

DNN Verification, Reachability, and the Exponential Function Problem

Omri Isac

The Hebrew University of Jerusalem

Yoni Zohar

Bar-Ilan University

Clark Barrett

Stanford University

Guy Katz

The Hebrew University of Jerusalem

Abstract

Deep neural networks (DNNs) are increasingly being deployed to perform safety-critical tasks. The opacity of DNNs, which prevents humans from reasoning about them, presents new safety and security challenges. To address these challenges, the verification community has begun developing techniques for rigorously analyzing DNNs, with numerous verification algorithms proposed in recent years. While a significant amount of work has gone into developing these verification algorithms, little work has been devoted to rigorously studying the computability and complexity of the underlying theoretical problems. Here, we seek to contribute to the bridging of this gap. We focus on two kinds of DNNs: those that employ piecewise-linear activation functions (e.g., ReLU), and those that employ piecewise-smooth activation functions (e.g., Sigmoids). We prove the two following theorems:

1. The decidability of verifying DNNs with piecewise-smooth activation functions is equivalent to a well-known, open problem formulated by Tarski; and
2. The DNN verification problem for any quantifier-free linear arithmetic specification can be reduced to the DNN reachability problem, whose approximation is NP-complete.

These results answer two fundamental questions about the computability and complexity of DNN verification, and the ways it is affected by the network’s activation functions and error tolerance; and could help guide future efforts in developing DNN verification tools.

2012 ACM Subject Classification Theory of Computation

Keywords and phrases Formal Verification, Computability Theory, Deep Neural Networks

1 Introduction

The use of artificial intelligence, and specifically that of deep neural networks (DNNs), is becoming extremely widespread — as DNNs are often able to solve complex tasks more successfully than any other computational approach. These include critical tasks in healthcare [13], autonomous driving [8], communication networks [10], and also the task of communicating with humans through text [9] — which seems to bring DNNs closer and closer to passing the famous Turing test [53].

However, it has been shown that even state-of-the-art DNNs are susceptible to various errors. In one infamous example, known as *adversarial perturbations*, small input perturbations that are imperceptible to the human eye are crafted to fool modern DNNs, causing them to output incorrect results selected by an attacker [19]. Adversarial perturbations thus constitute a safety and security threat, to which most DNNs are susceptible [48]. Other issues, such as privacy concerns and bias against various groups, have also been observed, making it clear that a high bar of trustworthiness must be met before stakeholders can fully accept DNNs [26].

Overcoming these weaknesses of DNNs is a significant challenge, due to their size and complexity. This is further aggravated by the fact that DNNs are machine-generated (automatically *trained* over many examples). Consequently, they are opaque to human engineers, and often fail to generalize their results to examples sufficiently different from the set of examples used for training [29]. This has sparked much interest in the verification community, which began studying verification techniques for DNNs, in order to guarantee their compliance with given specifications. In recent years, the verification community has designed and implemented multiple verification algorithms for DNNs, relying on techniques such as SMT solving [22, 25, 56], abstract interpretation [16], convex relaxation [30], adversarial search [20], and many others [57, 5, 7, 15, 2, 37, 43, 46, 52, 54, 12, 47, 4]. Indeed, DNN verification technology has been making great strides in recent years [32].

Modern verification algorithms depend heavily on the structure of the DNN being verified, and specifically on the type of its *activation functions*. Initial efforts at DNN verification focused almost exclusively on DNNs with piecewise-linear (PWL) activation functions. It has been shown that the verification of such networks is an NP-complete problem [25, 41], and multiple algorithms have been proposed for solving it [28, 25, 16]. Although more recent approaches can handle DNNs with smooth activation functions, these algorithms are often approximate and/or incomplete [33, 20, 46]; in fact, to the best of our knowledge, there is not a single algorithm that is guaranteed to terminate with a correct answer when verifying such DNNs. This raises two important questions:

1. Does there exist a non-approximating algorithm that can *always* solve verification queries involving DNNs with non-PWL activation functions? Or, in other words, is the verification problem of DNNs with smooth and piecewise-smooth activation functions *decidable*? and
2. When introducing approximations, how difficult does the verification problem become with respect to the DNN, the specification, and the size of the approximation? In other words, what is the computational complexity of DNN verification with smooth and piecewise-smooth activation functions and with ϵ -error tolerance?

In this paper, we provide a partial answer for the first question, by showing that the verification problem of DNNs with smooth and piecewise-smooth activation functions is equivalent to a well-known, open problem from the field of model theory — Tarski’s exponential function problem [50]. We do so by introducing a constructive bijection between verification queries of such DNNs, and instances of Tarski’s open problem.

In addition, we provide a partial answer to the second question, by studying the relations between DNN verification and DNN reachability problems, and ultimately proving that they are equivalent. This result enables further investigation of DNN verification as a specific case of DNN reachability, without loss of generality. The latter problem is known to be NP-complete when ϵ -error tolerance is introduced in the result [40].

Formally, we prove the two following theorems:

1. The DNN verification problem for DNNs with smooth and piecewise-smooth activation functions is equivalent to Tarski’s exponential function problem [50], which is a well-known open problem.
2. The DNN verification problem, with any quantifier-free linear arithmetic specification formula, can be reduced to the DNN reachability problem, which is NP-complete when some ϵ -error tolerance is allowed [40].

Our results imply a fundamental difference between the hardness of verification of DNNs with piecewise-smooth and with piecewise-linear activation functions. As far as we know, we are the first to provide any proof of this difference.

The rest of the paper is organized as follows. In Section 2 we provide background on DNNs, verification and other necessary mathematical concepts. In Section 3 and Section 4 we formally prove the two main results mentioned above. In Section 5 we discuss related work, and in Section 6 we describe our conclusions and directions for future work.

2 Background

2.1 Deep Neural Networks

Deep neural networks (DNNs) [18] are directed graphs whose nodes (neurons) are organized into layers, and whose nodes and edges are labeled with rational numbers. Nodes in the first layer, called the *input layer*, are assigned values matching the input to the DNN; and then the values of nodes in each of the subsequent layers are computed as functions of the values assigned to neurons in the preceding layer. More specifically, each node value is computed by first applying an affine transformation (linear transformation and addition of a constant) to the values from the preceding layer, and then applying a non-linear *activation function* [11] to the result. The final (output) layer, which corresponds to the output of the network, is computed without applying an activation function.

Three of the most common activation functions are the *rectified linear unit* (ReLU), which is defined as:

$$\text{ReLU}(x) = \begin{cases} x & x > 0 \\ 0 & \text{otherwise;} \end{cases}$$

the *Sigmoid* function, defined as:

$$\sigma(x) : \mathbb{R} \rightarrow (0, 1) = \frac{\exp(x)}{\exp(x)+1}$$

where \exp is the exponential function; and the *hyperbolic tangent*, defined as:

$$\tanh(x) : \mathbb{R} \rightarrow (-1, 1) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$$

The latter two activation functions are both injective, and their inverses are defined as follows:

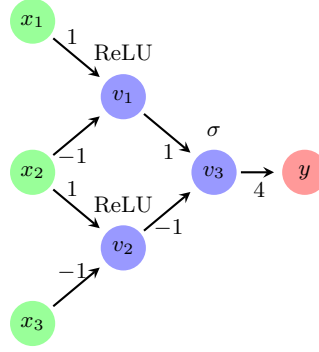
$$\begin{aligned} \forall x \in (0, 1) : \sigma^{-1}(x) &= \ln\left(\frac{x}{1-x}\right) \\ \forall x \in (-1, 1) : \tanh^{-1}(x) &= \frac{1}{2} \ln\left(\frac{1+x}{1-x}\right) \end{aligned}$$

where \ln is the natural logarithm function. In addition, we consider the NLRReLU activation function [27, 11], denoted τ for short, which is defined as:

$$\text{NLRReLU}(x) = \tau(x) = \ln(\text{ReLU}(x) + 1)$$

A simple DNN with four layers appears in Figure 1. For input $\langle 1, 2, 1 \rangle$, the first node in the second layer evaluates to $\text{ReLU}(1 \cdot 1 + 2 \cdot (-1)) = \text{ReLU}(-1) = 0$; the second node in the second layer evaluates to $\text{ReLU}(2 \cdot 1 + 1 \cdot (-1)) = \text{ReLU}(1) = 1$; and the node in the third layer evaluates to $\sigma(0 - 1) = \sigma(-1)$. Thus, the node in the fourth (output) layer evaluates to $4 \cdot \sigma(-1)$.

Formally, a DNN $\mathcal{N} : \mathbb{R}^m \rightarrow \mathbb{R}^k$, is a sequence of n layers L_0, \dots, L_{n-1} where each layer L_i consists of $s_i \in \mathbb{N}$ nodes, denoted $v_i^1, \dots, v_i^{s_i}$, and biases $p_i^j \in \mathbb{Q}$ for each v_i^j . Each directed edge in the DNN is of the form (v_{i-1}^l, v_i^j) and is labeled with $w_{i,j,l} \in \mathbb{Q}$. The assignment to the nodes in the input layer is defined by $v_0^j = x_j$, where $\bar{x} \in \mathbb{R}^m$ is the input vector, and the assignment for the j^{th} node in the $1 \leq i < n - 1$ layer is computed as



■ **Figure 1** A toy DNN.

$$v_i^j = f_i^j \left(\sum_{l=1}^{s_{i-1}} w_{i,j,l} \cdot v_{i-1}^l + p_i^j \right)$$

for some activation function $f_i^j : \mathbb{R} \rightarrow \mathbb{R}$. Finally, neurons in the output layer are computed as:

$$v_{n-1}^j = \sum_{l=1}^{s_{n-2}} w_{n-1,j,l} \cdot v_{n-2}^l + p_{n-1}^j$$

where $w_{i,j,l}$ and p_i^j are (respectively) the predetermined weights and biases of \mathcal{N} .

2.2 Formal Analysis of DNNs

The formal methods community has tackled the formal analysis of DNNs primarily along two axes: DNN *verification*, and DNN *reachability*. These are two related formulations, as reachability problems may be expressed as verification problems in a straightforward manner. In this paper, we further study the connections between these two formulations.

DNN Verification. Let $\mathcal{N} : \mathbb{R}^m \rightarrow \mathbb{R}^k$ be a DNN and $P : \mathbb{R}^{m+k} \rightarrow \{\top, \perp\}$ be a property, where \top, \perp represent the values for which the property does and does not hold, respectively. The *DNN verification problem* is to decide whether there exist $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^k$ such that $(\mathcal{N}(x) = y) \wedge P(x, y)$ holds. In particular, a verification query is always expressed as an existential formula. If such x and y exist, we say that the verification query $\langle \mathcal{N}, P \rangle$ is *satisfiable* (**SAT**); and otherwise, we say that it is *unsatisfiable* (**UNSAT**).

A verification algorithm is *sound* if it does not return **UNSAT** for satisfiable queries, and does not return **SAT** for unsatisfiable queries (in other words, if its answers are always correct); and is *complete* if it always terminates, for any query.

So far, there have been several efforts at studying the complexity-theoretical aspects of DNN verification [14, 21, 41, 42, 40, 24, 25]. Most previous work was focused on DNNs with piecewise-linear activation function (specifically, ReLUs), while leaving many open questions about the complexity-theoretical aspects of verifying DNNs with smooth and piecewise-smooth activation functions. The complexity of more general SMT problems has received significant attention [44].

DNN Reachability. Given a DNN $\mathcal{N} : \mathbb{R}^m \rightarrow \mathbb{R}$ and an input set $\mathcal{X} \subseteq [0, 1]^m$, the *DNN reachability problem* is to compute $\sup_{x \in \mathcal{X}} \mathcal{N}(x)$ and $\inf_{x \in \mathcal{X}} \mathcal{N}(x)$, perhaps up to some ϵ -error tolerance. For DNNs with Lipschitz-continuous activation functions, either smooth or

piecewise-linear, (such as σ and ReLU), the reachability problem with some ϵ -error tolerance is NP-complete, in the size of the network and ϵ [40]. In this work, we consider a decision version for the problem, deciding whether $\sup_{x \in \mathcal{X}} \mathcal{N}(x) \geq 0$ is achieved for some $x \in \mathcal{X}$. This decision version with some ϵ -error tolerance is then to decide whether $\sup_{x \in \mathcal{X}} \mathcal{N}(x) \geq -\epsilon$ is achieved for some $x \in \mathcal{X}$. In addition, we may assume that the input of \mathcal{N} is within $[0, 1]^m$, as the input domain may be normalized before the network is evaluated.

For example, consider the DNN depicted in Figure 1. A possible *verification* query for this DNN is given by a property P that returns \top if and only if $(x_1, x_2, x_3) \in [0, 1]^3 \wedge (y \in [0.5, 0.75] \vee y \in [0, 0.25])$; i.e., if there exists an input in the $[0, 1]^3$ cube, for which $y \in [0.5, 0.75] \vee y \in [0, 0.25]$. A possible *reachability* query is to check whether there exists an input in the domain $[0, 1] \times [0, 0.5] \times [0.5, 1]$, for which $y \geq 0$. This reachability query can trivially be represented as a verification property P' , which returns \top if and only if $(x_1, x_2, x_3) \in [0, 1] \times [0, 0.5] \times [0.5, 1] \wedge y \geq 0$. For any $\epsilon > 0$, the equivalent reachability query with ϵ tolerance is to decide if there exists an input in the domain $[0, 1] \times [0, 0.5] \times [0.5, 1]$, for which $y \geq -\epsilon$.

2.3 Decidability and Mathematical Logic

Mathematical Logic. In mathematical logic, a *signature* Σ is a set of symbols, representing functions and relations. A Σ -*formula* is a formula, comprised of atoms and relations that appear in Σ , the usual logical operators ($\wedge, \neg, \vee, \rightarrow, \leftrightarrow$), and the quantifiers \forall (universal) and \exists (existential). A variable affixed with a quantification symbol is a *bounded variable*; and otherwise it is a *free variable*. A formula without free variables is called a *sentence*, and a formula without bounded variables is called a *quantifier-free formula*. A formula with variables $\bar{x} = (x_1, \dots, x_n)$ of the form $\exists(\bar{x}).\varphi$ where φ is quantifier-free is called an *existential formula*. A Σ -*theory* is a set of Σ -sentences. A Σ -*model* \mathcal{M} is comprised of a set of elements, denoted $|\mathcal{M}|$, and an *interpretation* for all Σ functions and relations; that is, a definition $f^{\mathcal{M}} : |\mathcal{M}|^n \rightarrow |\mathcal{M}|$ for every n -ary function $f \in \Sigma$, and a definition $r^{\mathcal{M}} \subseteq |\mathcal{M}|^m$ for every m -ary relation $r \in \Sigma$. If the interpretation of a Σ -sentence φ is true within a model \mathcal{M} , we say that \mathcal{M} *satisfies* φ , and denote $\mathcal{M} \models \varphi$. If \mathcal{M} satisfies all sentences in a Σ -theory \mathcal{T} , then \mathcal{M} is a \mathcal{T} -*model*, denoted $\mathcal{M} \models \mathcal{T}$. Given some model \mathcal{M} over signature Σ , we define the theory $Th(\mathcal{M})$ as the set of all Σ -sentences φ such that $\mathcal{M} \models \varphi$. It is then trivial that $\mathcal{M} \models Th(\mathcal{M})$.

For example, let Σ be the set $\{+, -, \cdot, 0, 1, <\}$ where $+$, $-$, \cdot are 2-ary functions, $0, 1$ are 0-ary functions and $<$ is a 2-ary relation. Let \mathcal{M} be a model defined over \mathbb{R} with addition, subtraction, multiplication, the constant $0, 1$ and the usual order. Then $Th(\mathcal{M})$ is the set of all Σ -sentences that \mathcal{M} satisfies, such as $\forall x : x \cdot 0 = 0$.

Theory Decidability. For a Σ -theory \mathcal{T} and a Σ -sentence φ , we say that φ is \mathcal{T} -*valid* and denote $\mathcal{T} \vdash \varphi$, if every model of \mathcal{T} satisfies φ . Furthermore, we say that φ is \mathcal{T} -*satisfiable* if there exists a model \mathcal{M} of \mathcal{T} , for which $\mathcal{M} \models \varphi$; and that φ is \mathcal{T} -*unsatisfiable* if $\mathcal{M} \not\models \varphi$ for all models \mathcal{M} of \mathcal{T} . Satisfiability and validity are closely connected, as φ is \mathcal{T} -valid if and only if $\neg\varphi$ is \mathcal{T} -unsatisfiable. A theory \mathcal{T} is *decidable* if there exists an algorithm that, for any sentence φ , decides whether $\mathcal{T} \vdash \varphi$, within a finite number of steps. If φ is valid, the algorithm returns \top ; and otherwise, it returns \perp . Due to the connection of satisfiability and validity, validity-checking algorithms may also be used to decide satisfiability, and vice versa. In particular, for any theory $Th(\mathcal{M})$ for some model \mathcal{M} , all $Th(\mathcal{M})$ -models satisfy exactly the same sentences, so validity and satisfiability are equivalent. Thus, throughout

this paper we use decision procedures to decide the $Th(\mathcal{M})$ -satisfiability of formulas. In addition, when considering *quantifier-free* formulas (i.e., a formula where all variables are free), all of the formula's variables are implicitly existentially quantified. In this case, for any quantifier-free formula $\varphi(\bar{x})$ with variable vector \bar{x} , the satisfiability problem of $\varphi(\bar{x})$ with respect to a model \mathcal{M} is equivalent to deciding whether $\mathcal{M} \models \exists \bar{x}.\varphi(\bar{x})$. Similarly, the satisfiability problem of $\varphi(\bar{x})$ with respect to a theory \mathcal{T} is equivalent to deciding whether $\exists \bar{x}.\varphi(\bar{x})$ is \mathcal{T} -satisfiable.

It has previously been shown that the theory of the real field $Th(\mathbb{R}, +, -, \cdot, 0, 1, <)$ is decidable [49], and that the theory of the real field with the transcendental functions \exp, \sin and the constants $\log 2, \pi$ is undecidable [38]. The question of whether $Th(\mathbb{R}, +, -, \cdot, 0, 1, <, \exp)$ is decidable has remained an open problem since the 1950's [50], and it is commonly known as Tarski's *exponential function problem*.

A theory \mathcal{T} is *stably-infinite* if for every quantifier-free formula φ , the satisfiability of φ in \mathcal{T} implies that φ is satisfiable in some infinite model of \mathcal{T} . It is then immediate that for any infinite model \mathcal{M} , $Th(\mathcal{M})$ is stably-infinite.

Given two decidable, stably-infinite theories \mathcal{T}_1 and \mathcal{T}_2 defined over disjoint sets of symbols, for any quantifier-free formulas $F_1 \in \mathcal{T}_1, F_2 \in \mathcal{T}_2$, the formula $F_1 \wedge F_2$ is decidable as well [35]. The *Nelson-Oppen method* [35] is a well-known method for combining two decision procedures for two theories into a decision procedure for the quantifier-free fragment of their union.

Equisatisfiability. Two formulas φ and ψ are *equisatisfiable* if φ is satisfiable if and only if ψ is satisfiable. For example, the formulas $\varphi := (a + b) * (a - b) = 0$ and $\psi := c * d = 0 \wedge c = a + b \wedge d = a - b$ are equisatisfiable. Note that φ and ψ may be formulated in different theories, \mathcal{T}_1 and \mathcal{T}_2 , respectively. In this case, we say that the formulas are equisatisfiable if φ is \mathcal{T}_1 -satisfiable if and only if ψ is \mathcal{T}_2 -satisfiable.

Function Definability. For any signature Σ and an n -ary function f , not necessarily in Σ , we say that f is *definable* in a Σ -model \mathcal{M} if there exists a Σ -formula $\psi(x_1, \dots, x_n, y, z_1, \dots, z_m)$ over the variables x_1, \dots, x_n, y such that for any elements a_1, \dots, a_n, b in \mathcal{M} we have that $\mathcal{M} \models \exists z_1, \dots, z_m.\psi(a_1, \dots, a_n, b, z_1, \dots, z_m)$ if and only if $b = f(a_1, \dots, a_n)$. We say that f is definable in a Σ -theory \mathcal{T} if it is definable in all models of \mathcal{T} .

Model-Completeness. In model theory, the concept of model-completeness has a few equivalent definitions. For our purposes, a theory \mathcal{T} is model-complete if and only if any formula in the theory has an equivalent existential formula (modulo \mathcal{T}). This means that the existential formulas in \mathcal{T} can express all the formulas in it. The theory $Th(\mathbb{R}, +, -, \cdot, 0, 1, <, \exp)$ is known to be model-complete [55].

3 Decidability of DNN Verification

In this section, we prove our first main result: the decidability of verifying a DNN with the activation functions $\text{ReLU}, \sigma, \tanh$ and τ is equivalent to the decidability of $Th(\mathbb{R}, +, -, \cdot, 0, 1, <, \exp)$. The decidability of this theory is an open problem [50]. Thus, the equivalence implies that the decidability of DNN verification for DNNs with the activation functions $\text{ReLU}, \sigma, \tanh$ and τ is an open problem as well.

For simplicity, we denote $\mathcal{T}_{\mathbb{R}} = Th(\mathbb{R}, +, -, \cdot, 0, 1, <)$, $\mathcal{T}_{\exp} = Th(\mathbb{R}, +, -, \cdot, 0, 1, <, \exp)$, and $\mathcal{T}_{\sigma} = Th(\mathbb{R}, +, -, \cdot_q, 0, 1, <, \sigma, \tanh, \tau)$, where \cdot_q is a unary function, interpreted as the multiplication with a constant $q \in \mathbb{Q}$. We use $\Sigma_{\sigma}, \Sigma_{\exp}$ and $\Sigma_{\mathbb{R}}$ to denote the signatures of $\mathcal{T}_{\sigma}, \mathcal{T}_{\exp}$ and $\mathcal{T}_{\mathbb{R}}$, respectively. Note that for any DNN, weights and biases are in \mathbb{Q} , and can thus be expressed as $\mathcal{T}_{\mathbb{R}}$ -terms. Therefore, we can express the affine constraints of the network

as $\mathcal{T}_{\mathbb{R}}$ -Formulas. In addition, any constraint of the form $f = \text{ReLU}(b)$ can be expressed as the formula $(f = b \leftrightarrow b > 0) \wedge (f = 0 \leftrightarrow b \leq 0)$, and thus ReLU is definable in both \mathcal{T}_{exp} and \mathcal{T}_{σ} . Therefore, without loss of generality, we need not add a function symbol to Σ_{σ} to express DNNs with ReLU activation functions (or any other piecewise-linear function). Our goal is then to show that the decidability of \mathcal{T}_{exp} is equivalent to the decidability of all existential formulas of \mathcal{T}_{σ} .

Example: We begin with an example that illustrates this equivalence. For the first direction, consider the toy DNN depicted in Figure 1, and let $x_1, x_2, x_3, b_1, f_1, b_2, f_2, b_3, f_3$, and y be the variables of the network. Variables x_1, x_2, x_3 represent the input variables, variables $b_1, f_1, b_2, f_2, b_3, f_3$ represent the inputs and outputs of nodes v_1, v_2, v_3 , respectively, and variable y represents the network's output. Let P be the property restricting the input to be within $[0, 1]^3$ and the output to be in $[1, 2]$. The verification query for the network in Figure 1 and P is then:

$$\begin{aligned} & \bigwedge_{i \in \{1,2,3\}} [(x_i \geq 0) \wedge (x_i \leq 1)] \wedge \\ & (x_1 - x_2 = b_1) \wedge (x_2 + x_3 = b_2) \wedge \\ & \bigwedge_{i \in \{1,2\}} [(f_i = b_i) \leftrightarrow (b_i > 0)] \wedge [(f_i = 0) \leftrightarrow (b_i \leq 0)] \wedge \\ & (f_1 - f_2 = b_3) \wedge (f_3 = \sigma(b_3)) \wedge (4 \cdot f_3 = y) \wedge \\ & (1 < y) \wedge (y < 2) \end{aligned}$$

This is a \mathcal{T}_{σ} query, which can be expressed as a query in \mathcal{T}_{exp} , since $\sigma(x) = \frac{\exp(x)}{1+\exp(x)}$. The equivalent \mathcal{T}_{exp} query is:

$$\begin{aligned} & \bigwedge_{i \in \{1,2,3\}} [(x_i \geq 0) \wedge (x_i \leq 1)] \wedge \\ & (x_1 - x_2 = b_1) \wedge (x_2 + x_3 = b_2) \wedge \\ & \bigwedge_{i \in \{1,2\}} [(f_i = b_i) \leftrightarrow (b_i > 0)] \wedge [(f_i = 0) \leftrightarrow (b_i \leq 0)] \wedge \\ & (f_1 - f_2 = b_3) \wedge [(\exp(b_3) + 1) \cdot f_3 = \exp(b_3)] \wedge (4 \cdot f_3 = y) \wedge \\ & (1 < y) \wedge (y < 2) \end{aligned}$$

For the second direction, we begin by demonstrating a purification process of a given formula $\varphi := \exp(a+b) = \exp(a) \cdot \exp(b)$ into a formula in \mathcal{T}_{σ} . We assume that we can define $\psi_{c=a \cdot b}$ and $\psi_{y=\exp(x)}$ in Σ_{σ} , which are defined over the variables a, b, c and x, y , respectively, and that witness the definability of the functions \cdot and \exp in \mathcal{T}_{σ} . Therefore, the formula

$$\psi_{p=\exp(a)} \wedge \psi_{q=\exp(b)} \wedge \psi_{r=\exp(a+b)} \wedge \psi_{r=p \cdot q}$$

is equisatisfiable to $\exp(a+b) = \exp(a) \cdot \exp(b)$.

Formally, we prove the following theorem:

► **Theorem 1.** *The decidability of verifying DNNs with σ , \tanh , τ and ReLU activation functions is equivalent to the decidability of $\text{Th}(\mathbb{R}, +, -, \cdot, 0, 1, <, \exp)$.*

Proof. The first direction of the proof is similar to a technique proposed by Ivanov et al. [24]. Assume there exists a decision procedure for \mathcal{T}_{exp} , and let $F \in \Sigma_{\sigma}$ be a DNN verification query. For any appearance of $\sigma(t)$ for some term t , we replace $t, \sigma(t)$ with the fresh variables x, y , respectively and add the conjunction:

$$(\exp(x) + 1) \cdot y = \exp(x) \wedge x = t$$

to the resulting formula. This is done in a way similar to the one described in the example. Similarly, for any appearance of $\tanh(t)$ for some term t , we replace $t, \tanh(t)$ with the fresh variables x, y , respectively and add the conjunction:

$$(\exp(x) + \exp(-x)) \cdot y = \exp(x) - \exp(-x) \wedge x = t$$

to the resulting formula. For defining τ , we first define:

$$\psi_{f=\text{ReLU}(b)} := (f = b \leftrightarrow b > 0) \wedge (f = 0 \leftrightarrow b \leq 0)$$

Now, for any appearance of $\tau(t) = \ln(\text{ReLU}(t) + 1)$ for some term t , we replace $t, \tau(t)$ with the fresh variables x, y , respectively and add the conjunction:

$$\psi_{z=\text{ReLU}(x)} \wedge \exp(y) = z + 1 \wedge x = t$$

to the resulting formula, where z is an additional fresh variable. After repeating this process iteratively, we convert any $F \in \Sigma_\sigma$ to an equisatisfiable formula $F' \in \Sigma_{\text{exp}}$. We then use the decision procedure to decide the satisfiability of F' .

The second direction of the proof is more complex. Assume we have a sound and complete verification procedure for DNNs with σ , \tanh and τ activation functions; that is, a decision procedure for deciding the satisfiability of quantifier-free Σ_σ -formulas in \mathcal{T}_σ .

Since \mathcal{T}_{exp} is model-complete, it is tempting to try and construct a decision procedure for the existential formulas of \mathcal{T}_{exp} . However, to the best of our knowledge, given a general formula in \mathcal{T}_{exp} it is not known how to effectively derive its equivalent existential formula. In order to circumvent this issue, we consider instead a fourth theory, $\mathcal{T}_e = Th(\mathbb{R}, +, -, \cdot, 0, 1, <, e)$, defined over the signature Σ_e , where $e : \mathbb{R} \rightarrow \mathbb{R}$ with $e(x) = \exp(\frac{1}{1+x^2})$ is the *restricted* exponential function. It has been shown by Angus and Wilkie [31] that the decidability of this theory implies the decidability of \mathcal{T}_{exp} , and that given any formula in the language of \mathcal{T}_e , one can effectively find an equivalent existential formula (in \mathcal{T}_e) [31]. Therefore, it is enough for our purpose to consider any existential formula $\exists \bar{x}. \varphi \in \Sigma_e$, and decide the satisfiability of φ in \mathcal{T}_e .

Let $\exists \bar{x}. \varphi \in \Sigma_e$ be an existential formula, where φ is a quantifier-free formula. We construct a Σ_σ -formula ψ , equisatisfiable to φ . In this construction, all variables are implicitly existentially quantified. In order to do so, it is enough to define formulas $\psi_{c=a \cdot b}$ and $\psi_{y=e(x)}$ over the variables a, b, c and x, y respectively, and witness the definability of the functions \cdot and e in \mathcal{T}_σ . In this case, given any formula $\varphi \in \Sigma_e$, we can iteratively replace any occurrence of terms of the form $t \cdot s$ with the fresh variable p and add the conjunction $\psi_{p=t \cdot s}$, and occurrences of terms of the form $e(x)$ with the fresh variable q and add the conjunction $\psi_{q=e(x)}$. This process terminates with a Σ_σ -formula ψ equisatisfiable to φ , allowing us to apply the decision procedure to ψ .

To complete the proof, it remains to show how $\psi_{c=a \cdot b}$ and $\psi_{y=e(x)}$ can be defined using the formula $\psi_{y=\ln(x)}$. We show the construction of $\psi_{y=\ln(x)}$ later, in Lemma 2, and we use it here to define both $\psi_{c=a \cdot b}$ and $\psi_{y=e(x)}$.

We start by defining $\psi_{c=a \cdot b}$. Note that $\forall a, b > 0$, it holds that $\ln(a \cdot b) = \ln(a) + \ln(b)$; and so $a \cdot b = \exp(\ln(a) + \ln(b))$, assuming $a, b > 0$. This equality can be expressed using the formula:

$$\theta_{c=a \cdot b} := \psi_{p=\ln(a)} \wedge \psi_{q=\ln(b)} \wedge \psi_{p+q=\ln(c)},$$

where c represents the value of $a \cdot b$, and p, q are fresh variables. For defining $\psi_{c=a \cdot b}$ for all $a, b \in \mathbb{R}$, we split into cases, and write:

$$\begin{aligned}
\psi_{c=a \cdot b} := & [(a > 0 \wedge b > 0) \rightarrow \theta_{c=a \cdot b}] \wedge \\
& [(a < 0 \wedge b > 0) \rightarrow \theta_{c=-a \cdot b}] \wedge \\
& [(a > 0 \wedge b < 0) \rightarrow \theta_{c=-a \cdot -b}] \wedge \\
& [(a < 0 \wedge b < 0) \rightarrow \theta_{c=-a \cdot -b}] \wedge \\
& [(a = 0 \vee b = 0) \leftrightarrow c = 0],
\end{aligned}$$

which represents the function \cdot and witnesses its definability in \mathcal{T}_σ .

We now define $\psi_{y=e(x)}$. Recall that $e(x) = \exp(\frac{1}{x^2+1})$, so in order to define $\psi_{y=e(x)}$ we use both $\psi_{c=a \cdot b}$ and $\psi_{y=\ln(x)}$:

$$\psi_{y=e(x)} := \psi_{a=\ln(y)} \wedge \psi_{1=a \cdot (b+1)} \wedge \psi_{b=x \cdot x}$$

where a, b are fresh variables.

We have defined both $\psi_{c=a \cdot b}$ and $\psi_{y=e(x)}$, which concludes our proof. \blacktriangleleft

For the completeness of this section, we provide now the proof of Lemma 2, which shows the construction of $\psi_{y=\ln(x)}$:

► **Lemma 2.** *The natural logarithm function \ln is definable in \mathcal{T}_σ .*

Proof. First, observe that for any $x \geq 1$ we have that $\tau(x-1) = \ln(\text{ReLU}(x-1) + 1) = \ln(x-1+1) = \ln(x)$. Second, observe that $\forall x \in (0, 1)$, the inverses of σ and \tanh are defined and are equal to:

$$\sigma^{-1}(x) = \ln\left(\frac{x}{1-x}\right) = \ln(x) - \ln(1-x)$$

and

$$\tanh^{-1}(x) = \frac{1}{2} \ln\left(\frac{1+x}{1-x}\right) = \frac{1}{2}(\ln(1+x) - \ln(1-x))$$

We conclude that:

$$\forall x \in (0, 1) : \sigma^{-1}(x) - 2 \tanh^{-1}(x) + \tau(x) = \ln(x) - \ln(1-x) - \ln(1+x) + \ln(1-x) + \ln(x+1) = \ln(x)$$

We can express this relation using the following formula, and the fresh variables a, b, c :

$$\theta_{x,y} := [x = \sigma(a)] \wedge [x = \tanh(b)] \wedge [c = \tau(x)] \wedge [y = a - 2b + c]$$

Where $2b$ is syntactic sugar for $\cdot_2(b)$. Thus, we can define:

$$\psi_{y=\ln(x)} := [(1 < x) \rightarrow (y = \tau(x-1))] \wedge [(x = 1) \leftrightarrow (y = 0)] \wedge [(0 < x < 1) \rightarrow \theta_{x,y}] \wedge [0 < x]$$

which concludes the proof. \blacktriangleleft

4 DNN Verification is DNN Reachability

The two main formal analysis approaches for DNNs, verification and reachability, are closely connected: a DNN reachability instance can be formulated as DNN verification in a straightforward manner, as in the example in Section 2.2. Presently, DNN analysis algorithms and tools typically support one of the two formulations. Here, we prove that DNN verification and DNN reachability are in fact equivalent.

In this part, we consider DNNs that use both piecewise-linear and Sigmoidal activation functions. We formally prove that any instance of the DNN verification problem, with any specification expressible by a quantifier-free linear arithmetic formula, can be reduced to an

instance of the DNN reachability problem. Since the reachability problem is a specific case of verification, we ultimately prove that for DNNs, reachability and verification are equivalent. Since it was shown that approximation of DNN reachability queries with Lipschitz-continuous activation functions (such is σ and ReLU) is NP-complete [40], we deduce that the DNN verification problem is reducible to a problem whose approximation is NP-complete. The reduction involves adding an additional input, denoted ϵ , and we use (x, ϵ) to denote the concatenation of ϵ to the input vector x . Formally, we prove the following theorem:

► **Theorem 3.** *Let $\mathcal{N} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a neural network, let φ be a quantifier-free property with atoms expressing affine constraints over variables y_i of \mathcal{N} , and let $X \subseteq \mathbb{R}^n$. There exists a neural network $\mathcal{N}' : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$, with $|\mathcal{N}'| = O(|\mathcal{N}| + |\varphi|)$ such that the two following conditions are equivalent:*

- $\exists x \in X. \mathcal{N}(x) \models \varphi$
- $\exists (x, \epsilon) \in X \times (0, 1]. \mathcal{N}'(x, \epsilon) \geq 0$

Example. We begin with an example for constructing \mathcal{N}' , given some DNN \mathcal{N} , and a property φ . Consider first $\mathcal{N} : \mathbb{R}^4 \rightarrow \mathbb{R}^2$ as depicted in Figure 2a and $\varphi := (y_1 > 0) \wedge (y_1 \geq y_2)$. We denote $\theta := y_1 \geq y_2$ and $\psi := y_1 > 0 \equiv \neg(-y_1 \geq 0)$. In Figure 2 we start with the initial DNN \mathcal{N} , and then iteratively add new nodes to \mathcal{N} . In particular, we show how to add neurons that are active if and only if $\mathcal{N} \models \theta$ and $\mathcal{N} \models \psi$, respectively in Figure 2b and Figure 2c. Lastly, in Figure 2d we show how to add the output neuron, such that $\exists x \in X. \mathcal{N}(x) \models \varphi$ if and only if $\exists (x, \epsilon) \in X \times (0, 1]. \mathcal{N}'(x, \epsilon) \geq 0$. This concludes our example.

We now prove the theorem by induction on the generating sequence of φ ; that is, a sequence of sub-formulas of φ : $\varphi_1, \dots, \varphi_n$ such that $\forall i, j$ if $j > i$ then φ_j cannot be a sub-formula of φ_i , and $\varphi_n = \varphi$. This allows inductive proofs over the formulas [45]. For example, a generating sequence for the formula

$$\varphi := \exists x, y. (3x \geq 7) \wedge \neg(y \geq x)$$

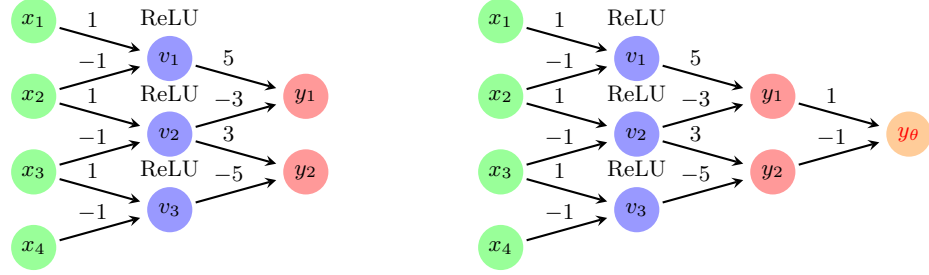
is:

$$y \geq x, 3x \geq 7, \neg(y \geq x), (3x \geq 7) \wedge \neg(y \geq x), \exists x, y. (3x \geq 7) \wedge \neg(y \geq x)$$

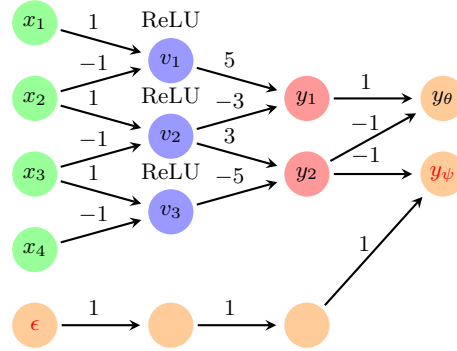
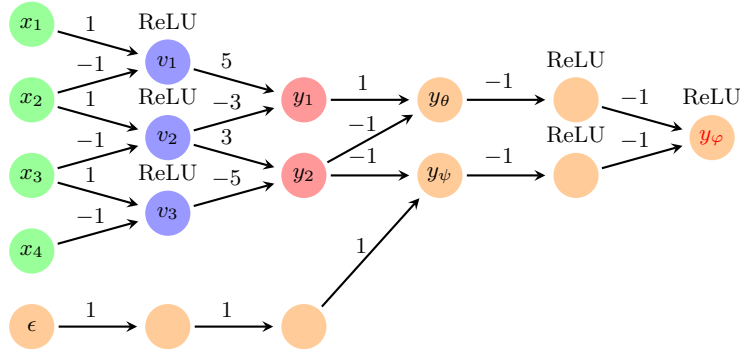
Proof. Without loss of generality, assume that φ is composed of atoms, negations, and conjunctions. In addition, assume that each variable y_j is an output variable (otherwise, we may add neurons with the identity as activation function from the neuron outputting y_j to the output layer). For every step i in the generating sequence $\varphi_1, \dots, \varphi_k = \varphi$, we add a constant number of output neurons, such that for any $x \in \mathbb{R}^n$, the resulting DNN \mathcal{N}'_i , satisfies $\mathcal{N}'_i \geq 0$ (for the last constructed output neuron) if and only if $\mathcal{N}(x) \models \varphi_i$. Below we explain the construction and prove its correctness; and in Figure 3 we show its visual representation.

Base Cases:

1. Let $\varphi := \top$. In this case, \mathcal{N}' is constructed from \mathcal{N} by adding a single ReLU neuron, with input edges with weight 0 from all output nodes of \mathcal{N} . We may add another neuron with the identity activation, to maintain the convention that the output layer does not have an activation function. Therefore, $\forall x \in \mathbb{R}^n : \mathcal{N}'(x) \geq 0$ if and only if $\text{ReLU}(\sum_i 0) \geq 0$, which is equivalent to \top . The case of \perp is covered by our handling of negations.



(a) The initial network.

(b) Adding a construct for $y_1 \geq y_2$.(c) Adding a construct for $y_2 > 0$.(d) Adding a construct for φ , as a conjunction.

■ **Figure 2** Constructing a reachability problem for $\mathcal{N} \models \varphi$.

2. Let $\varphi := \sum_i c_i \cdot y_i + b \geq 0$. In this case, \mathcal{N}' is constructed from \mathcal{N} by adding a single affine neuron with no activation function, with its input edges with weight c_i from every output neuron y_i of \mathcal{N} , and a bias b . Therefore, $\forall x \in \mathbb{R}^n$ and $\mathcal{N}(x) = y$ we have that $\sum_i c_i \cdot y_i + b \geq 0$ if and only if $\mathcal{N}'(x) \geq 0$. We note that equality can be handled using conjunctions, while strict inequalities can be handled using negations.

Inductive step:

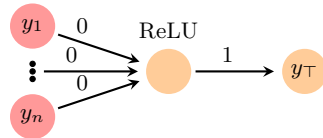
1. Let $\varphi := \psi \wedge \theta$, and let y_ψ, y_θ be the values of the neurons such that $y_\psi \geq 0, y_\theta \geq 0$ if and only if $\mathcal{N}(x) \models \psi, \theta$, respectively. Consider:

$$y_\varphi = -\text{ReLU}(-y_\psi) - \text{ReLU}(-y_\theta)$$

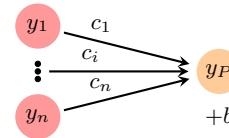
In this case, we have that $y_\varphi \geq 0$ if and only if $y_\psi \geq 0 \wedge y_\theta \geq 0$. We can see this since if $y_\psi \geq 0 \wedge y_\theta \geq 0$ then both $-\text{ReLU}(-y_\psi)$ and $-\text{ReLU}(-y_\theta)$ equal zero. Otherwise, at least one of $-\text{ReLU}(-y_\psi)$ and $-\text{ReLU}(-y_\theta)$ is negative (and the other is non-positive). Thus, we add two ReLU neurons, with a single -1 input edges from the nodes corresponding to y_ψ, y_θ , respectively. We then add a third neuron with two -1 edges from the ReLU nodes and no activation function.

2. Let $\varphi := \neg\psi$, and let y_ψ be the value of the neuron such that $y_\psi \geq 0$ if and only if $\mathcal{N}(x) \models \psi$. In this case, we first need to add a new ϵ_φ input neuron, and restrict it to $\epsilon_\varphi > 0$. Then, observe that $\neg(y_\psi \geq 0) \equiv y_\psi < 0$ if and only if there exists some $\epsilon > 0$ s.t. $\epsilon + y_\psi \leq 0$, or equivalently $-\epsilon - y_\psi \geq 0$. Therefore, we add a new neuron with no activation function, with a skip connection from the ϵ_φ neuron with weight -1 , and with a -1 weight from y_ψ , resulting in $y_\varphi = -\epsilon_\varphi - y_\psi$. We note that since there are finitely many such constructions, we can choose the minimal ϵ implied by all them, and again choose the minimum of it and 1. Thus, a single $\epsilon \in (0, 1]$ suffices. In addition, the use of the skip connections can be replaced with a line of ReLU neurons, starting with the ϵ neuron and feed-forwarding to a neuron on every layer. This construction does not affect the asymptotic size of \mathcal{N}' .

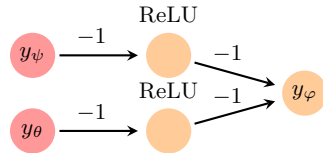
On every step of the recursion, we added only a constant number of neurons to the network, such that $x \in \mathbb{R}^n, \mathcal{N}' \geq 0$ if and only if $\mathcal{N}(x) \models \varphi_i$. This concludes our proof. ◀



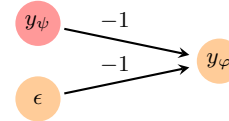
(a) An atom predicate of the form \top



(b) An atom predicate of the form $P := \sum_i c_i \cdot y_i + b \geq 0$



(c) A formula of the form $\varphi = \psi \wedge \theta$



(d) A formula of the form $\varphi = \neg\psi$

■ **Figure 3** Constructs for each step of the induction

5 Related Work

The complexity of DNN verification has been studied mainly for DNNs with piecewise-linear activation functions — specifically, the ReLU function. It has previously been shown that DNN verification is NP-complete, even for simple specifications [25, 41]. However, when certain restricted classes of DNN architectures and specifications are considered, DNNs with ReLUs only can be verified in polynomial time [14]. The verification complexity and computability in the case of reactive systems controlled by ReLU DNNs (i.e., the DNN acts as an agent that repeatedly interacts with an environment) has also been studied recently [2, 3]. One recent work showed that verifying CTL properties in this context is undecidable [1]. In our work, however, we consider DNNs as stand-alone functions.

When considering a realistic implementations of the ReLU function, e.g., in *quantized neural networks* [23], the DNN verification problem for bit-vector specifications is PSPACE-hard, introducing a big complexity gap from the case of ideal mathematical form. When specific models of *graph neural networks* [58] are considered, the verification problem is undecidable [42].

For DNNs with Sigmoidal activation functions, two main results are presently known. First, it was shown that reachability analysis with some error tolerance ϵ , for any Lipschitz-continuous activation function, is NP-complete in the size of the network and of ϵ [40]. Second, it was shown that the decidability of \mathcal{T}_{exp} implies the decidability of verifying DNNs with Sigmoidal activation functions, and that verifying such DNNs with a single hidden layer is decidable [24]. Our work here is another step towards a better understanding of the complexity of verifying such DNNs.

The connections between DNN verification and DNN reachability have also been studied before. Most prominently, it was shown that any local-robustness verification query can be reduced to a DNN reachability query [40].

6 Conclusion and Future Work

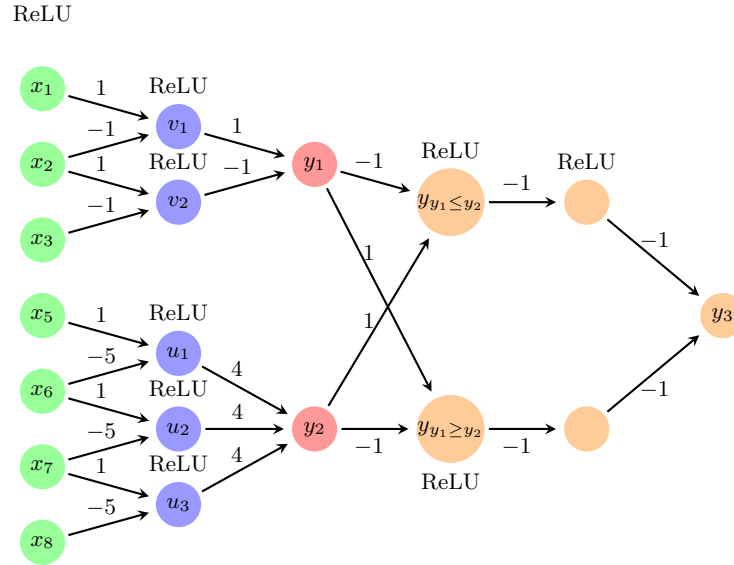
Our results show that for DNNs with ReLU, σ , tanh and NLRReLU activation functions, the decidability of the verification problem is equivalent to a well-known open problem; and that it can be reduced to a problem whose approximation is decidable, and for whose complexity an upper bound is known. This was achieved by reducing the verification problem to the corresponding reachability problem. These results show a significant difference between the verification problem for DNNs with piecewise-smooth activation functions and for DNNs with piecewise-linear activation functions, which is known to be NP-complete [25, 41].

Moving forward, one goal that we plan to pursue is a version of our first result that does not rely on the NLRReLU function, which is not as mainstream as the other functions that we considered. Although discarding this function does not alter the first direction of the proof, the second reduction currently requires it; and we plan to circumvent this requirement by defining a reduction from a Σ_{exp} -formula to a formula in the signature of $\mathcal{T}_{\mathbb{R}} \cup Th(\mathbb{R}, +, -, \cdot, q \in \mathbb{Q}, 0, 1, <, \sigma, \tanh)$, and then using a combination of the decision procedures for these two theories. It is noteworthy that the Nelson-Oppen method [35] cannot be directly applied here, since it requires the combined theories to be disjoint, which is not the case. Several generalizations of the Nelson-Oppen method for non-disjoint theories have previously been proposed [51, 17, 36, 39], and we speculate that these could be useful in this context.

Another interesting direction that we plan to pursue is to combine our work with approaches for switching between different kinds of machine learning models. For example, it would be intriguing to study whether DNNs with smooth activation functions can be reduced

to decision trees, as can apparently be done for piecewise-linear DNNs [6]. Equivalently, fundamental differences between piecewise-linear DNNs and smooth DNNs might imply similar differences between other classes of machine learning models.

Our second result could also be generalized, in two different manners. First, our construction could be applied to verification queries that involve multiple DNNs, e.g., verification queries used for proving DNN equivalence [34]. For two DNNs $\mathcal{N}_1, \mathcal{N}_2$ operating on the same domain, the verification query $\mathcal{N}_1(x) \stackrel{?}{=} \mathcal{N}_2(x)$ can be reduced to $\mathcal{N}'(x) \stackrel{?}{=} 0$, where \mathcal{N}' is constructed from copies of $\mathcal{N}_1, \mathcal{N}_2$ with outputs y_1, y_2 , and where additional neurons are used to stipulate that $y_3 \geq 0 \iff y_1 = y_2$, using our construction. In this case, we have that $\mathcal{N}' = O(|\mathcal{N}_1| + |\mathcal{N}_2|)$. An illustration of this construction appears in Figure 4. These results, in turn, could be generalized to queries over any finite number of DNNs.



■ **Figure 4** Reducing DNN equivalence to DNN reachability.

Second, similar constructions can support specification formulas over arithmetics that include any activation function (even piecewise-smooth). In this case, the number of added neurons is proportional not only to the sizes of the original network and the formula, but also to the number of activation functions composing the atoms. This construction is straightforward, and is omitted.

A final line of work that we intend to pursue in the future is to consider a more realistic framework of verification, with a concrete implementation of σ , rather than its pure mathematical form. This is similar to what was done for DNNs with ReLU activation functions [21]. In addition, we intend to characterize *decidable fragments* of the DNN verification problem, by restricting specifications and/or architectures; that is, we plan to identify sufficient conditions on the DNNs and specifications, which would render the resulting verification problem decidable. For such decidable fragments, studying the computational complexity of the verification problem is yet another intriguing line of work.

Acknowledgments. This work was supported by the Israel Science Foundation (grant number 619/21), the Binational Science Foundation (grant numbers 2020250, 2021769, 2020704), and by the National Science Foundation (grant numbers 1814369 and 2110397).

References

- 1 M. Akintunde, E. Botoeva, P. Kouvaros, and A. Lomuscio. Formal Verification of Neural Agents in Non-Deterministic Environments. *Autonomous Agents and Multi-Agent Systems*, 36:1–36, 2022.
- 2 M. Akintunde, A. Kevorchian, A. Lomuscio, and E. Pirovano. Verification of RNN-Based Neural Agent-Environment Systems. In *Proc. 33rd AAAI Conf. on Artificial Intelligence (AAAI)*, pages 197–210, 2019.
- 3 M. Akintunde, A. Lomuscio, L. Maganti, and E. Pirovano. Reachability Analysis for Neural Agent-Environment Systems. In *Proc. 16th Int. Conf. on Principles of Knowledge Representation and Reasoning*, 2018.
- 4 G. Amir, M. Schapira, and G. Katz. Towards Scalable Verification of Deep Reinforcement Learning. In *Proc. 21st Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 193–203, 2021.
- 5 G. Avni, R. Bloem, K. Chatterjee, T. Henzinger, B. Konighofer, and S. Pranger. Run-Time Optimization for Learned Controllers through Quantitative Games. In *Proc. 31st Int. Conf. on Computer Aided Verification (CAV)*, pages 630–649, 2019.
- 6 C. Aytekin. Neural Networks are Decision Trees, 2022. Technical Report. <https://arxiv.org/abs/2210.05189>.
- 7 T. Baluta, S. Shen, S. Shinde, K. Meel, and P. Saxena. Quantitative Verification of Neural Networks And its Security Applications. In *Proc. ACM SIGSAC Conf. on Computer and Communications Security (CCS)*, pages 1249–1264, 2019.
- 8 M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to End Learning for Self-Driving Cars, 2016. Technical Report. <http://arxiv.org/abs/1604.07316>.
- 9 OpenAI. ChatGPT: Optimizing Language Models for Dialogue, 2022. <https://openai.com/blog/chatgpt>.
- 10 M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah. Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial. *IEEE Communications Surveys & Tutorials*, 21(4):3039–3071, 2019.
- 11 S. Dubey, S. Singh, and B. Chaudhuri. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. *Neurocomputing*, 2022.
- 12 T. Eliyahu, Y. Kazak, G. Katz, and M. Schapira. Verifying Learning-Augmented Systems. In *Proc. Conf. of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 305–318, 2021.
- 13 A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean. A Guide to Deep Learning in Healthcare. *Nature Medicine*, 25(1):24–29, 2019.
- 14 J. Ferlez and Y. Shoukry. Bounding the Complexity of Formally Verifying Neural Networks: a Geometric Approach. In *Proc. 60th IEEE Conf. on Decision and Control (CDC)*, pages 5104–5109, 2021.
- 15 D. Fremont, J. Chiu, D. Margineantu, D. Osipychiev, and S. Seshia. Formal Analysis and Redesign of a Neural Network-Based Aircraft Taxiing System with VerifAI. In *Proc. 32nd Int. Conf. on Computer Aided Verification (CAV)*, pages 122–134, 2020.
- 16 T. Gehr, M. Mirman, D. Drachler-Cohen, E. Tsankov, S. Chaudhuri, and M. Vechev. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *Proc. 39th IEEE Symposium on Security and Privacy (S&P)*, 2018.

- 17 S. Ghilardi. Model-Theoretic Methods in Combined Constraint Satisfiability. *Journal of Automated Reasoning*, 33:221–249, 2004.
- 18 I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- 19 I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples, 2014. Technical Report. <http://arxiv.org/abs/1412.6572>.
- 20 P. Henriksen and A. Lomuscio. Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search. In *Proc. 24th European Conf. on Artificial Intelligence (ECAI)*, pages 2513–2520, 2020.
- 21 T. Henzinger, M. Lechner, and Đ. Žikelić. Scalable Verification of Quantized Neural Networks. In *Proc. AAAI Conf. on Artificial Intelligence*, pages 3787–3795, 2021.
- 22 X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety Verification of Deep Neural Networks. In *Proc. 29th Int. Conf. on Computer Aided Verification (CAV)*, pages 3–29, 2017.
- 23 I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.
- 24 R. Ivanov, J. Weimer, R. Alur, G. Pappas, and I. Lee. Verisig: Verifying Safety Properties of Hybrid Systems with Neural Network Controllers. In *Proc. 22nd ACM Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pages 169–178, 2019.
- 25 G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: a Calculus for Reasoning about Deep Neural Networks. *Formal Methods in System Design (FMSD)*, 2021.
- 26 X. Liu, L. Xie, Y. Wang, J. Zou, J. Xiong, Z. Ying, and A. Vasilakos. Privacy and Security Issues in Deep Learning: A Survey. *IEEE Access*, 9:4566–4593, 2020.
- 27 Y. Liu, J. Zhang, C. Gao, J. Qu, and L. Ji. Natural-Logarithm-Rectified Activation Function in Convolutional Neural Networks. In *Proc. 5th Int. Conf. on Computer and Communications (ICCC)*, pages 2000–2008, 2019.
- 28 A. Lomuscio and L. Maganti. An Approach to Reachability Analysis for Feed-Forward ReLU Neural Networks, 2017. Technical Report. <http://arxiv.org/abs/1706.07351>.
- 29 A. Lukina, C. Schilling, and T. Henzinger. Into the Unknown: Active Monitoring of Neural Networks. In *Proc. 21st Int. Conf. Runtime Verification (RV)*, pages 42–61, 2021.
- 30 Z. Lyu, C.-Y. Ko, Z. Kong, N. Wong, D. Lin, and L. Daniel. Fastened Crown: Tightened Neural Network Robustness Certificates. In *Proc. 34th AAAI Conf. on Artificial Intelligence (AAAI)*, pages 5037–5044, 2020.
- 31 A. Macintyre and A. Wilkie. On the Decidability of the Real Exponential Field. *Kreisel’s Mathematics*, 115:451, 1996.
- 32 M. Müller, C. Brix, S. Bak, C. Liu, and T. Johnson. The Third International Verification of Neural Networks Competition (VNN-COMP 2022): Summary and Results, 2022. Technical Report. <http://arxiv.org/abs/2212.10376>.
- 33 M. Müller, G. Makarchuk, G. Singh, M. Püschel, and M. Vechev. PRIMA: General and Precise Neural Network Certification via Scalable Convex Hull Approximations. In *Proc. 49th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*, pages 1–33, 2022.
- 34 N. Narodytska, S. Kasiviswanathan, L. Ryzhyk, M. Sagiv, and T. Walsh. Verifying Properties of Binarized Deep Neural Networks, 2017. Technical Report. <http://arxiv.org/abs/1709.06662>.
- 35 G. Nelson and D. Oppen. Simplification by Cooperating Decision Procedures. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1(2):245–257, 1979.
- 36 E. Nicolini, C. Ringeissen, and M. Rusinowitch. Data Structures with Arithmetic Constraints: A Non-Disjoint Combination. In *Proc. 7th Int. Symposium of Frontiers of Combining Systems (FroCoS)*, pages 319–334, 2009.

- 37 L. Pulina and A. Tacchella. An Abstraction-Refinement Approach to Verification of Artificial Neural Networks. In *Proc. 22nd Int. Conf. on Computer Aided Verification (CAV)*, pages 243–257, 2010.
- 38 D. Richardson. Some Undecidable Problems Involving Elementary Functions of a Real Variable. *The Journal of Symbolic Logic*, 33(4):514–520, 1969.
- 39 C. Ringeissen. Cooperation of Decision Procedures for the Satisfiability Problem. In *Proc. 1st Int. Workshop of Frontiers of Combining Systems (FroCoS)*, pages 121–139, 1996.
- 40 W. Ruan, X. Huang, and M. Kwiatkowska. Reachability Analysis of Deep Neural Networks with Provable Guarantees, 2018. Technical Report. <https://arxiv.org/abs/1805.02242>.
- 41 M. Sälzer and M. Lange. Reachability Is NP-Complete Even for the Simplest Neural Networks. In *Proc. 15th Int. Conf. on Reachability Problems (RP)*, pages 149–164, 2021.
- 42 M. Sälzer and M. Lange. Fundamental Limits in Formal Verification of Message-Passing Neural Networks. In *Proc. 11th Int. Conf. on Learning Representations (ICLR)*, 2023.
- 43 S. Sankaranarayanan, S. Dutta, and S. Mover. Reaching Out towards Fully Verified Autonomous Systems. In *Proc. 13th Int. Conf. on Reachability Problems (RP)*, pages 22–32, 2019.
- 44 R. Sebastiani. Colors Make Theories Hard. In *Proc. 8th Int. Joint Conf. on Automated Reasoning (IJCAR)*, pages 152–170, 2016.
- 45 J. Shoenfield. *Mathematical Logic*. Addison-Wesley publishing, 1967.
- 46 G. Singh, T. Gehr, M. Püschel, and M. Vechev. An Abstract Domain for Certifying Neural Networks. In *Proc. 46th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*, pages 1–30, 2019.
- 47 C. Strong, H. Wu, A. Zeljić, K. Julian, G. Katz, C. Barrett, and M. Kochenderfer. Global Optimization of Objective Functions Represented by ReLU Networks. *Journal of Machine Learning*, pages 1–28, 2021.
- 48 C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks, 2013. Technical Report. <http://arxiv.org/abs/1312.6199>.
- 49 A. Tarski. *Project Rand: A Decision Method for Elementary Algebra and Geometry*. Rand Corporation, 1948.
- 50 A. Tarski. A Decision Method for Elementary Algebra and Geometry. *Journal of Symbolic Logic*, 17(3):24–84, 1952.
- 51 C. Tinelli and C. Ringeissen. Unions of Non-Disjoint Theories and Combinations of Satisfiability Procedures. *Theoretical Computer Science*, 290(1):291–353, 2003.
- 52 H.-D. Tran, S. Bak, W. Xiang, and T. Johnson. Verification of Deep Convolutional Neural Networks Using ImageStars. In *Proc. 32nd Int. Conf. on Computer Aided Verification (CAV)*, pages 18–42, 2020.
- 53 A. Turing. Computing Machinery and Intelligence. *Mind*, LIX(236), 1950.
- 54 S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Formal Security Analysis of Neural Networks using Symbolic Intervals. In *Proc. 27th USENIX Security Symposium*, 2018.
- 55 A. Wilkie. Model Completeness Results for Expansions of the Ordered Field of Real Numbers by Restricted Pfaffian Functions and the Exponential Function. *Journal of the American Mathematical Society*, 9(4):1051–1094, 1996.
- 56 H. Wu, A. Zeljić, G. Katz, and C. Barrett. Efficient Neural Network Analysis with Sum-of-Infeasibilities. In *Proc. 28th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 143–163, 2022.

- 57 H. Zhang, M. Shinn, A. Gupta, A. Gurfinkel, N. Le, and N. Narodytska. Verification of Recurrent Neural Networks for Cognitive Tasks via Reachability Analysis. In *Proc. 24th European Conf. on Artificial Intelligence (ECAI)*, pages 1690–1697, 2020.
- 58 J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph Neural Networks: A Review of Methods and Applications. *AI open*, 1:57–81, 2020.