Neuro-Symbolic Verification of Deep Neural Networks

Xuan Xie[1] , **Kristian Kersting**[2,3] and **Daniel Neider**[1,4]
[1]Max Planck Institute for Software Systems, Kaiserslautern, Germany
[2]Computer Science Department and Centre for Cognitive Science, TU Darmstadt, Germany
[3]Hessian Center for AI (hessian.AI), Germany
[4]Carl von Ossietzky University of Oldenburg, Germany
{xie, neider}@mpi-sws.org, kersting@cs.tu-darmstadt.de

Abstract

Formal verification has emerged as a powerful approach to ensure the safety and reliability of deep neural networks. However, current verification tools are limited to only a handful of properties that can be expressed as first-order constraints over the inputs and output of a network. While adversarial robustness and fairness fall under this category, many real-world properties (e.g., "an autonomous vehicle has to stop in front of a stop sign") remain outside the scope of existing verification technology. To mitigate this severe practical restriction, we introduce a novel framework for verifying neural networks, named neuro-symbolic verification. The key idea is to use neural networks as part of the otherwise logical specification, enabling the verification of a wide variety of complex, real-world properties, including the one above. A defining feature of our framework is that it can be implemented on top of existing verification infrastructure for neural networks, making it easily accessible to researchers and practitioners.

1 Introduction

The exceptional performance of deep neural networks in areas such as perception and natural language processing has made them an integral part of many real-world AI systems, including safety-critical ones such as medical diagnosis and autonomous driving. However, neural networks are inherently opaque, and various defects have been observed and studied for state-of-the-art network architectures. The perhaps best known one among those is the lack of adversarial robustness [Szegedy *et al.*, 2014], which describes the phenomenon that even slight perturbations of an input to a neural network can cause entirely different outputs. In fact, defects in learning-based systems are so prevalent in practice that a dedicated database to monitor AI incidents and avoid repeated undesired outcomes has been introduced recently [McGregor, 2021].

Motivated by decade-long advances in software reliability, formal verification has emerged as a powerful approach to ensure the correctness and safety of neural networks (we refer the reader to Section 3 for further details). In contrast to (empirical) statistical evaluation methods from machine learning, such as cross-validation, formal verification techniques have the great advantage that they are not limited to checking a given property on just a finite number of inputs. Instead, they allow one to check whether a property holds for all (or at least infinitely many) inputs to a deep neural network, including unseen data and corner cases. However, formal verification has a fundamental limitation that often constitutes a significant obstacle in practice: it requires that the property to verify can be expressed as "simple" (typically quantifier-free, first-order) constraints over the inputs and output of a neural network.

While adversarial robustness and fairness fall under the above category, many real-world properties remain outside the scope of existing verification technology. Consider, for instance, a deep neural network controlling an autonomous vehicle and the property that the vehicle needs to decelerate as soon as a stop sign appears in the front view. It is not hard to see that formalizing this property in terms of constraints on inputs and outputs is extremely difficult, if not impossible, as it would require capturing all essential features of all possible stop signs on the level of image pixels (e.g., their position, shape, etc.). If this was possible, machine learning would not be necessary in the first place: one could simply implement a detection algorithm based on such a formal specification.

To overcome this severe limitation and make formal verification applicable to real-world scenarios, we propose a neuro-symbolic framework for verifying neural networks. Following the general idea of neuro-symbolic reasoning [d'Avila Garcez *et al.*, 2019; De Raedt *et al.*, 2020], our key contribution is a novel specification language, named *Neuro-Symbolic Assertion Language (*NeSAL*)*, allowing one to combine logical specifications and arbitrary neural networks. The neural networks, which we call *specification networks*, serve as proxies for complex, semantic properties and enable the integration of advances in fields such as perception and natural language recognition into formal verification. In the context of our example above, one could train a highly-specialized specification network to detect stop signs. Then, the desired property can be expressed straightforwardly as "if the specification network detects a stop sign, the network controlling the autonomous vehicle has to issue a braking command". We present our neuro-symbolic framework in Section 4, where we also discuss ways of obtaining specification networks in practice.

An essential feature of our framework is that it can be built on top of the existing verification infrastructure for neural networks. We demonstrate this fact in Section 5, where we describe a prototype implementation of a verifier for NeSAL properties based on the popular Marabou verifier [Katz *et al.*, 2019]. Specifically, we show that our prototype effectively verifies a wide range of neuro-symbolic properties. As targets for our verification, we have trained deep neural networks on the German Traffic Sign Recognition Benchmark (GTSRB) [Stallkamp *et al.*, 2011] and MNIST [LeCun *et al.*, 2010].

Finally, we want to highlight that our neuro-symbolic framework is general and not limited to deep neural networks. Instead, it can—in principle—be applied to any system that allows for suitable verification techniques, including differential models, hardware, and software. However, we leave an in-depth study of this promising direction to future work.

2 Related Work

Driven by the demand for trustworthy and reliable artificial intelligence, the formal verification of deep neural networks has become a very active and vibrant research area over the past five years (we refer the reader to a textbook [Albarghouthi, 2021] for a detailed overview). To the best of our knowledge, Seshia *et al.* [2018] conducted the first comprehensive survey of correctness properties arising in neural network verification. The authors classify these properties into several categories, including system-level specifications, robustness, fairness, semantic invariance, and monotonicity. However, we are not aware of any work proposing a neuro-symbolic verification approach, neither for deep neural networks or other differential models nor for hardware or software.

The key motivation of neuro-symbolic AI is to combine the advantages of symbolic and deep neural representations into a joint system [d'Avila Garcez *et al.*, 2009; d'Avila Garcez and Lamb, 2020]. This is often done in a hybrid fashion where a neural network acts as a perception module that interfaces with a symbolic reasoning system (e.g., see Yi *et al.* [2018]). The goal of such an approach is to mitigate the issues of one type of representation by the other (e.g., using the power of symbolic reasoning to handle the generalizability issues of neural networks and to handle the difficulty of noisy data for symbolic systems via neural networks). Recent work has also demonstrated the advantage of neuro-symbolic XAI [Stammer *et al.*, 2021] and commonsense reasoning [Arabshahi *et al.*, 2021]. The link to verification, however, has not been explored much. Indeed, Yang *et al.* [2021] explore symbolic propagation, but a higher-order specification framework does not exist.

To automatically verify correctness properties of deep neural networks, a host of distinct techniques have been proposed. The arguably most promising and, hence, most popular ones are *abstract interpretation* [Gehr *et al.*, 2018; Henriksen and Lomuscio, 2021] and *deductive verification* [Ehlers, 2017; Katz *et al.*, 2019] (we survey both in Section 3). In addition, various other approaches have been suggested. Examples include optimization-based methods [Gowal *et al.*, 2019], concolic testing [Sun *et al.*, 2018], and decomposition-based methods [Batten *et al.*, 2021]. While our neuro-symbolic framework is independent of the actual verification technique, this paper focuses on deductive verification.

3 Background on Neural Network Verification

Neural Network Verification is the task of formally proving that a deep neural network satisfies a semantic property (i.e., a property that refers to the semantic function computed by the network). To not clutter this section with too many technical details, let us illustrate this task through two popular examples: adversarial robustness and fairness. We will later formalize neural network verification in Section 4 when we introduce our neuro-symbolic verification framework.

In the case of adversarial robustness, one wants to prove that a neural network is robust to small perturbations of its inputs (i.e., that small changes to an input do not change the output). To make this mathematically precise, let us assume that we are given a multi-class neural network $f\colon \mathbb{R}^m \to \{c_1, \ldots, c_n\}$ with m features and n classes, a specific input $\vec{x}^\star \in \mathbb{R}^m$, a distance function $d\colon \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+$, and a distance $\varepsilon \geq 0$. Then, the task is to prove that

$$d(\vec{x}^\star, \vec{x}) \leq \varepsilon \text{ implies } f(\vec{x}^\star) = f(\vec{x}) \qquad (1)$$

for all inputs $\vec{x} \in \mathbb{R}^m$. In other words, the classes of $\vec{x}^\star$ and every input at most ε away from $\vec{x}^\star$ must coincide. An input $\vec{x} \in \mathbb{R}^m$ violating Property (1) is called an adversarial example and witnesses that f is not adversarially robust.

In the case of fairness, one wants to prove that the output of a neural network is not influenced by a sensitive feature such as sex or race. Again, let us assume that we are given a neural network $f\colon \mathbb{R}^m \to \mathbb{R}^n$ with m features, including a sensitive feature $i \in \{1, \ldots, m\}$. Then, the task is to prove that

$$x_i \neq x'_i \wedge \bigwedge_{j \neq i} x_j = x'_j \text{ implies } f(\vec{x}) = f(\vec{x}') \qquad (2)$$

for all pairs $\vec{x}, \vec{x}' \in \mathbb{R}^m$ of inputs with $\vec{x} = (x_1, \ldots, x_m)$ and $\vec{x}' = (x'_1, \ldots x'_m)$. In other words, if two inputs $\vec{x}$ and $\vec{x}'$ only differ on a sensitive feature, then the output of f must not change. Note that in the case of fairness, a counterexample consists of pairs $\vec{x}, \vec{x}'$ of inputs.

Properties (1) and (2) demonstrate a fundamental challenge of neural network verification: the task is to prove a property for *all* (usually infinitely many) inputs. Thus, cross-validation or other statistical approaches from machine learning are no longer sufficient because they test the network only on a finite number of inputs. Instead, one needs to employ methods that can reason symbolically about a given network.

Motivated by the success of modern software verification, a host of symbolic methods for the verification of neural networks have been proposed recently [Albarghouthi, 2021]. Among the two most popular are deductive verification [Ehlers, 2017; Katz *et al.*, 2019] and abstract interpretation [Gehr *et al.*, 2018; Singh *et al.*, 2018]. Let us briefly sketch both.

The key idea of deductive verification is to compile a deep neural network $f\colon \mathbb{R}^m \to \mathbb{R}^n$ together with a semantic property P into a logic formula $\psi_{f,P}$, called *verification condition*. This formula typically falls into the quantifier-free fragment of real arithmetic and is designed to be valid (i.e., satisfied by all inputs) if and only if f satisfies P. To show the validity of $\psi_{f,P}$, one checks whether its negation $\neg\psi_{f,P}$ is satisfiable. This can be done either with the help of an off-the-shelf Satisfiability Modulo Theory solver (such as Z3 [de Moura and

Bjørner, 2008]) or using one of the recently proposed, specialized constraint solvers such as Planet [Ehlers, 2017] or Marabou [Katz *et al.*, 2019]. If $\neg\psi_{f,P}$ is unsatisfiable, then $\psi_{f,P}$ is valid, and—by construction—f satisfies P. If $\neg\psi_{f,P}$ is satisfiable, on the other hand, then $\psi_{f,P}$ is not valid, implying that f violates the property P. In the latter case, most constraint solvers (including the ones mentioned above) can produce an assignment satisfying $\neg\psi_{f,P}$, which can then be used to extract inputs to f that witness a violation of P.

Abstract interpretation is a mathematical framework for computing sound and precise approximations of the semantics of software and other complex systems [Cousot and Cousot, 1977]. When applied to neural network verification, the basic idea is to over-approximate the computation of a deep neural network on an infinite set of inputs. Each such infinite set is symbolically represented by an element of a so-called *abstract domain*, which consists of logical formulas capturing shapes such as n-dimensional boxes, polytopes, or zonotopes. To approximate a network's computation, an element of the abstract domain is propagated through the layers of the network. Since layers operate on concrete values and not abstract elements, this propagation requires replacing each layer with an abstract one (called abstract transformer) that computes the effects of the layer on abstract elements. Thus, when given an abstract element A in the input space of a network f (e.g., representing the neighborhood of a fixed input $\vec{x}^\star$), the result of abstract interpretation is an abstract element A' in the output space over-approximating all outputs $f(x)$ of concrete inputs $x \in A$. To verify that a property P holds, it is then enough to check whether A' is included in an abstract element representing all outputs satisfying P. Since abstract interpretation computes over-approximations of the actual input-output behavior of a network, the property P typically describes a safety condition.

While neural network verification is a vibrant and highly active field, virtually all existing research suffers from three substantial shortcomings:

1. Existing research focuses on verifying "simple" properties that can be formalized using quantifier-free first-order constraints on the inputs and outputs of a network. Examples of such properties include adversarial robustness and fairness, illustrated by Properties (1) and (2) above. However, the overwhelming majority of relevant correctness properties cannot be expressed in this simple way. As an example, consider a neural network controlling an autonomous car and the property that the car needs to decelerate as soon as a stop sign appears in the front view. It is clear that formalizing this property is extremely hard: it would require us to mathematically capture all essential features of all possible stop signs, including their position, shape, angle, color, etc.

2. Virtually all properties considered in neural network verification today are either *local* (referring to inputs in the neighborhood of an a priori fixed input $\vec{x}^\star$) or *global* (referring to all inputs). Adversarial robustness is an example of the former type, while fairness illustrates the latter. However, a more natural and helpful approach would be to restrict the verification to inputs from the underlying data distribution since we do typically not expect our networks to process out-of-distribution data. Again, such a restriction is very hard to capture logically and, therefore, not featured by current methods.

3. A fundamental problem, especially when verifying global properties, is that counterexamples (i.e., inputs witnessing the violation of the property) are often out of distribution and, hence, of little value. Again, restricting the verification to inputs originating from the underlying data distribution would mitigate this issue but is not supported by current approaches.

In the next section, we address these drawbacks by introducing a neuro-symbolic framework for neural network verification.

4 A Neuro-Symbolic Verification Framework

As illustrated by Properties (1) and (2), the primary obstacle in today's neural network verification is that correctness properties have to be formalized in a suitable—often relatively simple—logical formalism that relates inputs and outputs of a neural network (e.g., the quantifier-free fragment of real arithmetic). This requirement fundamentally limits current verification approaches to only a few different types of correctness properties, arguably making them ill-equipped to tackle real-world AI verification tasks.

As a first step towards overcoming this severe practical limitation, we propose a neuro-symbolic approach to neural network verification. Our main idea is seemingly simple yet powerful: we propose the use of highly specialized deep neural networks, named *specification networks*, as proxy objects for capturing semantic correctness properties. We introduce the concept of specification networks and possible ways of how to obtain them in Section 4.1. In Section 4.2, we then propose a fragment of quantifier-free first-order logic to formalize correctness properties involving specification networks. We call this type of properties *neuro-symbolic* and the resulting assertion language NeSAL. In an extended version of this paper [Xie *et al.*, 2022], we demonstrate how checking NeSAL properties can be reduced to traditional deductive verification of deep neural networks. This reduction allows utilizing any existing deductive verifier (e.g., Planet [Ehlers, 2017] or Marabou [Katz *et al.*, 2019]), making our neuro-symbolic verification framework easily accessible to researchers and practitioners alike. The key idea is that—on the level of logic formulas—it does not matter whether a verification condition involves one or multiple deep neural networks.

4.1 Specification Networks

Generally speaking, a specification network is a highly specialized deep neural network trained for a specific task (e.g., perception, anomaly detection, recognizing the underlying data distribution, etc.). We use one (or multiple) of such networks as proxy objects to capture correctness properties. Their precise architecture does not matter at this point, but might influence the choice of which verification engine to use.

Let us illustrate the concept of specification networks using the autonomous vehicle example from Section 3. For the sake of simplicity, let us assume that we are given

- a deep neural network f that takes pictures $\vec{x}$ from the front camera as input and outputs the steering commands "left", "right", "accelerate", and "decelerate"; and
- a property P stating "f has to issue a deceleration command as soon as a stop sign appears in the front camera".

Instead of trying to formalize all characteristics of stop signs in logic (i.e., their possible positions, shapes, colors, etc.), we now train a second deep neural network g for the specific perception task of recognizing stop signs. Assuming that g is a binary-class network (outputting "yes" if it detects a stop sign in the image $\vec{x}$ and "no" otherwise), one can then express the property P above straightforwardly as

$$\text{if } g(\vec{x}) = \text{yes, then } f(\vec{x}) = \text{decelerate.} \tag{3}$$

Note that our original property P now amounts to a simple constraint over the inputs and outputs of the networks f and g.

An essential requirement of our framework is the availability of adequate specification networks. We here sketch three conceivable ways of how to obtain them:

1. The perhaps simplest way of obtaining specification networks is to train them explicitly. To avoid systematic errors, it is crucial to train a specification network on a dataset that is different from the one used to train the network under verification. Preferably, one should additionally use a different architecture and hyperparameters.
2. Similar to standard datasets such as MNIST [LeCun *et al.*, 2010], researchers and interested companies might create public repositories for specification networks (such as Hugging Face[1]). To boot-strap such efforts, we have made the specification networks used in our experimental evaluation available on GitHub.[2]
3. Finally, regulatory bodies or notified bodies might provide specification networks as references for future AI-enabled systems. Such an approach can be used, for instance, to guarantee minimum standards for the correctness and reliability of deep neural networks in safety-critical applications.

4.2 A Neuro-Symbolic Assertion Language

Inspired by neuro-symbolic reasoning [d'Avila Garcez *et al.*, 2019; De Raedt *et al.*, 2020], we now describe how to use specification networks to formalize correctness properties of neural networks. Specifically, we introduce an assertion language, named Neuro-Symbolic Assertion Language, which is inspired by the Hoare logic used in software verification and adapts the notation introduced by Albarghouthi [Albarghouthi, 2021]. This language is a fragment of the quantifier-free first-order logic over the reals and allows formalizing complex correctness properties—involving multiple specification networks—in an interpretable and straightforward manner.

Throughout the remainder of this paper, we assume that we are given $k \in \mathbb{N}$ specification networks $g_1, \ldots, g_k$ with $g_i \colon \mathbb{R}^{m_i} \to \mathbb{R}^{n_i}$ for $i \in \{1, \ldots, k\}$. Moreover, let us assume that we want to formalize a correctness property for a single deep neural network $f \colon \mathbb{R}^{m_0} \to \mathbb{R}^{n_0}$, which we call the *network under verification (NUV)*. Note that the latter assumption is not a restriction of our framework, but it simplifies the following presentation. Our framework can easily be extended to multiple networks under verification.

Let us now turn to the definition of our *Neuro-Symbolic Assertion Language (*NeSAL*)*. Formally, NeSAL is the quantifier-free fragment of first-order logic over the reals that contains all logic formulas of the form

$$\begin{gathered}\big\{\varphi_{pre}(\vec{x}_1, \ldots, \vec{x}_\ell)\big\}\\ \vec{y}_1 \leftarrow h_1(\vec{x}) \wedge \cdots \wedge \vec{y}_\ell \leftarrow h_\ell(\vec{x}_\ell)\\ \big\{\varphi_{post}(\vec{x}_1, \ldots, \vec{x}_\ell, \vec{y}_1, \ldots, \vec{y}_\ell)\big\},\end{gathered}$$

where

- $h_i \in \{f, g_1, \ldots, g_k\}$ for $i \in \{1, \ldots, \ell\}$ are function symbols representing the given neural networks, one of which is the NUV f;
- $\vec{x}_1, \ldots, \vec{x}_\ell$ are vectors of real variables representing the input values of the networks $h_i \in \{f, g_1, \ldots, g_k\}$;
- $\vec{y}_1, \ldots, \vec{y}_\ell$ are vectors of real variables representing the output values of the networks $h_i \in \{f, g_1, \ldots, g_k\}$;
- the expressions $\vec{y}_i \leftarrow h_i(\vec{x}_i)$ store the results of the computations $h_i(\vec{x}_i)$ in the variables $\vec{y}_i$, where we assume that $\vec{x}_i$ and $\vec{y}_i$ match the dimensions of the input and output space of h_i, respectively;
- φ_{pre} is a quantifier-free first-order formula over the free variables $\vec{x}_1, \ldots, \vec{x}_\ell$, called *pre-condition*, expressing constraints on the inputs to the networks $f, g_1, \ldots, g_k$;
- φ_{post} is a quantifier-free first-order formula over the free variables $\vec{x}_1, \ldots, \vec{x}_\ell$ and $\vec{y}_1, \ldots, \vec{y}_\ell$, called *post-condition*, expressing desired properties of f while considering the computations of $g_1, \ldots, g_k$.

We call each such formula a *neuro-symbolic property* to emphasize that correctness properties are no longer restricted to simple first-order constraints on the inputs and outputs of the NUV but can also depend on other networks.

The intuitive meaning of a neuro-symbolic property is that if the inputs $\vec{x}_1, \ldots, \vec{x}_\ell$ satisfy φ_{pre} and the output of the networks on these inputs is $\vec{y}_1, \ldots, \vec{y}_\ell$, then φ_{post} has to be satisfied as well. Let us illustrate this definition with our example of Section 4.1. In this example, we are given a NUV $f \colon \mathbb{R}^{m \times m} \to \big\{\text{left}, \text{right}, \text{accelerate}, \text{decelerate}\big\}$ mapping $m \times m$ pixel images to steering commands and a single specification network $g \colon \mathbb{R}^{m \times m} \to \big\{\text{yes}, \text{no}\big\}$ detecting stop signs. Then, Property (3) can be formalized in NeSAL as

$$\begin{gathered}\big\{\vec{x}_1 = \vec{x}_2\big\}\\ y_1 \leftarrow f(\vec{x}_1) \wedge y_2 \leftarrow g(\vec{x}_2)\\ \big\{y_2 = \text{yes} \rightarrow y_1 = \text{decelerate}\big\}.\end{gathered}$$

This neuro-symbolic property is a prototypical example of how our approach mitigates the first shortcoming of classical neural network verification discussed in Section 3. To address the second and third shortcomings, we can train an autoencoder $g \colon \mathbb{R}^m \to \mathbb{R}^m$ to capture the distribution underlying the training data. To restrict the verification of a network

[1]https://huggingface.co

[2]https://github.com/LebronX/Neuro-Symbolic-Verification

$f\colon \mathbb{R}^m \to \mathbb{R}^n$ to the underlying data distribution, we can use the neuro-symbolic property

$$\{\mathit{true}\}\ \vec{y}_1 \leftarrow f(\vec{x}) \wedge \vec{y}_2 \leftarrow g(\vec{x})\ \{d(\vec{x}, \vec{y}_2) \le \varepsilon \to P(\vec{x}, \vec{y}_1)\}$$

where P is the original property we want to verify and the condition $d(\vec{x}, \vec{y}_2) \le \varepsilon$ for some $\varepsilon \ge 0$ follows the usual idea of using autoencoders to detect out-of-distribution data [Sakurada and Yairi, 2014]. As a byproduct, we obtain that any counterexample to this new property violates the original property P and originates from the underlying data distribution.

It is not hard to verify that "simple" properties, such as adversarial robustness and fairness, can easily be expressed as neuro-symbolic properties as well. For instance, adversarial robustness can be formalized in NeSAL as

$$\{d(\vec{x}^\star, \vec{x}) \le \varepsilon\}\ \vec{y}^\star \leftarrow f(\vec{x}^\star) \wedge \vec{y} \leftarrow f(\vec{x})\ \{\vec{y}^\star = \vec{y}\}$$

where $\vec{x}^\star \in \mathbb{R}^m$ is a fixed input, $\epsilon \ge 0$, and assuming that the distance function d can be expressed in the quantifier-free fragment of first-order logic over the reals. Note that individual networks can appear multiple times in a neuro-symbolic property.

Given a neuro-symbolic property $\{\varphi_{pre}\}\ \varphi_{assign}\ \{\varphi_{post}\}$ with $\varphi_{assign} := \bigwedge_{i=1}^{\ell} \vec{y}_i \leftarrow h_i(\vec{x}_i)$, the overall goal is to check whether the logic formula

$$\psi := (\varphi_{pre} \wedge \varphi_{assign}) \to \varphi_{post}$$

is valid (i.e., whether $\forall \vec{x}, \vec{y}\colon \psi$ is a tautology). In analogy to software verification, we call this task the *neuro-symbolic verification problem* and the formula ψ a *neuro-symbolic verification condition*. The supplementary material shows how this verification problem can be reduced to deductive verification.

5 Empirical Evaluation

We have implemented a Python3 prototype based on the state-of-the-art deductive verifier Marabou [Katz *et al.*, 2019], named *Neuro-Symbolic Verifier (*NSV*)*. Since the original Marabou tool does not support NeSAL properties—or any verification query with multiple networks—, we have modified it as described in the extended version of this paper [Xie *et al.*, 2022]. Our code and all experimental data can be found at https://github.com/LebronX/Neuro-Symbolic-Verification.

In our experimental evaluation, we have considered two widely used datasets:

1. The MNIST dataset [LeCun *et al.*, 2010], containing 60,000 training images and 10.000 test images of handwritten digits.
2. The German Traffic Sign Recognition Benchmark (GTSRB) [Stallkamp *et al.*, 2011], containing 39,209 training images and 12,630 test images with 43 types of German traffic signs. To not repeat similar experiments over and over, we restrict ourselves to the first ten classes.

It is paramount to stress that we are not interested in the absolute performance of NSV, how well it scales to huge networks, or how it compares to other verification techniques on non-neuro-symbolic properties. Instead, our goals are twofold: (1) we demonstrate that our neuro-symbolic approach can be implemented on top of existing verification infrastructure for deep neural networks and is effective in verifying neuro-symbolic properties; and (2) we showcase that our neuro-symbolic framework can find more informative counterexamples than a purely deductive verification approach. Note that the former makes it possible to leverage future progress in neural network verification to our neuro-symbolic setting, while the latter greatly improves the debugging of learning systems.

Effectiveness of Verifying Neuro-Symbolic Properties For both the GTSRB and the MNIST datasets, we considered the following three prototypical neuro-symbolic properties. By convention, we use f to denote the network under verification (NUV) and g to denote a specification network. Moreover, we use the L_∞-norm as distance function d.

P_1: "If the input image is of class c, then the NUV outputs c", expressed in NeSAL as

$$\{\mathit{true}\}$$
$$\vec{y}_1 \leftarrow f(\vec{x}) \wedge y_2 \leftarrow g(\vec{x})$$
$$\{y_2 = 1 \to \arg\max(\vec{y}_1) = c\}.$$

Here, the NUV f is a multi-class deep neural network mapping images to their class (i.e., one of the 43 traffic signs or ten digits), while the specification network g is a deep neural network specifically trained by an authority to detect the specific class c (outputting 0 or 1).[3]

P_2: "If the input follows the distribution of the underlying data, then the NUV classifies the input correctly with high confidence", expressed in NeSAL as

$$\{\mathit{true}\}$$
$$\vec{y}_1 \leftarrow f(\vec{x}) \wedge \vec{y}_2 \leftarrow g(\vec{x})$$
$$\{(d(\vec{y}_2, \vec{x}) \le \varepsilon \wedge \arg\max(\vec{y}_1) = c) \to \mathit{conf} > \delta\},$$

where $\varepsilon, \delta \ge 0$ and $\mathit{conf} := (|\vec{y}_1| \cdot y_i - \sum_{j \ne i} y_j)/|\vec{y}_1|$ is the confidence of the NUV that the input $\vec{x}$ is of class c. Here, the NUV f is a multi-class deep neural network mapping images to their class, while the specification network g is an autoencoder used to detect out-distribution data (via $d(\vec{y}_2, \vec{x}) > \varepsilon$) [Sakurada and Yairi, 2014].

P_3: "Two deep neural networks (of different architecture) compute the same function up to a maximum error of ε", expressed in NeSAL as

$$\{\mathit{true}\}\ \vec{y}_1 \leftarrow f(\vec{x}) \wedge \vec{y}_2 \leftarrow g(\vec{x})\ \{d(\vec{y}_1, \vec{y}_2) \le \varepsilon\},$$

where $\varepsilon \ge 0$. Here, the NUV f and the specification network g have the same dimensions of the input and output space but potentially distinct architectures.

For each benchmark suite, each class (remember that we have only considered ten classes of GTSRB), and each of the three properties, we have trained one NUV and one specification network with the architectures shown in Table 1 (all using ReLU activation functions). We have resized the MNIST images for property P_2 to 14×14 and the GTSRB images to 16×16 for all properties to keep the verification tractable. For property P_2, we have chosen $0.05 \le \varepsilon \le 0.14$ with step size 0.01 and $1 \le \delta \le 20$ with step size 1. For property P_3, we have chosen $0.05 \le \varepsilon \le 0.14$ with step size 0.01.

[3]This property is a simplified version of Property (3) on Page 4.

Property and benchmarks	NUV in	NUV out	NUV hid.	Spec. network in	Spec. network out	Spec. network hid.
P_1-MNIST	784	10	$3 \cdot 10$	784	2	$3 \cdot 10$
P_2-MNIST	196	10	$1 \cdot 10$	196	196	$1 \cdot 10$
P_3-MNIST	784	10	$3 \cdot 20$	784	10	$3 \cdot 20$
P_1-GTSRB	256	43	$3 \cdot 12$	256	2	$3 \cdot 5$
P_2-GTSRB	256	43	$1 \cdot 10$	256	256	$1 \cdot 10$
P_3-GTSRB	256	43	$3 \cdot 35$	256	43	$3 \cdot 35$

Table 1: Architectures used in our experimental evaluation. Each line lists the number of neurons in the input layer ("in"), output layer ("out"), and hidden layers ("hid."), respectively.

To avoid statistical anomalies, we have repeated all experiments five times with different neural networks (trained using different parameters) and report the average results. This way, we obtained $5 \cdot 2 \cdot (10 + 200 + 10) = 2,200$ verification tasks in total. We have conducted our evaluation on an Intel Core i5-5350U CPU (1.80 GHz) with 8 GB RAM running MacOS Catalina 10.15.7 with a timeout of $1,800\,s$ per benchmark.

Figure 1 depicts the results of our experiments in terms of the accumulated average runtimes. On the MNIST benchmark suite, NSV timed out on one benchmark (for property P_1) and terminated on all others. It found a counterexample in all cases (i.e., none of the NUVs satisfied the properties). On the GTSRB suite, NSV always terminated. It proved that all NUVs satisfied property P_1, while finding counterexamples for all benchmarks of properties P_2 and P_3. Note that the single timeout on the MNIST benchmark suite causes the steep increase in the graph of property P_1 on the left of Figure 1. To avoid any bias, we have not taken measures during the training process to ensure that our NUVs satisfy any of the properties, which explains the large number of counterexamples. We believe that the relatively low resolution of the GTSRB images caused property P_1 to be vacuously true since the specification network g did not detect the correct class (i.e., $y_2 \neq 1$).

In total, our experiments show that NSV is effective at verifying a diverse set of neuro-symbolic properties. The fact that it was built on top of existing infrastructure shows that our neuro-symbolic framework is easy to adopt in practice.

Quality of Counterexamples

To assess the quality of the counterexamples generated by NSV, we have modified Property P_2 to exclude the requirement that the data must come from the underlying distribution.

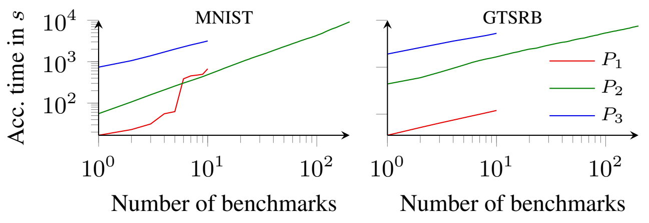

Figure 1: Accumulated average runtimes for the experiments on the MNIST dataset (left) and the GTSRB dataset (right)

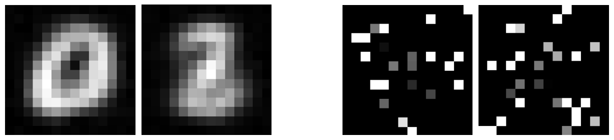

Figure 2: Counterexamples to the neuro-symbolic property P_2 (left) and the corresponding non-neuro-symbolic property P_2' (right)

The resulting property P_2', expressed in NeSAL, is

$$\{true\}\ \vec{y}_1 \leftarrow f(\vec{x})\ \{\arg\max(\vec{y}_1) = c \rightarrow conf > \delta\}.$$

Note that P_2' involves only one network and represents a typical property arising in classical neural network verification.

We have verified Property P_2' on the deep neural networks from the MNIST dataset using the original Marabou framework. The result is shown on the right-hand-side of Figure 2. Since Property P_2' is global, the verification has to consider all possible inputs. As Figure 2 demonstrates, counterexamples to such properties are often random noise and arguably of little value. In fact, we could not identify a single counterexample that looked close to the original dataset.

By contrast, the left-hand-side of Figure 2 shows two counterexamples to the neuro-symbolic property P_2. These counterexamples are substantially more meaningful and intuitive because they originate from the underlying distribution of the data (as captured by an autoencoder trained to reconstruct the data). This demonstrates that neuro-symbolic verification produces meaningful counterexamples that can greatly simplify the development and debugging of learning systems.

6 Conclusion and Future Work

We have introduced the first neuro-symbolic framework for neural network verification, which allows expressing complex correctness properties through deep neural networks. We have demonstrated that our framework can straightforwardly be implemented on top of existing verification infrastructure and provides more informative counterexamples as existing methods. To the best of our knowledge, we are the first to propose a neuro-symbolic approach to formal verification.

The concept of neuro-symbolic verification can, in principle, also be applied to hardware and software verification (e.g., to express properties involving perception), and we believe that this is a promising direction of future work. Another essential future task will be to develop novel verification algorithms (e.g., based on abstract interpretation) that exploit the neuro-symbolic nature of correctness properties and can reason about multiple, inter-depending deep neural networks. To further improve scalability, we also intend to investigate neuro-symbolic approaches to testing.

Acknowledgements

This work benefited from the National Research Center for Applied Cybersecurity ATHENE, a joint effort of the German Ministry of Education and Research (BMBF) and the Hessian Ministry of Higher Education, Research, Science and the Arts (HMWK), from the HMWK cluster project "The Third Wave of AI", as well as the Hessian research priority programme LOEWE within the project WhiteBox.

References

[Albarghouthi, 2021] Aws Albarghouthi. *Introduction to Neural Network Verification*. verifieddeeplearning.com, 2021.

[Arabshahi *et al.*, 2021] Forough Arabshahi, Jennifer Lee, Mikayla Gawarecki, Kathryn Mazaitis, Amos Azaria, and Tom M. Mitchell. Conversational neuro-symbolic commonsense reasoning. In *AAAI*, pages 4902–4911, 2021.

[Batten *et al.*, 2021] Ben Batten, Panagiotis Kouvaros, Alessio Lomuscio, and Yang Zheng. Efficient neural network verification via layer-based semidefinite relaxations and linear cuts. In *IJCAI*, pages 2184–2190, 2021.

[Cousot and Cousot, 1977] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approx. of fixpoints. In *POPL*, pages 238–252, 1977.

[d'Avila Garcez and Lamb, 2020] Artur d'Avila Garcez and Luís C. Lamb. Neurosymbolic AI: the 3rd wave. *CoRR*, abs/2012.05876, 2020.

[d'Avila Garcez *et al.*, 2009] Artur S. d'Avila Garcez, Luís C. Lamb, and Dov M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer, 2009.

[d'Avila Garcez *et al.*, 2019] Artur S. d'Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *FLAP*, 6(4):611–632, 2019.

[de Moura and Bjørner, 2008] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340, 2008.

[De Raedt *et al.*, 2020] Luc De Raedt, Sebastijan Dumancic, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence. In *IJCAI*, pages 4943–4950, 2020.

[Ehlers, 2017] Rüdiger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *ATVA*, volume 10482 of *LNCS*, pages 269–286, 2017.

[Gehr *et al.*, 2018] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. AI2: safety and robustness certification of neural networks with abstract interpretation. In *SP*, pages 3–18, 2018.

[Gowal *et al.*, 2019] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Timothy A. Mann, and Pushmeet Kohli. A dual approach to verify and train deep networks. In *IJCAI*, pages 6156–6160, 2019.

[Henriksen and Lomuscio, 2021] Patrick Henriksen and Alessio Lomuscio. DEEPSPLIT: an efficient splitting method for neural network verification via indirect effect analysis. In *IJCAI*, pages 2549–2555, 2021.

[Katz *et al.*, 2019] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljic, David L. Dill, Mykel J. Kochenderfer, and Clark W. Barrett. The marabou framework for verification and analysis of deep neural networks. In *CAV*, volume 11561 of *LNCS*, pages 443–452, 2019.

[LeCun *et al.*, 2010] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[McGregor, 2021] Sean McGregor. Preventing repeated real world AI failures by cataloging incidents: The AI incident database. In *AAAI*, pages 15458–15463, 2021.

[Sakurada and Yairi, 2014] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with non-linear dimensionality reduction. In *MLSDA*, page 4, 2014.

[Seshia *et al.*, 2018] Sanjit A. Seshia, Ankush Desai, Tommaso Dreossi, Daniel J. Fremont, Shromona Ghosh, Edward Kim, Sumukh Shivakumar, Marcell Vazquez-Chanlatte, and Xiangyu Yue. Formal specification for deep neural networks. In *ATVA*, volume 11138 of *LNCS*, pages 20–34, 2018.

[Singh *et al.*, 2018] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. Fast and effective robustness certification. In *NeurIPS*, pages 10825–10836, 2018.

[Stallkamp *et al.*, 2011] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *IJCNN*, pages 1453–1460, 2011.

[Stammer *et al.*, 2021] Wolfgang Stammer, Patrick Schramowski, and Kristian Kersting. Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. In *CVPR*, pages 3619–3629, 2021.

[Sun *et al.*, 2018] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. Concolic testing for deep neural networks. In *ASE*, pages 109–119, 2018.

[Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[Xie *et al.*, 2022] Xuan Xie, Kristian Kersting, and Daniel Neider. Neuro-symbolic verification of deep neural networks. *CoRR*, abs/2203.00938, 2022.

[Yang *et al.*, 2021] Pengfei Yang, Jianlin Li, Jiangchao Liu, Cheng-Chao Huang, Renjue Li, Liqian Chen, Xiaowei Huang, and Lijun Zhang. Enhancing robustness verification for deep neural networks via symbolic propagation. *Formal Aspects Comput.*, 33(3):407–435, 2021.

[Yi *et al.*, 2018] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic VQA: disentangling reasoning from vision and language understanding. In *NeurIPS*, pages 1039–1050, 2018.