymbol{w}_i^\top \boldsymbol{v} \geq 0\}; \tag{4}$$

2. if $\boldsymbol{x} \in \bigcap_{k=1}^m \mathcal{X}_{i_k}$ and $\boldsymbol{x} \notin \overline{\mathbb{R}^n \setminus \bigcup_{k=1}^m \mathcal{X}_{i_k}}$, $i_1, \cdots, i_m \in \{1, 2, \cdots, N\}$, then

$$T_{\mathcal{C}}^B(\boldsymbol{x}) = \bigcup_{k=1}^m \{\boldsymbol{v} \in \mathbb{R}^n : \bigwedge_{j \in E} \boldsymbol{A}_{i_k}(j)\boldsymbol{v} \leq 0 \wedge \boldsymbol{w}_{i_k}^\top \boldsymbol{v} \geq 0\}, \tag{5}$$

$$T_{\mathcal{C}}^B(\boldsymbol{x}) \supset \{\boldsymbol{v} \in \mathbb{R}^n : \bigwedge_{k \in I} \boldsymbol{w}_{i_k}^\top \boldsymbol{v} \geq 0\}, \tag{6}$$

where the set E, I are defined as $E \triangleq \{j \in \{1, \cdots, \operatorname{rows}(\boldsymbol{A})\} : \boldsymbol{A}_{i_k}(j)\boldsymbol{x} = \boldsymbol{d}_{i_k}(j)\}, I \triangleq \{k \in \{1, \cdots, m\} : \mathcal{X}_{i_k}$ is a valid linear region$\}$.

3. $\partial\mathcal{C} \subset \bigcup_{l \in J} \mathcal{X}_l$, where $J \triangleq \{l \in \{1, \cdots, N\} : \mathcal{X}_l$ is a valid linear region$\}$.

Proposition 2. *Given a CPLF h as defined in* (3)*, let $\mathcal{C} = \{\boldsymbol{z} \in \mathbb{R}^n : h(\boldsymbol{z}) \geq 0\}$ denote its 0-superlevel set. Under Assumption 1, the Clarke tangent cone to $\mathcal{C}$ at any point $\boldsymbol{x} \in \partial\mathcal{C}$ is given by*

$$T_{\mathcal{C}}^C(\boldsymbol{x}) = \begin{cases} \{\boldsymbol{v} \in \mathbb{R}^n : \boldsymbol{w}_i^\top \boldsymbol{v} \geq 0\}, \text{ if} \\ \quad \boldsymbol{x} \in \operatorname{Int}\mathcal{X}_i, i = 1, \cdots, N \\ \{\boldsymbol{v} \in \mathbb{R}^n : \bigwedge_{k \in I} \boldsymbol{w}_{i_k}^\top \boldsymbol{v} \geq 0\}, \text{ if} \\ \quad \boldsymbol{x} \in \bigcap_{k=1}^m \mathcal{X}_{i_k} \wedge \boldsymbol{x} \notin \overline{\mathbb{R}^n \setminus \bigcup_{k=1}^m \mathcal{X}_{i_k}} \end{cases} \tag{7}$$

where $i_1, \cdots, i_m \in \{1, 2, \cdots, N\}, I = \{k \in \{1, \cdots, m\} : \mathcal{X}_{i_k}$ is a valid linear region$\}$.

Based on the equivalence of assertions (a) and (c) in Theorem 1, the necessary and sufficient condition for the positive invariance of the 0-superlevel set of a CPLF under system (2) is established in Theorem 2.

Theorem 2. *Consider the system* (2) *and the set $\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) \geq 0\}$, where h is a CPLF as defined in* (3)*. Under Assumption 1, the set $\mathcal{C}$ is positively invariant for the system* (2) *if and only if, for each valid linear region $\mathcal{X}_i$,*

$$\boldsymbol{w}_i^\top \boldsymbol{f}(\boldsymbol{x}) \geq 0, \forall \boldsymbol{x} \in \partial\mathcal{C} \cap \mathcal{X}_i. \tag{8}$$

Recall that a ReLU neural network is essentially a CPLF. By applying Theorem 2 and using the notations introduced in Section 2.2, we derive the necessary and sufficient condition for the positive invariance of the 0-superlevel set represented by a ReLU neural network under system (2).

Theorem 3. *Consider the system* (2) *and the set $\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) \geq 0\}$, where h is a ReLU neural network as described in Section 2.2. Under Assumption 1, the set $\mathcal{C}$ is positively invariant for system* (2) *if and only if, for each activation indicator $\mathscr{C}$ such that $\mathcal{X}(\mathscr{C})$ is a valid linear region,*

$$\boldsymbol{w}(\mathscr{C})^\top \boldsymbol{f}(\boldsymbol{x}) \geq 0, \forall \boldsymbol{x} \in \partial\mathcal{C} \cap \mathcal{X}(\mathscr{C}). \tag{9}$$

Remark 2. *In fact, with the equivalence of assertions (a) and (b) in Theorem 1, one can also derive a necessary and sufficient condition for the positive invariance of the 0-superlevel set of a ReLU neural network, as investigated in (Zhang et al. 2023). However, compared to our condition in Theorem 3, the condition in (Zhang et al. 2023) has to additionally enumerate all possible intersection combinations of activated regions $\mathcal{X}(\mathscr{C})$ that intersect the boundary $\partial\mathcal{C}$, which significantly increases computational complexity. For instance, if there are n activated regions $\mathcal{X}(\mathscr{C}_1), \cdots, \mathcal{X}(\mathscr{C}_n)$ with a nonempty intersection $\bigcap_{k=1}^n \mathcal{X}(\mathscr{C}_k) \neq \emptyset$, the total number of intersection combinations among these regions is $\sum_{k=1}^n \binom{n}{k} = 2^n - 1$. However, according to Definition 5, the number of valid linear regions that must be enumerated in our method is fewer than n.*

4 Verification Algorithm

In this section we present our verification algorithm based on the necessary and sufficient condition in Theorem 3. The algorithm proceeds in three steps: we first search for an initial valid linear region (Sect. 4.1); using this initial region, the boundary propagation algorithm enumerates all valid linear regions, i.e., compute $\mathcal{A} \triangleq \{\mathscr{C} : \mathcal{X}(\mathscr{C}) \text{ is a valid linear region}\}$ (Sect. 4.2); finally, for each valid activation indicator $\mathscr{C} \in \mathcal{A}$, we verify whether $\boldsymbol{w}(\mathscr{C})^\top \boldsymbol{f}(\boldsymbol{x}) \geq 0$ holds for all $\boldsymbol{x} \in \partial\mathcal{C} \cap \mathcal{X}(\mathscr{C})$ (Sect. 4.3). If this condition holds for every $\mathscr{C} \in \mathcal{A}$, then $\mathcal{C}$ is positively invariant for system (2); otherwise, if any $\mathscr{C}' \in \mathcal{A}$ violates the condition, $\mathcal{C}$ is not positively invariant. Figure 2 illustrates the procedure of enumerating all valid linear regions.

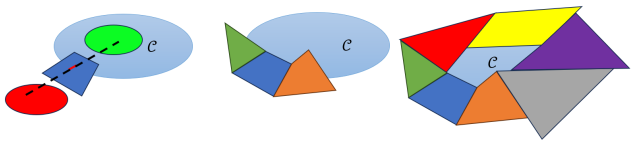


Figure 2: Illustration of enumerating valid linear regions: We begin by randomly selecting two points—one from the initial set (colored in green) and one from the unsafe set (colored in red)—and iteratively shrinking the segment connecting them until an initial valid linear region is found. The boundary propagation algorithm then expands from this region to identify neighboring valid linear regions. By iteratively applying this propagation step, the algorithm eventually enumerates all valid linear regions.

4.1 Searching for an Initial Valid Linear Region

We use the following procedure to search for an initial valid linear region.

1. Randomly select points until two points $\boldsymbol{x}_1, \boldsymbol{x}_2$ are found such that $h(\boldsymbol{x}_1)h(\boldsymbol{x}_2) < 0$. For example, $\boldsymbol{x}_1$ can be sampled from the unsafe set and $\boldsymbol{x}_2$ from the initial set.
2. Utilize the method of the bisection to shrink the length of the line segment $\{t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2 : 0 \leq t \leq 1\}$, until $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \leq \epsilon$, where $\epsilon > 0$ is a predefined threshold. Specifically, suppose initially $h(\boldsymbol{x}_1) < 0, h(\boldsymbol{x}_2) > 0$, if $h(\frac{\boldsymbol{x}_1+\boldsymbol{x}_2}{2}) < 0$, then $\boldsymbol{x}_1 := \frac{\boldsymbol{x}_1+\boldsymbol{x}_2}{2}$, else $\boldsymbol{x}_2 := \frac{\boldsymbol{x}_1+\boldsymbol{x}_2}{2}$.
3. Compute the interval hull $\mathrm{IntHull}(\{\boldsymbol{x}_1, \boldsymbol{x}_2\})$ of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and use Interval Bound Propagation (IBP) to determine the candidate activation indicator $\mathscr{C}'$. For each neuron j in layer i with input interval $[l_1, l_2]$:
   - if $l_1 > 0$, set $\boldsymbol{s}_i(j) := 1$;
   - if $l_2 < 0$, set $\boldsymbol{s}_i(j) := 0$;
   - else, set $\boldsymbol{s}_i(j) := -1$.
4. Generate all feasible activation indicators $\mathscr{C}$ from $\mathscr{C}'$ by replacing each -1 in $\boldsymbol{s}_i(j)$ with both 1 and 0.
5. For each feasible activation indicator $\mathscr{C}$, check whether $\mathcal{X}(\mathscr{C})$ passes the **Valid Test** below. If a valid indicator is found, denote it by $\mathscr{C}_0$, and take the corresponding region $\mathcal{X}(\mathscr{C}_0)$ as the initial valid linear region.

Valid Test When $\mathcal{X}(\mathscr{C})$ is an n-dimensional valid linear region, it follows that $\dim(\partial\mathcal{C} \cap \mathcal{X}(\mathscr{C})) = n - 1$. Therefore, the validity of $\mathcal{X}(\mathscr{C})$ can be verified by checking whether this dimensional condition holds. Note that the redundant case where $\partial\mathcal{C} \cap \mathcal{X}(\mathscr{C}) = \mathcal{X}(\mathscr{C})$ can also be included, but this does not affect the correctness of our method.

Given an activated region of the form $\mathcal{X}(\mathscr{C}) = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{d}\}$, define the set $\mathcal{P} = \partial\mathcal{C} \cap \mathcal{X}(\mathscr{C})$ and let $\tilde{\boldsymbol{A}} = [\boldsymbol{A}; \boldsymbol{w}^\top(\mathscr{C}); -\boldsymbol{w}^\top(\mathscr{C})], \tilde{\boldsymbol{d}} = [\boldsymbol{d}; -\boldsymbol{b}(\mathscr{C}); \boldsymbol{b}(\mathscr{C})]$, then $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^n : \tilde{\boldsymbol{A}}\boldsymbol{x} \leq \tilde{\boldsymbol{d}}\}$ is also a polyhedron. In a system of linear inequalities $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{d}$, the inequality $\boldsymbol{A}(j)\boldsymbol{x} \leq \boldsymbol{d}(j), j \in \{1, \cdots, \mathrm{rows}(\boldsymbol{A})\}$, is called an *implicit equality* if $\boldsymbol{A}(j)\boldsymbol{x} = \boldsymbol{d}(j)$ holds for all solutions of the system $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{d}$. We denote by $\boldsymbol{A}^=\boldsymbol{x} \leq \boldsymbol{d}^=$ the system consisting of all implicit equalities within $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{d}$.

To identify the implicit equalities in $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^n : \tilde{\boldsymbol{A}}\boldsymbol{x} \leq \tilde{\boldsymbol{d}}\}$, we solve the following pair of LPs for each row $j = 1, \cdots, \mathrm{rows}(\tilde{\boldsymbol{A}})$:

$$\begin{aligned} &\min_{\boldsymbol{x}} \tilde{\boldsymbol{A}}(j)\boldsymbol{x} && \max_{\boldsymbol{x}} \tilde{\boldsymbol{A}}(j)\boldsymbol{x} \\ &\text{s.t. } \tilde{\boldsymbol{A}}\boldsymbol{x} \leq \tilde{\boldsymbol{d}} && \text{s.t. } \tilde{\boldsymbol{A}}\boldsymbol{x} \leq \tilde{\boldsymbol{d}} \end{aligned}$$

If the optimal values of both LPs are equal, then $\tilde{\boldsymbol{A}}(j)\boldsymbol{x} \leq \tilde{\boldsymbol{d}}(j)$ is an implicit equality. According to (Conforti, Cornuéjols, and Zambelli 2014, Theorem 4.17), the dimension of $\mathcal{P}$ satisfies $\dim(\mathcal{P}) = n - \mathrm{rank}(\tilde{\boldsymbol{A}}^=)$. Therefore, if $\mathrm{rank}(\tilde{\boldsymbol{A}}^=) = 1$, then $\dim(\mathcal{P}) = n - 1$.

4.2 Boundary Propagation Algorithm

Next we introduce the boundary propagation algorithm, which can enumerate all valid activation indicators, i.e., the set $\mathcal{A} \triangleq \{\mathscr{C} : \mathcal{X}(\mathscr{C}) \text{ is a valid linear region}\}$. This algorithm proceeds as follows:

1. Initialize the set of valid activation indicators as $\mathcal{A} := \{\mathscr{C}_0\}$ and the visited set as $\mathcal{B} := \emptyset$.
2. Select an activation indicator $\mathscr{C}$ in $\mathcal{A} \setminus \mathcal{B}$ and add $\mathscr{C}$ to $\mathcal{B}$. For each $j = 1, \cdots, \mathrm{rows}(\boldsymbol{A})$, solve the following LP feasibility problem to find the facet of $\mathcal{X}(\mathscr{C})$ which has the intersection with $\partial\mathcal{C}$.

$$\begin{cases} \boldsymbol{w}(\mathscr{C})^\top \boldsymbol{x} + b(\mathscr{C}) = 0 \\ \boldsymbol{A}(j)\boldsymbol{x} = \boldsymbol{d}(j) \\ \boldsymbol{A}(k)\boldsymbol{x} \leq \boldsymbol{d}(k), k \in \{1, \cdots, \mathrm{rows}(\boldsymbol{A})\} \setminus \{j\} \end{cases}$$

   If the above LP feasibility problem admits a feasible solution $\boldsymbol{x}^\star$, then enumerate all feasible activation indicators that are activated by $\boldsymbol{x}^\star$. For each indicator that passes the **Valid Test**, add it to the set $\mathcal{A}$.
3. The algorithm terminates when $\mathcal{A} = \mathcal{B}$, meaning all valid activation indicators have been explored and no new ones have been found.

Remark 3. *When the system dimension is 2, the number of facets of $\mathcal{X}(\mathscr{C})$ that intersect $\partial\mathcal{C}$ is at most 4. Therefore once four such facets have been identified, it is unnecessary to examine the remaining facets.*

For ReLU neural networks, the partitioning of the input space exhibits a special structure: every n-dimensional linear region shares a facet with each of its adjacent linear regions. This property arises from the fact that the partitioning is generated through a sequence of hyperplanes, layer by layer (Raghu et al. 2017; Serra, Tjandraatmadja, and Ramalingam 2018). Moreover, since $\partial\mathcal{C}$ is contained within the union of valid linear regions (see Proposition 1), if we further assume that $\partial\mathcal{C}$ is connected, then given an initial valid linear region, one can examine its facets to identify those intersecting the boundary. By recursively expanding to adjacent regions via these shared facets, we can enumerate all valid linear regions. The following theorem formalizes this claim.

Theorem 4. *Assuming that $\partial\mathcal{C}$ is connected, the boundary propagation algorithm can identify all valid linear regions.*

4.3 Verification and Falsification in Each Valid Linear Region

After enumerating all valid linear regions, we can verify or falsify the three conditions in Definition 2 within each valid linear region.

the Positively Invariant Condition The necessary and sufficient condition in Theorem 3 can be described as a quantified formula:

$$\forall \mathscr{C} \in \mathcal{A}, \forall \boldsymbol{x} \ \left(\boldsymbol{x} \in \partial\mathcal{C} \cap \mathcal{X}(\mathscr{C}) \rightarrow \boldsymbol{w}(\mathscr{C})^\top \boldsymbol{f}(\boldsymbol{x}) \geq 0\right). \tag{10}$$

We can translate (10) into an SMT problem that determines whether the following quantifier-free formula with Disjunctive Normal Form (DNF) is satisfiable (SAT) or not (UNSAT).

$$\begin{aligned} \bigvee_{\mathscr{C} \in \mathcal{A}} \big(&\boldsymbol{w}(\mathscr{C})^\top \boldsymbol{x} + b(\mathscr{C}) = 0 \wedge \boldsymbol{x} \in \mathcal{X}(\mathscr{C}) \\ &\wedge \boldsymbol{w}(\mathscr{C})^\top \boldsymbol{f}(\boldsymbol{x}) < 0\big). \end{aligned} \tag{11}$$

If the SMT problem (11) is proven to be UNSAT, then the Boolean value of (10) is true, implying that $\mathcal{C}$ is positively invariant under system (2). When the system (2) is polynomial, the SMT (11) can be solved using an SMT solver that employs Cylindrical Algebraic Decomposition (Caviness and Johnson 2012), such as Z3 (De Moura and Bjørner 2008). In this case, verification by solving (11) is both sound and complete: an UNSAT return implies positive invariance of $\mathcal{C}$, while a SAT return indicates the falsification of positive invariance. When the system (2) is non-polynomial, we can use dReal (Gao, Kong, and Clarke 2013), an SMT solver that supports non-polynomial functions such as trigonometric and exponential functions. dReal employs δ-complete decision procedures, returning either UNSAT or δ-SAT, where δ is a user-specified numerical error bound. Therefore, positive invariance verification using dReal is sound: when dReal returns UNSAT for SMT (11), $\mathcal{C}$ is positively invariant. However, a "δ-SAT" result does not necessarily falsify positive invariance due to potential numerical errors.

The quantified formula (10) can also be described as a series of optimization problems: for each valid linear region $\mathcal{X}(\mathscr{C}), \mathscr{C} \in \mathcal{A}$, we solve the following optimization.

$$\begin{aligned} &\min_{\boldsymbol{x}} \boldsymbol{w}(\mathscr{C})^\top \boldsymbol{f}(\boldsymbol{x}) \\ &\text{s.t.} \begin{cases} \boldsymbol{w}(\mathscr{C})^\top \boldsymbol{x} + b(\mathscr{C}) = 0, \\ \boldsymbol{x} \in \mathcal{X}(\mathscr{C}). \end{cases} \end{aligned} \tag{12}$$

If the optimal value of (12) is non-negative, the positive invariant condition is satisfied for $\mathscr{C}$. When all activation indicators in $\mathcal{A}$ pass the verification, we can deduce that $\mathcal{C}$ is positively invariant for system (2); otherwise, it is not. When the system (2) is linear, the optimization is an LP. If the optimal value of (12) exists, it can be obtained by the simplex method or interior-point method. In contrast, when the system (2) is nonlinear, the optimization (12) may be nonconvex, making it challenging to obtain the optimal value. Nevertheless, we can utilize (12) to falsify the positive invariance. Specifically, if there exists a valid linear region $\mathcal{X}(\mathscr{C}), \mathscr{C} \in \mathcal{A}$ such that (12) yields a negative feasible value, then $\mathcal{C}$ is not positively invariant.

Remark 4. *The condition based on the Bouligand tangent cone cannot be directly translated into a series of optimization problems, in contrast to condition derived from the Clarke tangent cone. This limitation arises from the presence of strict inequalities — specifically, the restriction "for $\boldsymbol{x} \in \operatorname{Int}\mathcal{X}_i$ ($\boldsymbol{A}_i\boldsymbol{x} < \boldsymbol{d}_i$)" in the first assertion of Proposition 1 — which are not permitted in standard optimization formulations. The detailed explanation can be found in the appendix.*

the Initial and Unsafe Set Conditions We can also encode the verification of the initial and unsafe set conditions as SMT and optimization problems like the positively invariant condition. To this end, we introduce the following propositions, which provide necessary and sufficient conditions.

Proposition 3. *$\mathcal{S}_I \subset \mathcal{C}$ if and only if $\mathcal{S}_I \cap \partial\mathcal{C} = \emptyset$ and $\mathcal{S}_I \cap \operatorname{Int}\mathcal{C} \neq \emptyset$.*

Proposition 4. *$\mathcal{S}_U \cap \mathcal{C} = \emptyset$ if and only if $\mathcal{S}_U \cap \partial\mathcal{C} = \emptyset$ and $\mathcal{S}_U \cap \mathcal{C}^c \neq \emptyset$.*

The condition $\mathcal{S}_I \cap \partial\mathcal{C} = \emptyset$ can also be described as a quantified formula:

$$\forall \mathscr{C} \in \mathcal{A}, \forall \boldsymbol{x} \ \left(\boldsymbol{x} \in \partial\mathcal{C} \cap \mathcal{X}(\mathscr{C}) \rightarrow h_I(\boldsymbol{x}) \leq 0\right). \tag{13}$$

Similar to the positive invariant condition, the quantified formula (13) can be encoded into SMT and optimization problems. For simplicity, we present the details in the appendix.

To verify $\mathcal{S}_I \cap \operatorname{Int}\mathcal{C} \neq \emptyset$, we can randomly select a point $\boldsymbol{x} \in \mathcal{S}_I$ and check whether $h_I(\boldsymbol{x}) > 0$. If both conditions are satisfied, we conclude that $\mathcal{S}_I \subset \mathcal{C}$. Otherwise, the verification fails.

Note that the condition in Proposition 4 for verifying the unsafe set condition $\mathcal{S}_U \cap \mathcal{C} = \emptyset$ is analogous to that used for verifying the initial set condition. Therefore, a similar verification procedure can be applied to the unsafe set condition as well.

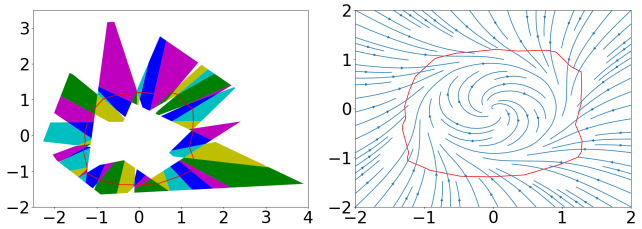
5 Experiments

In this section, we evaluate the proposed method to demonstrate the validity and effectiveness of our verification algorithm. Specifically, we conduct experiments on four systems: Arch3, Complex, Linear4d, and Decay. Among them, Linear4d is a linear system, while the others are nonlinear. System descriptions, experimental settings, and detailed experimental results are provided in the appendix.

For linear systems, we only solve the optimization problem (12) to check the positive invariant condition, as it reduces to an LP whose optimal value can be reliably obtained when it exists. For nonlinear systems, we first solve the optimization (12). If there exists a valid linear region for which the suboptimal value is negative, then the positive invariant condition is falsified. Otherwise, if all instances of (12) yield nonnegative suboptimal values for each $\mathscr{C} \in \mathcal{A}$, we proceed to solve the SMT problem (11) using the dReal solver (Gao, Kong, and Clarke 2013). If dReal returns UNSAT for every region, positive invariance is verified. However, if it returns δ-SAT for any region, the validity of the certificate remains inconclusive. The same procedure is applied to verify the initial set and unsafe set conditions.

We compare our method with that in (Zhang et al. 2023), as it is, to the best of our knowledge, the only method that does not rely on the Heaviside step function assumption for the derivative of the ReLU activation. Notably, our method supports both verification and falsification of barrier certificates, whereas the method in (Zhang et al. 2023) only performs verification. Consequently, if a certificate fails their test, no conclusion can be drawn about its correctness. Furthermore, the correctness guarantees in (Zhang et al. 2023) hold only if the solver returns optimal solutions for all underlying optimization problems. In practice, however, solvers often yield suboptimal results in nonlinear programming, potentially undermining the reliability of their verification.

The running time and validity of barrier certificates for each case are summarized in Table 1. Our method successfully verifies or falsifies all cases, whereas the approach in (Zhang et al. 2023) can only verify five out of seven barrier certificates in the two-dimensional system Arch3. In other cases, it either returns "unknown" or crashes due to memory exhaustion. Furthermore, their results are reliable only when all underlying optimization problems can be solved to optimality by the solvers, which is difficult to rigorously verify in nonlinear programming. In terms of running time, our method consistently outperforms theirs, particularly in high-dimensional systems. This improvement stems from the fact that our algorithm enumerates far fewer regions than (Zhang et al. 2023), as discussed in Remark 2. Figure 3 illustrates the valid linear regions enumerated by our boundary propagation algorithm for the first barrier certificate in system Arch3.



(a) (b)

Figure 3: Enumerated valid linear regions and the vector field of system Arch3.

Case	HNN	t_{our}	t_Z	V_{our}	V_Z
Arch3 (2-d) (nonlinear)	32	3.89	24.10	✓	✓
	64	28.40	141.01	✓	✓
	96	94.75	528.36	✓	✓
	128	204.78	198.17	✗	?
	32-32	19.75	41.10	✓	✓
	64-64	215.99	283.00	✓	✓
	96-96	774.27	1644.60	✗	?
Complex (3-d) (nonlinear)	32	5.29	135.22	✓	?
	64	92.36	698.35	✓	?
	96	469.53	1377.00	✓	?
	32-32	79.57	1132.23	✓	?
	64-64	500.08	–	✗	–
	96-96	19686.48	–	✓	–
Linear4d (4-d) (linear)	16	9.61	286.46	✓	?
	32	88.12	1110.54	✓	?
	48	9992.90	–	✓	–
	8-8	7.84	212.19	✓	?
	12-12	55.05	593.78	✓	?
	16-16	524.73	2243.71	✗	?
	24-24	22571.44	–	✗	–
Decay (6-d) (nonlinear)	20	19842.22	–	✓	–
	8-8	347.72	4381.28	✓	?
	10-10	1953.08	4035.94	✗	?
	12-12	31112.56	–	✗	–

Table 1: The first two columns list the system names and neural network architectures (HNN denotes the number of neurons per hidden layer). We report the running time and certificate validity of our method as t_{our} and V_{our}, and use t_Z and V_Z for the results of (Zhang et al. 2023). A dash ($-$) indicates a program crash. A ✓ denotes verification (✓ indicates verification under the assumption that all underlying optimization problems can be solved to optimality), ✗ denotes falsification, and ? indicates unknown validity.

6 Conclusion

In this paper, we proposed a necessary and sufficient condition for verifying ReLU neural barrier certificates and developed a corresponding algorithm that supports both verification and falsification. To the best of our knowledge, this is the first method capable of falsifying ReLU neural barrier certificates, thereby filling a critical gap for their reliable application in safety-critical systems. Through numerical experiments, we demonstrated that our method achieves superior verification performance compared to existing approaches, particularly in high-dimensional systems.

References

Abate, A.; Ahmed, D.; Edwards, A.; Giacobbe, M.; and Peruffo, A. 2021. FOSSIL: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th international conference on hybrid systems: computation and control*, 1–11.

Ames, A. D.; Coogan, S.; Egerstedt, M.; Notomista, G.; Sreenath, K.; and Tabuada, P. 2019. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, 3420–3431. IEEE.

Ames, A. D.; Xu, X.; Grizzle, J. W.; and Tabuada, P. 2016. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8): 3861–3876.

Aubin, J.-P.; and Frankowska, H. 2009. *Set-valued analysis*. Springer Science & Business Media.

Caviness, B. F.; and Johnson, J. R. 2012. *Quantifier elimination and cylindrical algebraic decomposition*. Springer Science & Business Media.

Clark, A. 2021. Verification and synthesis of control barrier functions. In *2021 60th IEEE Conference on Decision and Control (CDC)*, 6105–6112. IEEE.

Clarke, F. H.; Ledyaev, Y. S.; Stern, R. J.; and Wolenski, P. R. 2008. *Nonsmooth analysis and control theory*, volume 178. Springer Science & Business Media.

Conforti, M.; Cornuéjols, G.; and Zambelli, G. 2014. *Integer Programming*. Graduate Texts in Mathematics. Springer International Publishing. ISBN 9783319110080.

Dai, L.; Gan, T.; Xia, B.; and Zhan, N. 2017. Barrier certificates revisited. *Journal of Symbolic Computation*, 80: 62–86.

Dawson, C.; Gao, S.; and Fan, C. 2023. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 39(3): 1749–1767.

Dawson, C.; Qin, Z.; Gao, S.; and Fan, C. 2022. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, 1724–1735. PMLR.

De Moura, L.; and Bjørner, N. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, 337–340. Springer.

Gao, S.; Kong, S.; and Clarke, E. M. 2013. dReal: An SMT solver for nonlinear theories over the reals. In *International conference on automated deduction*, 208–214. Springer.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

Hu, H.; Yang, Y.; Wei, T.; and Liu, C. 2024. Verification of neural control barrier functions with symbolic derivative bounds propagation. In *8th Annual Conference on Robot Learning*.

Liu, S.; Liu, C.; and Dolan, J. 2023. Safe control under input limits with neural control barrier functions. In *Conference on Robot Learning*, 1970–1980. PMLR.

Maas, A. L.; Hannun, A. Y.; Ng, A. Y.; et al. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, 3. Atlanta, GA.

Mathiesen, F. B.; Calvert, S. C.; and Laurenti, L. 2022. Safety certification for stochastic systems via neural barrier functions. *IEEE Control Systems Letters*, 7: 973–978.

Montufar, G. F.; Pascanu, R.; Cho, K.; and Bengio, Y. 2014. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27.

Qin, Z.; Zhang, K.; Chen, Y.; Chen, J.; and Fan, C. 2021. Learning safe multi-agent control with decentralized neural barrier certificates. *arXiv preprint arXiv:2101.05436*.

Raghu, M.; Poole, B.; Kleinberg, J.; Ganguli, S.; and Sohl-Dickstein, J. 2017. On the expressive power of deep neural networks. In *international conference on machine learning*, 2847–2854. PMLR.

Serra, T.; Tjandraatmadja, C.; and Ramalingam, S. 2018. Bounding and counting linear regions of deep neural networks. In *International conference on machine learning*, 4558–4566. PMLR.

Xiao, W.; Wang, T.-H.; Hasani, R.; Chahine, M.; Amini, A.; Li, X.; and Rus, D. 2023. Barriernet: Differentiable control barrier functions for learning of safe robot control. *IEEE Transactions on Robotics*, 39(3): 2289–2307.

Zhang, H.; Qin, Z.; Gao, S.; and Clark, A. 2024. SEEV: Synthesis with Efficient Exact Verification for ReLU Neural Barrier Functions. *arXiv preprint arXiv:2410.20326*.

Zhang, H.; Wu, J.; Vorobeychik, Y.; and Clark, A. 2023. Exact verification of relu neural control barrier functions. *Advances in neural information processing systems*, 36: 5685–5705.

Zhao, H.; Zeng, X.; Chen, T.; and Liu, Z. 2020. Synthesizing barrier certificates using neural networks. In *Proceedings of the 23rd international conference on hybrid systems: Computation and control*, 1–11.

Zhao, H.; Zeng, X.; Chen, T.; Liu, Z.; and Woodcock, J. 2021a. Learning safe neural network controllers with barrier certificates. *Formal Aspects of Computing*, 33: 437–455.

Zhao, Q.; Chen, X.; Zhang, Y.; Sha, M.; Yang, Z.; Lin, W.; Tang, E.; Chen, Q.; and Li, X. 2021b. Synthesizing ReLU neural networks with two hidden layers as barrier certificates for hybrid systems. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 1–11.

Zhao, Q.; Chen, X.; Zhao, Z.; Zhang, Y.; Tang, E.; and Li, X. 2022. Verifying neural network controlled systems using neural networks. In *Proceedings of the 25th ACM International Conference on Hybrid Systems: Computation and Control*, 1–11.