When to Trust AI: Advances and Challenges for Certification of Neural Networks

Marta Kwiatkowska, Xiyue Zhang
0000-0001-9022-7599
0000-0003-1649-7165
University of Oxford, UK
Email: {marta.kwiatkowska, xiyue.zhang}@cs.ox.ac.uk

***Abstract*—Artificial intelligence (AI) has been advancing at a fast pace and it is now poised for deployment in a wide range of applications, such as autonomous systems, medical diagnosis and natural language processing. Early adoption of AI technology for real-world applications has not been without problems, particularly for neural networks, which may be unstable and susceptible to adversarial examples. In the longer term, appropriate safety assurance techniques need to be developed to reduce potential harm due to avoidable system failures and ensure trustworthiness. Focusing on certification and explainability, this paper provides an overview of techniques that have been developed to ensure safety of AI decisions and discusses future challenges.**

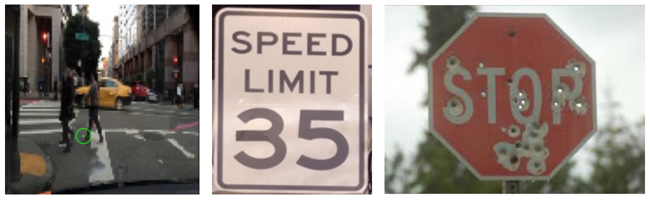


Fig. 1: Challenges of safe traffic sign recognition. Single-pixel adversarial attack from [1] (left), physical attack (middle) and a real traffic sign (right).

I. Introduction

Artificial intelligence (AI) has advanced significantly in recent years, largely due to the step improvement enabled by deep learning in data-rich tasks such as computer vision or natural language processing. AI technologies are being widely deployed and enthusiastically embraced by the public, as is evident from the take up of ChatGPT and Tesla. However, deep learning lacks robustness, and neural networks (NNs), in particular, are unstable with respect to so called *adversarial perturbations*, often imperceptible modifications to inputs that can drastically change the network's decision. Many such examples have been reported in the literature and the media. Figure 1 (left) shows a dashboard camera image from [1], for which a change of a single pixel to green changes the classification of the image from red traffic light to green, which is potentially unsafe if there is no fallback safety measure; while this is arguably an artificial example, some modern cars have been observed to mis-read traffic signs, including the physical attack in Figure 1 (middle), where the digit 3 has been modified. Traffic sign recognition is a complex problem to specify and solve, see Figure 1 (right), which shows a real traffic sign in Alaska. As with any maturing technology, it is natural to ask if AI is ready for wide deployment, and what steps – scientific, methodological, regulatory, or societal – can be taken to achieve its trustworthiness and reduce potential for harm through rushed roll-out. This is particularly important given the fast-paced development of AI technologies and the natural propensity of humans to overtrust automation.

For AI to be trusted, particularly in high-stakes situations, where avoidable failure or wrong decision can lead to harm or high cost being incurred, it is essential to provide *provable guarantees* on the critical decisions taken autonomously by the system. Traditionally, for software systems this has been achieved with *formal verification* techniques, which aim to formally prove whether the system satisfies a given specification, and if not provide a diagnostic counter-example. Founded on logic, automated verification, also known as model checking, achieves this goal by means of executing a verification algorithm on a suitably encoded model of the system. Software verification has become an established methodology and a variety of tools of industrial relevance are employed in application domains such as distributed computation, security protocols or hardware. Beginning with [2], [3], over the past few years a number of formal verification techniques have been adapted to neural networks, which are fully data-driven and significantly differ from the state-based transition system models of conventional software, and have given rise to practical, algorithmic techniques that provide provable guarantees on neural network decisions [4].

This paper aims to provide an overview of existing techniques that can be used to increase trust in AI systems and outline future scientific challenges, while at the same time raising awareness of potential risks with early adoption. It is taken as granted that safety assurance of AI systems is complex and needs to involve appropriately regulated processes and assignment of accountability. The topics discussed in this paper are by no means exhaustive, but offer a representative selection of techniques and tools that can be used within such safety assurances processes, and can be adapted, extended or built upon to increase robustness and trustworthiness of AI systems. The paper will focus on highlighting the following two aspects:

- **Certification**: focusing on individual decisions (possibly critical to the integrity of the system) that are made by neural networks, we provide an overview of the main methodological approaches and techniques that have been developed to obtain provable guarantees on the correctness of the decision, which can thus be used for certification. The sources of computational complexity of neural network verification will be discussed, as well as limitations of existing methods and ways to address them.
- **Explainability**: neural networks are 'black boxes' that are trained from data using obscure optimization processes and objectives, and it is argued that users of AI systems will benefit from the ability to obtain explanations for the decisions. We summarise the main approaches to producing explanations and discuss that they may lack robustness and how this issue can be addressed.

The overview includes high-level description of main algorithms, which are illustrated by worked examples to explain their behaviour to the interested reader. This is followed by a selection of case studies of robustness analysis and/or certification drawn from a variety of application domains, with the aim to highlight the strengths and weaknesses of the approaches. Finally, future challenges and suggestions for fruitful directions to guide the developments in this actively studied and important area will be outlined.

The paper is organised as follows. Section II introduces the main concepts, focusing on neural networks in the supervised learning setting. Section III provides an overview of the main (forward and backward) analysis approaches, with a description of the working for a selection of algorithms illustrated by worked examples. Section IV includes a few excerpts from a selection of verification and certification experiments, aimed at highlighting the uses of the main methods, and Section V outlines future challenges. Finally, Section VI concludes the paper.

II. Safety, Robustness and Explainability

In the context of safety-critical systems, safety assurance techniques aim to prevent, or minimise the probability of, a hazard occurring, and appropriate safety measures are invoked in case of failures. In this paper, we focus on critical decisions made by neural networks, which we informally refer to as safe if they satisfy a given property, which can be shown or disproved by formal verification. Before discussing formal verification techniques, we begin with background introduction to the main concepts of deterministic neural networks, their (local) robustness and explanations.

A. Neural Networks

We consider neural networks in the supervised learning setting. A neural network is a function $f : \mathbb{R}^n \to \mathbb{R}^m$ mapping from the input space to the output space, which is typically trained based on a dataset $\mathcal{D}$ of pairs (x, y) of input x and ground truth label y. A neural network consisting of $L+1$ layers (including the input layer) can be characterized by a

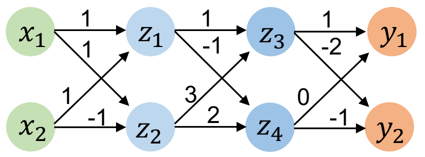


Fig. 2: A feed-forward neural network.

set of matrices $\{W^{(i)}\}_{i=1}^{L}$ and bias vectors $\{b^{(i)}\}_{i=1}^{L}$ for linear (affine) transformations, followed by pointwise activation functions, such as $ReLU$, $Sigmoid$, and $Tanh$, for nonlinear transformations. We use $\hat{z}^{(i)}$ and $z^{(i)}$ to denote the pre-activated and activated vectors of the i-th layer, respectively. The layer-by-layer forward computation of neural networks can be described as follows:

- *Linear transformation.* The linear transformation generates a pre-activated vector $\hat{z}^{(i)} = W^{(i)} \cdot z^{(i-1)} + b^{(i)}$ ($i \in [1, L]$) from the output of the previous layer, and $z^{(0)} = x$ denotes the input vector.
- *Pointwise nonlinear transformation.* The pointwise nonlinear transformation generates the activation vector $z^{(i)} = \sigma(\hat{z}^{(i)})$ ($i \in [1, L]$). In practice, $softmax$ is usually employed as the activation function for the output layer in classification tasks, which provides the normalised relative probabilities of classifying the input into each label.

Given an input $x \in \mathbb{R}^n$, the output of f on x is defined by $f(x) = f^{(L)} \circ \cdots \circ f^{(1)}(x)$, where $f^{(i)}$ denotes the mapping function of the i-th layer, which is the composition of linear and pointwise nonlinear transformations.

Example 1. *Figure 2 shows a simple feed-forward (and fully connected (FC)) neural network with four layers and ReLU as the activation function. x_1, x_2 represent two input neurons. z_1, z_2 and z_3, z_4 represent the activated neurons of the two hidden layers. y_1, y_2 are two output neurons. The forward computation from the input layer to the output layer is as follows (ReLU is denoted as σ).*

$$z_1 = \sigma(x_1 + x_2), \quad z_2 = \sigma(x_1 - x_2) \tag{1}$$

$$z_3 = \sigma(z_1 + 3z_2), \quad z_4 = \sigma(-z_1 + 2z_2) \tag{2}$$

$$y_1 = z_3, \qquad y_2 = -2z_3 - z_4 \tag{3}$$

We will use this neural network as a running example to illustrate different problem formulations and methods to address them.

B. Robustness

Robustness focuses on neural networks' resilience to adversarial attacks, noisy input data, etc., at test time, known as evasion attacks [6], [7], [8]. Attacks at training time are known as poisoning attacks [9], which have been omitted from this overview.

Adversarial robustness [7] of neural networks formalizes the desirable property that a well-trained model makes consistent predictions when its input data point is subjected to

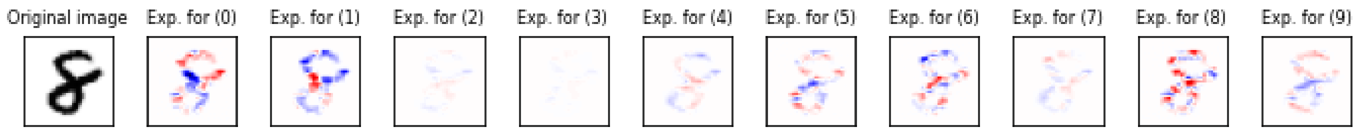


Fig. 3: The IG explanation for each of the classes of the MNIST dataset, where red indicates a positive contribution and blue a negative. Figure taken from [5].

small adversarial perturbations. *Local (adversarial) robustness* pertains to a given input point x with ground truth label y, and is usually defined in terms of invariance of the network's decision within a small neighbourhood $\mathbb{B}_p(x, \epsilon)$ of x, for a class of perturbations bounded by ϵ with respect to the ℓ_p norm.

Definition 1. *Given a (deterministic) neural network f, a labelled input data point (x, y), and a perturbation bound ϵ, the local robustness property of f on x is defined as*

$$\forall x' \in \mathbb{B}_p(x, \epsilon). \operatorname*{arg\,max}_{i=1,\cdots,m} f_i(x') = y,$$

where $\mathbb{B}_p(x, \epsilon)$ denotes the adversarial ℓ_p-ball of radius ϵ around input x.

Should there exist a point x' in the neighbourhood whose class is different than y, it is referred to as an *adversarial example*.

A related concept is that of a *maximal safe radius (MSR)* [10], denoted $MSR(x)$, which is the minimum distance from $x \in \mathbb{R}^n$ to the decision boundary, and is defined as the largest $\epsilon > 0$ such that $\forall x' \in \mathbb{B}_p(x, \epsilon). \arg\max_{i=1,\cdots,m} f_i(x') = \arg\max_{i=1,\cdots,m} f_i(x)$. Computing the value of MSR, say γ, provides a *guarantee* that the decision is robust (safe) for perturbations up to γ. On the other hand, finding an adversarial example at distance γ' is witness to the failure of robustness.

Global robustness [11] concerns the stability of predictions over the whole input space and is omitted.

C. Explainability

Explainability [12], [13] aims to understand and interpret why a given neural network makes certain predictions. The term explainability is often used interchangeably with interpretability in the literature, though interpretability usually refers to explaining how the model works. In this overview, we focus on local (pointwise) explainability for an individual *model decision*, which is categorised into *feature attribution* methods, which heuristically estimate feature attribution scores for model predictions and include *gradient-based* [14], [15] and *perturbation-based* techniques [16], [17], and *abduction-based* methods [18], [19], which identify the features that imply the decision and can thus provide (safety) guarantees. Attribution scores can also be used for feature importance ranking [20] to provide an overall understanding of the importance of different input attributes on the model decisions.

1) Gradient-based methods: Gradient-based methods aim to estimate feature attribution scores for model predictions. Among these, a prominent method is the integrated gradients (IG) [15], which measures the attribution score of each input feature to the model's prediction by integrating the gradients of the model's output with respect to the input features along the path from a baseline input to the actual input.

Definition 2. *Given a neural network f, an input x and a baseline input x', the integrated gradients for each input feature $i \in [1, \cdots, n]$ are defined as the weighted (by input feature difference) integral of the gradients over the straight line path between x and x':*

$$IG_i(x) = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha \quad (4)$$

Figure 3 from [5] presents an illustrative example of IG explanations, showing the explanation for each class of a correctly classified handwritten-digit "8" from the MNIST dataset. In this example, positive contributions are highlighted in red, while negative contributions are indicated by blue.

2) Perturbation-based methods: LIME [16] and its successor Anchors [17] are representatives of explainability methods that deploy a perturbation-based strategy to generate local explanations for model predictions. LIME assumes local linearity in a small area around an input instance and generates a set of synthetic data by perturbing the original input. Anchors [17] explains the model predictions by identifying a set of decision rules that "anchors" the prediction. Compared with LIME, Anchors generates more explicit decision rules and derives local explanations by consulting x's perturbation neighbourhood in different ways. In particular, Anchors evaluates the coverage fraction of the perturbed data samples sharing the same class as x, matching the decision rules.

3) Robust explanations: The explanation techniques mentioned above use different heuristics to derive local explanations, demonstrating effective generality beyond the given input but lacking robustness to adversarial perturbations. The robustness notion for explanation is important to ensure the stability of the explanation in the sense that the explanation is logically sufficient to imply the prediction. Intuitively, the computed explanation for a perturbed input should remain the same as the original input.

To this end, [18], [19] introduce a principled approach to derive explanations with formal guarantees by exploiting abduction reasoning. This ensures the robustness of the explanation by requiring its invariance w.r.t. any perturbation of the remaining features that are left out. The explanation method of [19] focuses on *optimal robust explanations (OREs)*, to provide both robustness guarantees and optimality w.r.t. a cost function. Optimality provides the flexibility to control the desired properties of an explanation. For instance, the cost function could be defined as the length of the explanation to derive minimal but sufficient explanations.

III. Certification for Neural Networks

In this section, we present an overview of recent advances for certification of neural networks, with a focus on formal verification. Given a neural network $f : \mathbb{R}^n \to \mathbb{R}^m$, we consider the formal verification problem [4], defined for a property specified as a pair $(\phi_{\mathsf{pre}}, \phi_{\mathsf{post}})$ of precondition and postcondition, by requiring that $\forall x \in \mathbb{R}^n. x \models \phi_{\mathsf{pre}} \implies f(x) \models \phi_{\mathsf{post}}$, that is, for all inputs satisfying the precondition the corresponding (optimal softmax) decision must satisfy the postcondition. Typically, $\phi_{\mathsf{pre}} \subseteq \mathbb{R}^n$ and $\phi_{\mathsf{post}} \subseteq \mathbb{R}^m$, but can be respectively induced from subsets of input features or sets of labels. Formal verification then aims to establish algorithmically whether this property holds, thus resulting in a *provable guarantee*. Otherwise, the property may be falsified, in which case a witness is provided, or inconclusive. Sometimes, we may wish to compute the proportion of inputs that satisfy the postcondition, known as *quantitative verification* [21].

Various formal verification methods have been proposed to provide provable guarantees for neural networks. We classify existing verification methods into forward and backward analysis, depending on whether they start from the input or output space.

- *Forward analysis:* Forward analysis methods start from the precondition $X = \{x \in \mathbb{R}^n \mid x \models \phi_{\mathsf{pre}}\}$ defined on the input space, and check whether the outputs (corresponding to the input region) satisfy the postconditions ϕ_{post}. For example, robustness verification approaches [22], [23], [24], [25] start from the perturbation neighbourhood of a given input, e.g., an l_∞ ball around an input point x, and compute bounds on the outputs to check whether the predicted labels over the adversarial region are preserved.
- *Backward analysis:* Backward analysis methods start from the postcondition $Y = \{y \in \mathbb{R}^m \mid y \models \phi_{\mathsf{post}}\}$ and aim to find the set of inputs that lead to such outputs. For example, preimage generation (inverse abstraction) approaches [26], [27], [28], [29] start from the output constraints, e.g., a polytope constraining the probability of the target label is greater than the other labels, and derive the input set that provably leads to this particular decision.

We remark that, similarly to formal verification for conventional software, certification for machine learning models is computationally expensive, and it is therefore recommended for use in safety- or security-critical settings. In less critical situations, diagnostic methods [30], which approximate model decisions to analyse their predictions, can be employed to investigate both model- and data-related issues.

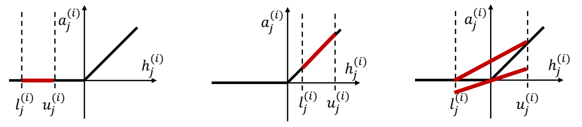


Fig. 4: Illustration of the convex relaxation for inactive (left), active (middle) and unstable (right) ReLU neurons.

A. Forward Analysis Methods

We categorize the forward analysis methods into two groups: *sound but incomplete* and *complete* methods. Soundness and completeness are essential properties of verification algorithms, which are defined as follows.

- *Soundness:* A verification algorithm is sound if the algorithm returns True and the verified property holds.
- *Completeness:* A verification algorithm is complete if (i) the algorithm never returns unknown; and (ii) if the algorithm returns False, the property is violated.

1) Incomplete methods: Incomplete verification methods leverage approximation techniques, such as search [1], [10], convex relaxation [31] and abstract interpretation [32], respectively to compute lower/upper bounds on MSR or the non-convex optimization problem. A safety property is verified when the reachable outputs satisfy the postcondition; otherwise, no conclusion can be drawn. At the same time, due to the relaxation introduced by the approximation techniques, incomplete methods have better scalability than complete ones.

a) Game-based search: Knowledge of the maximum safe radius (MSR) can serve as a guarantee on the maximum magnitude of the allowed adversarial perturbations. Unfortunately, MSR computation is intractable, and instead approximate algorithms have been developed for images in [10], and extended to videos in [33], that compute lower and upper bounds on MSR with provable guarantees, i.e., bounded error. The method relies on the network satisfying the Lipschitz condition and can be configured with a variety of feature extraction methods, for example SIFT. Given an over-approximation of the Lipschitz constant, the computation is reduced to a finite optimization over a discretisation of the input region X corresponding to the precondition ϕ_{pre}. The resulting finite optimization is solved in anytime fashion through a two-player game, where player 1 selects features and player 2 perturbs the image representation of the feature, and the objective is set to minimise the distance to an adversarial example. Under the assumptions, the game can be unfolded into a finite tree and Monte Carlo Tree Search (MCTS) used to approximate MSR upper bound, and Admissible A* MSR lower bound, respectively.

b) Bound propagation: A common technique for incomplete verification is applying convex relaxation to bound nonlinear constraints in neural networks. This way, the original non-convex optimization problem is transformed into a linear programming problem. With the relaxed linear constraints, the global lower and upper bounds can be computed more

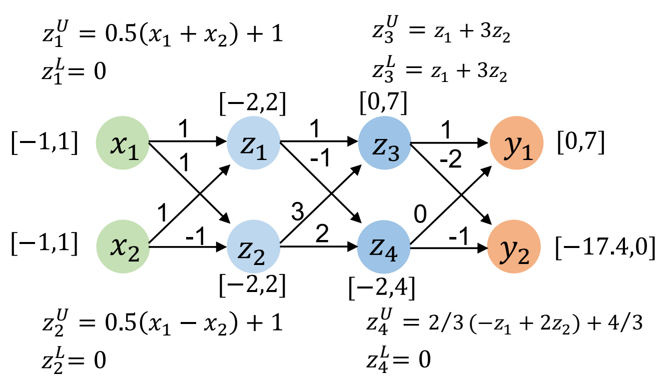


Fig. 5: Verification via bound propagation.

efficiently for the associated (relaxed) linear program. Representative methods that adopt efficient bound propagation include convex outer adversarial polytope [31], CROWN [34] and its generalization [35], [25]. Figure 4 illustrates convex relaxation using linear bounding functions to bind ReLU neurons. Note that relaxation is only introduced for unstable neurons, while the ReLU constraints for inactive and active ones are exact. For unstable neurons, the lower and upper bounding function for the j-th neuron of the i-th layer $a_j^{(i)}(x)$ (activated value) with regard to $h_j^{(i)}(x)$ (before activation) are:

$$\alpha_j^{(i)} h_j^{(i)}(x) \le a_j^{(i)}(x) \le -\frac{u_j^{(i)} l_j^{(i)}}{u_j^{(i)} - l_j^{(i)}} + \frac{u_j^{(i)}}{u_j^{(i)} - l_j^{(i)}} h_j^{(i)}(x) \quad (5)$$

where a flexible lower bound function with parameter $\alpha_j^{(i)}$ as in [35] is used, which leads to a valid lower bound for any parameter value within $[0, 1]$.

By propagating the linear (symbolic) upper and lower bounds layer by layer, we can obtain the linear bounding functions f^L, f^U for the entire neural network f, and it holds that $\forall x \in X.\, f^L(x) \le f(x) \le f^U(x)$. The non-convex verification problem is thus transformed into a linear program with the objective linear in the decision variables. The certified upper and lower bounds can be computed by taking the maximum, $\max_{x \in \mathbb{B}_p(x,\epsilon)} f^U(x)$, and the minimum, $\min_{x \in \mathbb{B}_p(x,\epsilon)} f^L(x)$, which have *closed-form* solutions for linear objectives (f^U, f^L) and convex norm constraints $\mathbb{B}_p(x, \epsilon)$.

Example 2. *Consider the neural network illustrated in Example 1. The verification problem we consider is given by the pre-condition* $\phi_{\text{pre}} = \{x \in \mathbb{R}^2 | x \in [-1,1] \times [-1,1]\}$ *and the post-condition* $\phi_{\text{post}} = \{y = f(x) \in \mathbb{R}^2 \mid y_1 \ge y_2\}$*, and we want to prove that* $\forall x.\, x \models \phi_{\text{pre}} \implies f(x) \models \phi_{\text{post}}$.

Figure 5 shows the overall bound propagation procedure for this verification problem, where the interval $[\cdot, \cdot]$ *represents the concrete value range computed for each neuron.* z_i^U, z_i^L *represent the linear upper and lower bounding functions for nonlinear neurons, which are computed according to Equation 5 based on the concrete value intervals. Starting from the input layer, we can first compute the concrete bounds (*$[-2, 2]$*) for* z_1 *and* z_2 *(before activation). The bounding functions (*z_1^L, z_1^U*), (*z_2^L, z_2^U*) are then computed according to Equation 5, where* $\alpha = 0$ *is taken as the lower bounding function coefficient. The linear bounding functions can directly propagate to the next layer via the linear matrix transformation. Then, by taking the minimum value of the lower bounding function and the maximum of the upper one, concrete value ranges (*$[0, 7]$ *and* $[-2, 4]$*) are computed for* z_3 *and* z_4*, based on which symbolic functions (*z_3^L, z_3^U*), (*z_4^L, z_4^U*) can be derived and further propagated to the output layer. In the end, we compute the global lower and upper bounds for* y_1 *and* y_2*, which are* $[0, 7]$ *and* $[-17.4, 0]$*, respectively. From the certified bounds on the output layer, it holds that* $\min(y_1) \ge \max(y_2)$ *for any input* $(x_1, x_2) \in [-1, 1] \times [-1, 1]$*. Therefore, the bound propagation method certifies that the neural network is robust in the input domain with respect to the ground-truth label* y_1.

c) Abstract interpretation: Abstract interpretation [32], [36] is a classic framework that can provide sound and computable finite approximations for infinite sets of behaviours. To provide sound analysis of neural networks, several works [22], [37], [38], [24] have exploited this technique to reason about safety properties. These methods leverage numerical abstract domains to overapproximate the inputs and compute an over-approximation of the outputs layer by layer. To this end, an abstract domain is selected to characterize the reachable output set for each layer as an abstract element. The choice of abstract domain is essential to balance the analysis precision and scalability. Commonly used abstract domains for neural network verification [39] include *Interval*, *Zonotope*, and *Polytope*, of which the general formulations are summarized in the following (increasing in precision):

$$\text{Interval:} \quad \{x \in \mathbb{R}^n | l_i \le x_i \le u_i\}$$

$$\text{Zonotope:} \{x \in \mathbb{R}^n | x_i = c_{i0} + \sum_{j=1}^{m} c_{ij} \cdot \epsilon_j, \epsilon_j \in [-1, 1]\}$$

$$\text{Polytope:} \{x \in \mathbb{R}^n | x_i = c_{i0} + \sum_{j=1}^{m} c_{ij} \cdot \epsilon_j, F(\epsilon_1, \cdots, \epsilon_m)\}$$

where ϵ_j ($j = 1, \cdots, m$) denote m generator variables. The generator variables are bounded within the interval $[-1, 1]$ for zonotopes and constrained by F for polytopes, where F takes in the form of a convex polytope $\mathbf{cx} \le \mathbf{d}$.

With the abstract domain capturing the reachable outputs of each layer, abstract transformers are defined to compute the effect of different layers on propagating the abstract element. Affine transformers are usually supported by the underlying abstract domain, such as Zonotope and Polytope, to abstract the linear functions. For nonlinear functions, case splitting and unifying is proposed in [22] by defining the *meet* and *join* operators to propagate zonotope abstraction through piecewise-linear layers. Convex approximations are adopted in [24] for abstract transformers of nonlinear functions where the approximation can be captured with the proposed polyhedra abstraction. At the end of the analysis, the abstract element of the output layer is an over-approximation of all possible concrete outputs corresponding to the input set. Then we can directly verify the over-approximation of the outputs against the postcondition ϕ_{post}, i.e., check whether the over-approximation is fully contained within ϕ_{post}. One drawback of this method is that the over-approximation may be quite loose.

2) Complete methods: Early complete verification approaches for neural networks [40], [3] encode the neural network into a set of constraints exactly and then check the satisfaction of the property with constraint solvers, e.g, SMT (Satisfiability Modulo Theory) or MILP (Mixed Integer Linear Programming) solvers. Since such constraint-solving methods encode the neural network in an exact way, they are able to ensure both soundness and completeness in providing certification guarantees. One limitation is that these methods suffer from exponential complexity in the worst case. To address the computational intractability, Branch and Bound techniques are adopted and customized for neural network verification, where efficient incomplete methods can be exploited to speed up the bound computation.

a) SMT solver: Reluplex [3] is proposed as a customized SMT solver for neural network verification. The core idea is to extend the simplex algorithm, a standard algorithm to solve linear programming problems, with additional predicates to encode (piecewise linear) ReLU functions and transition rules (*Pivot* and *Update*) to handle ReLU violations. The extended Reluplex algorithm allows variables that encode ReLU nodes to temporarily violate the ReLU constraints. Then, as the iteration proceeds, the solver picks variables that violate a ReLU constraint and modifies the assignment to fix the violation using *Pivot* and *Update* rules. When the attempts to fix a ReLU constraint using *Update* rules exceed a threshold, a ReLU splitting mechanism is applied to derive two sub-problems. Reluplex is then invoked recursively on these two sub-problems. Compared with the eager splitting on all ReLU neurons, Reluplex proposes a splitting-on-demand strategy to reduce unnecessary splitting and limit splits to ReLU constraints that are more likely to cause violation problems. Due to the exact encoding nature, Reluplex suffers from exponential complexity in the worst case and thus cannot scale to large neural networks.

b) MILP: MILP-based verification methods [41], [42], [43] encode a neural network with piecewise-linear functions as a set of mixed integer linear constraints. To encode the nonlinearities, they introduce an indicator decision variable δ to characterize the two statuses of unstable ReLU neurons. An unstable ReLU neuron $z = \max(\hat{z}, 0)$ with concrete bounds (l, u) can be encoded exactly using the following constraints:

$$\begin{aligned}
&z \geq 0, \quad z \leq u \cdot \delta, \\
&z \geq \hat{z}, \quad z \leq \hat{z} - l \cdot (1 - \delta), \\
&\delta \in \{0, 1\}
\end{aligned}$$

Note that the MILP constraints require the pre-computation of finite bounds for the nonlinear neurons, i.e., (l, u). It is known that the tightness of lower and upper bounds in the indicator constraints is crucial to the resolution of the MILP problem [44], [42], and consequently, the verification efficiency. MIPVerify [43] thus proposes a progressive bound tightening approach to improve upon existing MILP-based verifiers. The algorithm starts with coarse bounds computed using efficient bound computation procedures such as *Interval Arithmetic*. Bound refinement is performed only when the MILP problem can be further tightened. In such a case, more precise but less efficient bound computation procedures, e.g., *Linear Programming* (LP), are adopted to derive tighter bounds. This progressive bounding procedure can also be extended to other bound computation methods, such as dual optimization, to achieve a trade-off between tightness and computational complexity.

Example 3. *In this example, we encode the neural network, shown in Example 1, into the exact MILP formulation. The verification problem is the same as shown in Example 2, i.e., to determine whether $\phi_{\text{post}} = \{y \in \mathbb{R}^2 \mid y_1 \geq y_2\}$ holds for all inputs in the input domain $[-1, 1] \times [-1, 1]$. We encode the output property by specifying its negation, i.e., $\phi'_{\text{post}} = \neg\phi_{\text{post}}$. If there exists an instance where ϕ'_{post} does hold, then a witness to ϕ'_{post} is the counter-example for ϕ_{post}. If ϕ'_{post} is unsatisfiable, then the property ϕ_{post} is proved.*

Assume we have computed the concrete value of lower and upper bounds of z_i employing efficient bound propagation techniques. Then the neural network and the verification problem can be formulated as follows:

$$x_1 \geq -1,\, x_1 \leq 1,\, x_2 \geq -1, x_2 \leq 1\, (\phi_{\text{pre}}) \tag{6}$$
$$\hat{z}_1 = x_1 + x_2, \quad \hat{z}_2 = x_1 - x_2, \quad \delta_1, \delta_2 \in \{0, 1\} \tag{7}$$
$$z_1 \geq 0,\, z_1 \leq 2\delta_1,\, z_1 \geq \hat{z}_1,\, z_1 \leq \hat{z}_1 + 2(1 - \delta_1), \tag{8}$$
$$z_2 \geq 0,\, z_2 \leq 2\delta_2,\, z_2 \geq \hat{z}_2,\, z_2 \leq \hat{z}_2 + 2(1 - \delta_2), \tag{9}$$
$$\hat{z}_3 = z_1 + 3z_2, \quad \hat{z}_4 = -z_1 + 2z_2, \quad \delta_4 \in \{0, 1\} \tag{10}$$
$$z_3 = \hat{z}_3,\, \textit{(stable neuron)} \tag{11}$$
$$z_4 \geq 0,\, z_4 \leq 4\delta_4,\, z_4 \geq \hat{z}_4,\, z_4 \leq \hat{z}_4 + 2(1 - \delta_4), \tag{12}$$
$$y_1 = z_3, \qquad y_2 = -2z_3 - z_4 \tag{13}$$
$$y_1 < y_2\, (\neg\phi_{\text{post}}) \tag{14}$$

The lower and upper bounds l_i and u_i ($i \in \{1, 2, 3, 4\}$) are derived as shown in Example 2. The binary variables δ_i ($i \in \{1, 2, 4\}$) are introduced to indicate the status of unstable ReLUs and it holds that $\delta_i = 0 \Leftrightarrow z_i = 0$ and $\delta_i = 1 \Leftrightarrow z_i = \hat{z}_i$. Checking the feasibility of the above model using MILP solvers (e.g., Gurobi) will return infeasible, thus proving the original property.

c) Branch and Bound: To improve the scalability of verification algorithms to larger neural networks, a branch and bound framework (BaB) [45] has been proposed. The BaB framework mainly consists of two components: a branching method that splits the original verification problem into multiple subproblems and a bounding method to compute the upper and lower bounds of the subproblems. This modularized design provides a unifying formulation paradigm for different verifiers, with the main difference lying in the splitting function and the bounding method. For example, the verifier *ReluVal* [46] performs splitting on the input domain according to sensitivity analysis, e.g., input-output gradient information, and computes bounds using symbolic interval propagation. The aforementioned SMT-based verifier *Reluplex* [3] performs splitting on ReLU neurons guided by the violation frequency of the ReLU constraints and computes the bounds on the relaxed problems by dropping some constraints on the nonlinearities (which yields an over-approximation of the constraint optimization problems).

To further improve neural network verification, the BaB method introduces two new branching strategies: BaBSB for branching on input domains and BaBSR for branching on ReLU neurons. Both branching methods adopt a similar heuristic to decide which dimension or ReLU neuron to split on. BaBSB computes a rough estimate of the improvement on the bounds obtained with regard to every input dimension, where the estimation makes the split decision be set more efficiently. On the other hand, BaBSR estimates the bound improvement with regard to each unfixed ReLU neuron by computing the ReLU scores. The bounding methods resort to LP solvers to tighten the intermediate bounds on the subdomains or use more computationally efficient methods such as Interval Arithmetic.

B. Backward Analysis Methods

Backward analysis methods for neural networks, also known as preimage generation or inverse abstraction, aim at computing the input set that will lead the neural network to a target set, e.g., a safe or unsafe region. They complement the forward analysis methods, which may result in over-approximated bounds worsening as the computation progresses through the layers of the network. In the following, we categorize the representative approaches broadly into two groups: *exact* and *approximate* methods.

1) Exact methods: Exact backward analysis methods reason about the preimage of a target output set by encoding the neural network behaviours in an exact manner. These methods are able to compute the exact symbolic representation of the preimage for different output properties. One limitation suffered by these methods is that they can only process neural networks with piecewise-linear activation functions (e.g., $ReLU$), as they aim at an exhaustive decomposition of the non-convex function (the neural network) into a set of linear functions. The preimage (input set) for a target output set with regard to a neural network f is characterized as a union of polytopes, where the mapping functions are completely linear on each subregion.

a) Exact preimage: The exact preimage generation method [26] complements the forward analysis methods to reason about the inputs that lead to target outputs. The algorithm computes the exact preimage by relying on two elementary properties: (1) preimage of the composite functions is the reversed composition of preimages for each layer, i.e., $(f^{(L)} \circ \cdots \circ f^{(1)})^{-1} = (f^{(1)})^{-1} \circ \cdots \circ (f^{(L)})^{-1}$, and (2) preimage of a union set can be built up from the preimages

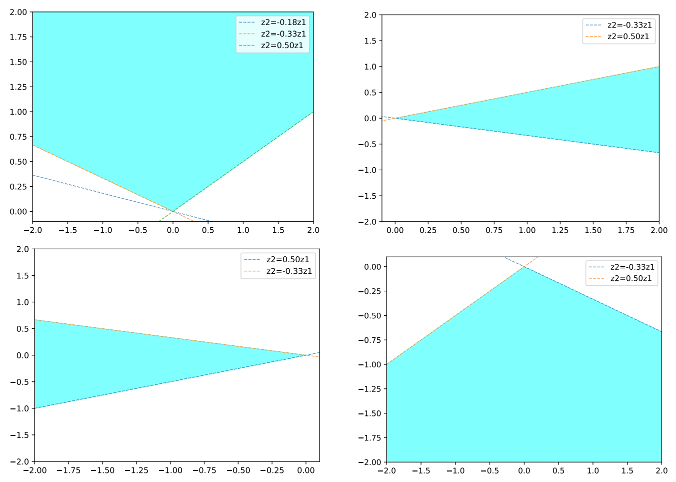


Fig. 6: Preimage polytopes by exact method.

of each subset in the union, i.e., $f^{-1}(\cup_j S_j) = \cup_j f^{-1}(s_j)$. This method assumes that the output set, e.g., a safe region, can be formulated as a polytope (intersection of half-planes) $\{y \in \mathbb{R}^m | Ay - b \leq 0\}$. It then propagates the polytope backwards through the layers.

For linear layers, the preimage is computed by applying the linear operations corresponding to the layer. Suppose we have a linear mapping in the form of $y = Wz + a$, then the preimage of the output polytope under this linear operation can be formulated as $\{z \in \mathbb{R}^{n_{L-1}} | AWz + (Aa - b) \leq 0\}$. For nonlinear layers, the algorithm restricts the backward propagation to a subset where the activation pattern of the ReLU neurons is fixed. Let $s(z)$ denote the activation status vector of the nonlinear neurons where $s(z)_j = 1$ if $z_j \geq 0$ and $s(z)_j = 0$ otherwise. A diagonal matrix $diag(s(z))$ is introduced to restrict to a fixed activation pattern, on which only linear computation is required to compute the preimage subset. The exact preimage can then be computed by taking the union of each partition (preimage property (2)).

$$\begin{aligned} &ReLU^{-1}(\{y \in \mathbb{R}^m | Ay - b \leq 0\}) \\ &= \bigcup_{s \in \{0,1\}^{n_i}} \{z \in \mathbb{R}^{n_i} | A diag(s) z - b \leq 0, -diag(s) z \leq 0, \\ &\qquad diag(1-s) z \leq 0\} \end{aligned}$$

Example 4. *In this example, we consider the same verification problem as in Example 2 and 3, but from the backward perspective. Preimage analysis aims to investigate whether the input region* $[-1, 1] \times [-1, 1]$, *which is expected to result in decision* y_1, *fails the safety check. We first formulate the target output region as a polytope. Since we only have two labels, the output constraint is, therefore, a single half-plane encoded as* $\{y \in \mathbb{R}^2 | y_1 - y_2 \geq 0\}$. *We then proceed to compute the preimage of the target polytope under the linear mapping (from the* 2^{nd} *hidden layer to the output layer), of* *which the result is* $\{(z_3, z_4) \in \mathbb{R}^2 \mid 3z_3 + z_4 \geq 0\}$. *Next, preimage computation for ReLU starts with partitioning the neuron vector space* $\mathbb{R}^2$ *into* 2^2 *sets where, for each subset, the status of nonlinear neurons is fixed and preimage computation proceeds similarly to the linear mapping. The partition leads to four result polytopes.*

Figure 6 shows the result of four preimage polytopes derived in the two-dimensional space (z_1, z_2). *As an example, the preimage polytope derived corresponding to the partition where both neurons are active is (upper left of Figure 6):*

$$\{z^{(1)} \in \mathbb{R}^2 \ : A^{(1)} z^{(1)} \geq 0\} \quad \textit{where}$$
$$A^{(1)} = \begin{bmatrix} 2 & 11 \\ 1 & 3 \\ -1 & 2 \end{bmatrix}, \quad z^{(1)} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

The other three polytopes are derived in the same way. The four preimage polytopes are then partitioned further into 16 polytopes to characterize the exact preimage of the input layer. Note that the combination of the four polytopes actually covers the hidden vector space $[-2, 2] \times [-2, 2]$, *and the resulting preimage polytopes on the input layer cover the region* $[-1, 1] \times [-1, 1]$, *which certifies that the correct decision is taken for the entire region under investigation.*

b) SyReNN: SyReNN is proposed in [47] to compute the symbolic representation of a neural network so as to understand and analyze its behaviours. It targets *low-dimensional* input subspaces and computes their exact symbolic partitioning, on which the mapping function is completely linear. This methodological design is also referred to as neural network decomposition. We classify it as a backward analysis method, as this method provides a symbolic representation in the input space. SyReNN focuses on neural networks with piecewise-linear activation functions. This restriction enables a precise characterisation of the input space X as a finite set of polytopes $\{X_1, \cdots, X_n\}$. Within each input polytope X_i, the neural network is equivalent to a linear function. By means of such a symbolic representation, safety verification is reduced to checking whether the vertices of every bounded convex polytope X_i satisfy the output property.

To compute the symbolic decomposition on the input domain, this algorithm starts with the trivial partition X and derives the linear partitions layer by layer. Given the partition hyperplanes of the nonlinear layer i, e.g., $z_1 = 0, z_2 = 0, \cdots, z_{n_i} = 0$ with n_i ReLUs, and the symbolic representation $\hat{f}_{i-1}$ (a set of polytopes) computed until layer $i-1$, $\hat{f}_i$ is computed by recursively partitioning the current polytopes based on the newly-added hyperplanes. For example, given a polytope Z_{i-1}, if an orthant boundary (e.g., hyperplane $z_i = 0$) is hit when traversing the boundary of Z_{i-1}, then Z_{i-1} is further partition into $Z_{i-1,1}$ and $Z_{i-1,2}$ which lie on the opposite sides of the hyperplane. This procedure terminates until all resulting polytopes lie within a completely linear region of the neural network f.

2) Approximate methods: Exact methods for preimage analysis suffer from exponential complexity in the worst case. Similarly to the development of incomplete verifiers, preimage approximation techniques begin to emerge by leveraging different approximation (relaxation) techniques. They compute a symbolic approximation of the preimages to bypass the intractability of computing exact preimage representations. Computational efficiency and scalability can be greatly improved with the sacrifice of precision.

a) Symbolic interpolation: Symbolic interpolation [48] has been used for program verification and SMT solving. To compute provable preimage approximations, [27] leverages interpolants, especially those with simple structures, and computes preimages from the output space through hidden layers to the input space. The generated approximations can then be applied to reason about the properties of the neural network itself. For example, in the case that a desired property (a target output set) Y should be satisfied when starting from a certain input set X, an under-approximation $\underline{X}$ of the preimage for Y can be computed. Then the property can be verified by checking whether $\underline{X} \rightarrow X$ holds.

[48] proposes an algorithm to compute the preimage approximation by iterating backwards through the layers. It encodes the neural network as constraints in the theory of *quantifier-free linear rational arithmetic* (QFLRA) and requires the output set to be encoded as a Boolean combination of atoms in the form of half-spaces. Suppose we now focus on deriving preimage over-approximations of the target output set Y. The algorithm starts by computing the (overapproximated) set of inputs to the last layer, denoted as $p_L^{f,Y}$, which leads to the output set Y, i.e., $f^{(L)}(p_L^{f,Y}) \models Y$. The algorithm then iteratively computes preimages of the other layers that satisfy $p_i^{f,Y} = \{z \mid f^{(i)}(z) \models p_{i+1}^{f,Y}\}$. This procedure leverages sampling techniques to construct a set of points mapped to the complement of Y, which are used to tighten the over-approximations. The algorithm relies on Craig's Interpolation theorem to guarantee the existence of an (over- and under-) approximation. It also leverages the bound propagation framework to compute a bounded domain on each layer, which speeds up the interpolation condition checking.

b) Inverse bounding: [28] points out two important applications based on preimage analysis: safety verification for dynamical systems and out-of-distribution input detection. Motivated by these use cases, an inverse bound propagation method is proposed to compute the over-approximation of the preimage. Bound propagation has been widely employed to build efficient verifiers in the forward direction (certified output bound computation). Compared with forward analysis, it is challenging to adopt bound propagation methods directly to compute tight intermediate bounds, and thus difficult to compute tight over-approximations. This is because, for the inverse problem, the constraints on the input are quite loose and even unbounded in some control applications. Simply applying the bound propagation procedure will not lead to useful intermediate bounds, which further impacts the tightness of the symbolic relaxation on nonlinear neurons.

Given this, an inverse propagation algorithm is proposed in [28] to compute a convex over-approximation of the preimage

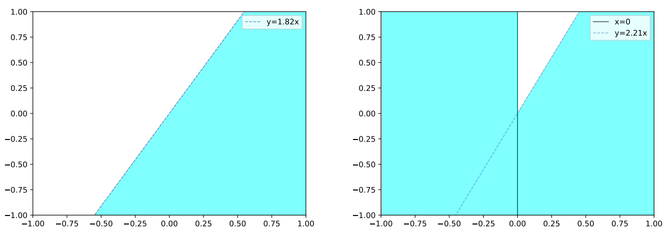


Fig. 7: Preimage approximation.

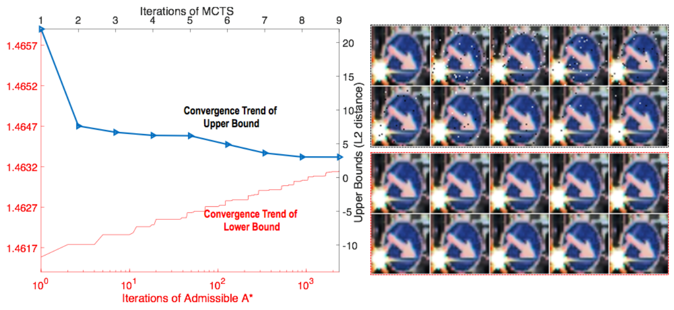


Fig. 8: Convergence of maximum safe radius computed using the game-based method for a traffic sign image from the GTSRB dataset originally classified as "keep right". Left: The convergence trends of the upper bound obtained with Monte Carlo Tree Search and the lower bound with Admissible A*. Right: unsafe images (top two rows) and certified safe images (bottom two rows). Figure taken from [10].

represented by a set of cutting planes. It first transforms the preimage over-approximation problem to a constrained optimization problem over the preimage and further relaxes it to Lagrangian dual optimization. To tighten the preimage and intermediate bounds, they introduce a dual variable with respect to the output constraints and tighten these bounds iteratively, leveraging standard gradient ascent algorithm.

c) Preimage approximation: Motivated by the practical needs of global robustness analysis [49], [11], [21] and quantitative verification [50], [51], an anytime algorithm is proposed in [29] to compute provable preimage approximation. The generated preimage is further applied to verify quantitative properties of neural networks, which is defined by the relative proportion of the approximated preimage volume against the input domain under analysis, formally defined as follows.

Definition 3. *Given a neural network $f : \mathbb{R}^n \to \mathbb{R}^m$, a measurable input set with non-zero measure (volume) $X \subseteq \mathbb{R}^n$, a measurable output set $Y \subseteq \mathbb{R}^m$, and a rational proportion $p \in [0, 1]$, the neural network satisfies the quantitative property (X, Y, p) if $\frac{\text{vol}(f_X^{-1}(Y))}{\text{vol}(X)} \geq p$.*

This approach targets safety properties that can be represented as polytopes and characterizes preimage under-approximation using a disjoint union of polytopes. To avoid the intractability of the exact preimage generation method, convex relaxation is used to derive sound under-approximations. However, one challenge is that the generated preimage under-approximation can be quite conservative when reasoning about properties in large input spaces with relaxation errors accumulated through each layer. To refine the preimage abstraction, a global branching method is introduced to derive tighter approximations on the input subregions. This procedure proposes a (sub-)domain search strategy prioritizing partitioning on most uncovered subregions and a greedy splitting rule leveraging GPU parallelization to achieve better per-iteration improvement. To further reduce the relaxation errors, this method formulates the approximation problem as an optimization problem on the preimage polytope volume. Then it proposes a differentiable relaxation to optimize bounding parameters using projected gradient descent.

Example 5. *In this example, we demonstrate how to construct a provable preimage (under-)approximation for the target output region, and apply it to quantitative analysis of the verification problem shown in previous examples. Consider the quantitative property with input set $\phi_{\text{pre}} = \{x \in \mathbb{R}^2 \mid x \in [-1, 1]^2\}$, output set $\phi_{\text{post}} = \{y \in \mathbb{R}^2 \mid y_1 - y_2 \geq 0\}$, and quantitative proportion $p = 0.9$. We apply the preimage approximation algorithm to verify this property. Figure 7 presents the computed preimage before (left) and after one-iteration refinement (right). Note that the partition is performed w.r.t. input x_1, which results in two polytopes for the subregions. We compute the exact volume ratio of the refined under-approximation against the input set. The quantitative proportion reached with the refinement is 94.3%, which verifies the quantitative property.*

IV. Application Examples

In this section we provide a selection of experimental results and lessons learnt from applying formal verification and certification approaches described in the previous section to neural network models drawn from a range of classification problems. These include image and video recognition, automated decisions in finance and text classification. In addition to adversarial robustness of the models, we demonstrate certification of individual fairness of automated decisions and discuss robust explanations.

A. MSR-based Certification for Images and Videos

The game-based method [10] has been applied to analyse and certify the robustness of image classification models to adversarial perturbations with respect to the maximal safe radius, working with a range of feature extraction methods and distance metrics. Figure 8 shows a typical outcome of such analysis, with converging lower and upper MSR bounds for an image of a traffic sign for l_2 distance and features extracted from the latent representation computed by a convolutional neural network (CNN) model. It can be seen that the image is certified safe for adversarial perturbations of up to 1.463 in l_2 distance, which is some distance away from the best upper bound at approx. 3, but can be improved with more iterations since the method is anytime.

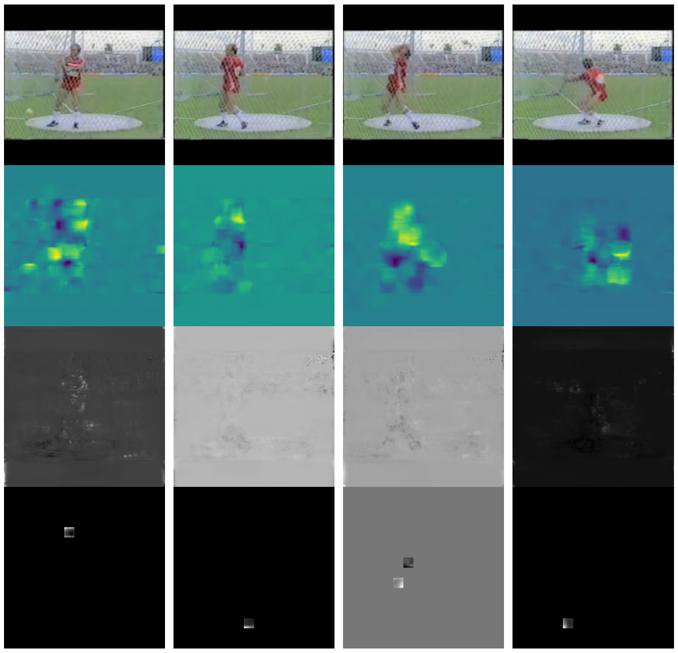


Fig. 9: Shown in top row are sampled frames of a HammerThrow video and the corresponding optical flows are in the 2nd row. Unsafe perturbations of flows are in 3rd row and safe in 4th. Figure taken from [33].

An extension of the game-based method was developed in [33] to provide MSR-based certification for videos, and specifically for neural network models consisting of a CNN to perform feature extraction and a recurrent neural network (RNN) to process video frames. Adversarial perturbations were defined with respect to optical flow, and the algorithmic techniques involve tensor-based computation. Examples of safe and unsafe perturbations are shown in Figure 9, and convergence trends for lower and upper bounds similar to those in Figure 8 can be observed.

B. *Robustness of Language Models*

As an example of application of convex relaxation tools (variants of CROWN [34]), we mention the study of [52], which aims to assess the robustness of Natural Language Processing tasks (sentiment analysis and text classification) to word substitution. It was reported that standard fully connected (FC) and CNN models are very brittle to such perturbations, which may make their certification unworkable. [53] critiqued the appropriateness of the classical concept of adversarial robustness defined in terms of word substitution in the context of NLP models. It was observed in an empirical study that models trained to be robust in the classical sense, for example, trained using interval bound propagation (IBP), lack robustness to syntax/semantic manipulations. It was then argued in [53] that a *semantic* notion of robustness that better captures linguistic phenomena such as shallow negation and sarcasm is needed for language models, where a framework based on templates was developed for evaluation of semantic robustness.

C. *Robust Explanations for Language Models*

Explainability of language models was studied in [19], with a focus on robust optimal explanations that imply the model prediction. Figure 10 shows examples of high-quality robust optimal explanations (using the minimum length of explanation as the cost function). In contrast, heuristic explanations such as integrated gradients or Anchors my lack of robustness, but it is possible to repair non-robust Anchors explanations by minimally extending them, see Figure 11.

D. *Fairness Certification Using MILP*

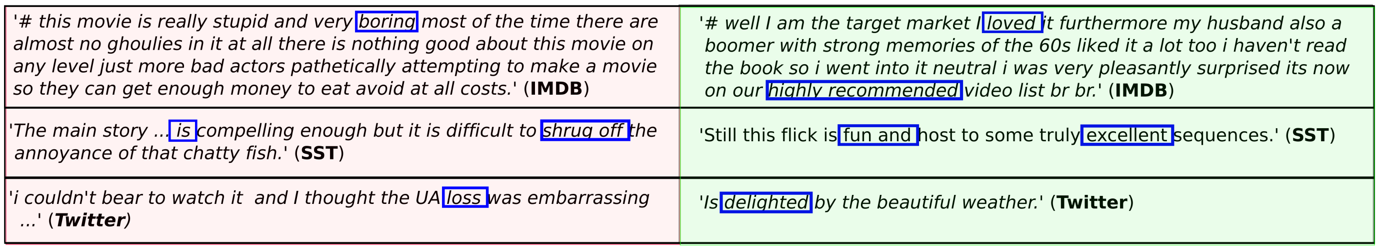
[54] developed methods for certification of individual fairness of automated decisions, defined, given a neural network and a similarity metric learnt from data, as requiring that the output difference between any pair of ϵ-similar individuals is bounded by a maximum decision tolerance $\delta \geq 0$. Working with a range of similarity metrics, including Mahalanobis distance, a MILP-based method was developed not only to compute certified bounds on individual fairness, but also to train certifiably fair models. The computed certified bounds δ_* are plotted in Figure 12 for the Adult and the Crime benchmarks. Each heat map depicts the variation of δ_* as a function of ϵ and the NN architecture. It can be observed that increasing ϵ correlates with an increase in the values for δ_*, as higher values of ϵ allow for greater feature changes.

V. Future Challenges

Formal verification and certification of neural network models has made steady progress in recent years, with several tools released to the community and an established tool competition [4]. Nevertheless, considerable scientific and methodological progress is needed before these tools are adopted by developers. Below we outline a number of research challenges.

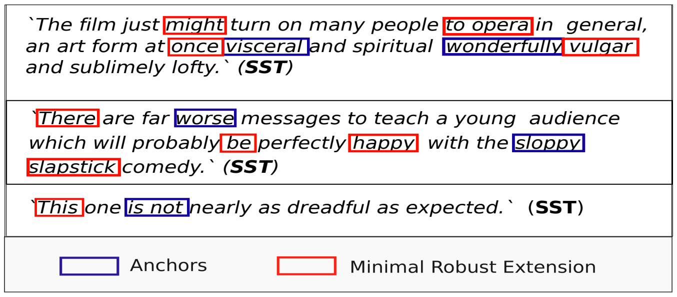
a) Beyond ℓ_p-norm robustness: The vast majority of robustness evaluation frameworks consider bounded ℓ_p-norm perturbations. While these suffice as proxies for minor visual image perturbations, real-world tasks rely on similarity measures, for example cosine similarity for word embeddings or Mahalanobis distance for images. It is desirable to define measures and certification algorithms for semantic robustness, which considers such similarity measures as first class citizens, and works with perturbations that reflect visual or geometric aspects characteristic of the application, such as object movement or lighting conditions. More generally, robustness evaluation frameworks for more complex properties induced by the use cases will be needed.

b) Beyond supervised robustness: Existing robustness formulations focus on the supervised learning setting. However, collecting and labelling large datasets that are necessary to ensure the high robustness performance needed in safety-critical applications is costly and may not be feasible for use cases such as autonomous driving. Instead, it is desirable to formulate robustness measures and evaluation frameworks directly in some appropriate semi-supervised, or even unsupervised, setting, where the definition of robustness needs to focus on the quality of the learned representations rather than



'*# this movie is really stupid and very boring most of the time there are almost no ghoulies in it at all there is nothing good about this movie on any level just more bad actors pathetically attempting to make a movie so they can get enough money to eat avoid at all costs.*' (**IMDB**)	'*# well I am the target market I loved it furthermore my husband also a boomer with strong memories of the 60s liked it a lot too i haven't read the book so i went into it neutral i was very pleasantly surprised its now on our highly recommended video list br br.*' (**IMDB**)
'*The main story ... is compelling enough but it is difficult to shrug off the annoyance of that chatty fish.*' (**SST**)	'Still this flick is fun and host to some truly excellent sequences.' (**SST**)
'*i couldn't bear to watch it and I thought the UA loss was embarrassing ...*' (***Twitter***)	'*Is delighted by the beautiful weather.*' (**Twitter**)

Fig. 10: Optimal robust explanations (highlighted in blue) for IMDB, SST and Twitter datasets (all the texts are correctly classified). Figure taken from [19].



`*The film just might turn on many people to opera in general, an art form at once visceral and spiritual wonderfully vulgar and sublimely lofty.*` (***SST***)

`*There are far worse messages to teach a young audience which will probably be perfectly happy with the sloppy slapstick comedy.*` (***SST***)

`*This one is not nearly as dreadful as expected.*` (**SST**)

Anchors
Minimal Robust Extension

Fig. 11: Examples of Anchors explanations (in blue) along with the minimal extension required to make them robust (in red). Figure taken from [19].

classification (prediction) because of the lack of labels. This may involve working with similarity measures such as Mahalanobis distance and will be challenging both theoretically and computationally to achieve provable robustness guarantees.

c) Scalability in network width and depth: Despite much progress, the scalability of robustness certification and evaluation frameworks remains limited to low-dimensional models. In order to apply certification to realistic use cases (such as object detection) will necessitate significant improvements with respect to input dimensionality and network depth, as well as the types of activation functions that can be handled.

d) Efficiency and precision trade-off: Robustness certifications and evaluation involves a variety of methods, including exact, approximate and statistical. While exact methods offer completeness, trading off exact precision for approximate bounding results in more efficient any time methods, and completeness can be recovered by combining fast approximate methods such as convex relaxation with branch-and-bound computation. Statistical methods provide estimates of robustness that may be unsound but fast and, in many cases, sufficient for the application being considered.

e) Compositionality and modularity of AI systems: Certification tools that have been developed to date are monolithic, which matches the monolithic structure of the vast majority of neural network models. Yet, similarly to safety-critical systems, it is anticipated that better structuring of models and tools is likely to improve their reliability and maintainability. Therefore, modularity, compositionality and, in particular, assume-guarantee compositional frameworks, are desirable future directions.

f) Calibrating uncertainty: It is recognised that deterministic neural networks can be overconfident in their decisions, and instead a variant known as Bayesian neural networks (BNNs), which admits a distribution over the weights and provides outputs in the form of the posterior distribution, is preferred, as it allows for a principled means to return an uncertainty measure alongside the network output. BNN certification methodologies are much more complex than for deterministic NNs, and still in early stages of development, including uncertainty quantification [55], computing lower bounds on safety probability [56] and certifiable adversarial robustness [57]. Unfortunately, the methods do not scale beyond small networks and standard Bayesian inference tends to underestimate uncertainty.

g) Robust learning: A drawback of certification as presented in this paper is that it pertains to trained models, and if the model fails certification, it is not clear how it can be repaired, and expensive retraining may be needed. A natural question then arises as to whether one can learn a model that is guaranteed to be robust. Building on the positive and negative theoretical results in the case of robust learning against evasion attacks [58], [59], [60], [61], it would be interesting to generalise these results to neural network models and development of implementable frameworks that can provide provable guarantees on robustness.

VI. Conclusion

We have provided a brief overview of formal verification approaches that can be employed to certify neural network models at test time, to train certifiably robust or fair models, and to provide meaningful explanations for network predictions. The methods can be categorised into forward and backward analysis, and involve techniques such as search, bound propagation, constraint solving and abstract interpretation. Both forward and backward analysis have the potential to support more complex verification properties, which have been little explored to date. Empirical results obtained on a range of standard benchmarks show that neural network models are often brittle to adversarial perturbations, but verification approaches can be used to strengthen their robustness and compute certification guarantees, thus improving trustworthiness of AI decisions.

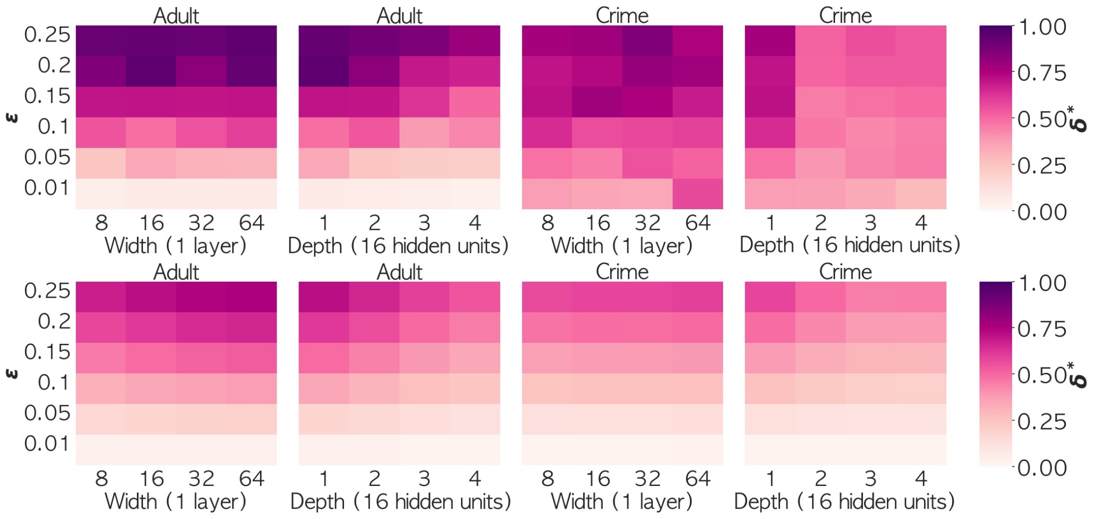


Fig. 12: Certified bounds on individual fairness (δ_*) for different architecture parameters (widths and depths) and maximum similarity (ϵ) for the Adult and the Crime datasets. Similarity metrics used are Mahalanobis (top row) and weighted ℓ_∞ metric (bottom row). Figure taken from [54].

Acknowledgments

This project received funding from the ERC under the European Union's Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No. 834115) and ELSA: European Lighthouse on Secure and Safe AI project (grant agreement No. 101070617 under UK guarantee).

References

[1] M. Wicker, X. Huang, and M. Kwiatkowska, "Feature-guided black-box safety testing of deep neural networks," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2018, pp. 408–426.

[2] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *29th International Conference on Computer Aided Verification*, ser. Lecture Notes in Computer Science, vol. 10426. Springer, 2017. doi: 10.1007/978-3-319-63387-9_1 pp. 3–29.

[3] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *29th International Conference on Computer Aided Verification*, ser. Lecture Notes in Computer Science, vol. 10426. Springer, 2017. doi: 10.1007/978-3-319-63387-9_5 pp. 97–117.

[4] C. Brix, M. N. Müller, S. Bak, T. T. Johnson, and C. Liu, "First three years of the international verification of neural networks competition (VNN-COMP)," *CoRR*, vol. abs/2301.05815, 2023. doi: 10.48550/arXiv.2301.05815

[5] R. Falconmore, "On the role of explainability and uncertainty in ensuring safety of AI applications," Ph.D. dissertation, University of Oxford, UK, 2022.

[6] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *European Conference on Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, vol. 8190. Springer, 2013. doi: 10.1007/978-3-642-40994-3_25 pp. 387–402.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[8] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018. doi: 10.1145/3243734.3264418 pp. 2154–2156.

[9] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *29th International Conference on Machine Learning*. icml.cc / Omnipress, 2012.

[10] M. Wu, M. Wicker, W. Ruan, X. Huang, and M. Kwiatkowska, "A game-based approximate verification of deep neural networks with provable guarantees," *Theoretical Computer Science*, vol. 807, pp. 298–329, 2020. doi: 10.1016/j.tcs.2019.05.046

[11] K. Leino, Z. Wang, and M. Fredrikson, "Globally-robust neural networks," in *38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 6212–6222.

[12] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Commun. ACM*, vol. 63, no. 1, pp. 68–77, 2020. doi: 10.1145/3359786

[13] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[14] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[15] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 3319–3328.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. doi: 10.1145/2939672.2939778 pp. 1135–1144.

[17] ——, "Anchors: High-precision model-agnostic explanations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[18] A. Ignatiev, N. Narodytska, and J. Marques-Silva, "Abduction-based explanations for machine learning models," in *Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33011511 pp. 1511–1519.

[19] E. L. Malfa, R. Michelmore, A. M. Zbrzezny, N. Paoletti, and M. Kwiatkowska, "On guaranteed optimal robust explanations for NLP models," in *Thirtieth International Joint Conference on Artificial Intelligence*. ijcai.org, 2021. doi: 10.24963/ijcai.2021/366 pp. 2658–2665.

[20] A. Janusz, D. Slezak, S. Stawicki, and K. Stencel, "A practical study of methods for deriving insightful attribute importance rankings using decision bireducts," *Inf. Sci.*, vol. 645, p. 119354, 2023. doi: 10.1016/j.ins.2023.119354

[21] B. Wang, S. Webb, and T. Rainforth, "Statistically robust neural network classification," in *Uncertainty in Artificial Intelligence*. PMLR, 2021, pp. 1735–1745.

[22] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. T. Vechev, "AI2: safety and robustness certification of neural net-

works with abstract interpretation," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2018. doi: 10.1109/SP.2018.00058 pp. 3–18.

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

[24] G. Singh, T. Gehr, M. Püschel, and M. T. Vechev, "An abstract domain for certifying neural networks," *Proc. ACM Program. Lang.*, vol. 3, no. POPL, pp. 41:1–41:30, 2019. doi: 10.1145/3290354

[25] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C. Hsieh, and J. Z. Kolter, "Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification," in *Annual Conference on Neural Information Processing Systems*, 2021, pp. 29 909–29 921.

[26] K. Matoba and F. Fleuret, "Exact preimages of neural network aircraft collision avoidance systems," in *Proceedings of the Machine Learning for Engineering Modeling, Simulation, and Design Workshop at Neural Information Processing Systems*, 2020, pp. 1–9.

[27] S. Dathathri, S. Gao, and R. M. Murray, "Inverse abstraction of neural networks using symbolic interpolation," in *Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33013437 pp. 3437–3444.

[28] S. Kotha, C. Brix, Z. Kolter, K. Dvijotham, and H. Zhang, "Provably bounding neural network preimages," *CoRR*, vol. abs/2302.01404, 2023. doi: 10.48550/arXiv.2302.01404

[29] X. Zhang, B. Wang, and M. Kwiatkowska, "On preimage approximation for neural networks," *CoRR*, vol. abs/2305.03686, 2023. doi: 10.48550/arXiv.2305.03686

[30] A. Janusz, A. Zalewska, L. Wawrowski, P. Biczyk, J. Ludziejewski, M. Sikora, and D. Slezak, "Brightbox - A rough set based technology for diagnosing mistakes of machine learning models," *Appl. Soft Comput.*, vol. 141, p. 110285, 2023. doi: 10.1016/j.asoc.2023.110285

[31] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 5283–5292.

[32] P. Cousot and R. Cousot, "Abstract interpretation frameworks," *J. Log. Comput.*, vol. 2, no. 4, pp. 511–547, 1992. doi: 10.1093/logcom/2.4.511

[33] M. Wu and M. Kwiatkowska, "Robustness guarantees for deep neural networks on videos," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00039 pp. 308–317.

[34] H. Zhang, T. Weng, P. Chen, C. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Annual Conference on Neural Information Processing Systems*, 2018, pp. 4944–4953.

[35] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, and C. Hsieh, "Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers," in *9th International Conference on Learning Representations*. OpenReview.net, 2021.

[36] P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Fourth ACM Symposium on Principles of Programming Languages*. ACM, 1977. doi: 10.1145/512950.512973 pp. 238–252.

[37] M. Mirman, T. Gehr, and M. T. Vechev, "Differentiable abstract interpretation for provably robust neural networks," in *35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 3575–3583.

[38] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. T. Vechev, "Fast and effective robustness certification," in *Annual Conference on Neural Information Processing Systems*, 2018, pp. 10 825–10 836.

[39] A. Albarghouthi, "Introduction to neural network verification," *Found. Trends Program. Lang.*, vol. 7, no. 1-2, pp. 1–157, 2021. doi: 10.1561/2500000051

[40] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *Automated Technology for Verification and Analysis*. Springer International Publishing, 2017, pp. 269–286.

[41] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari, "Output range analysis for deep feedforward neural networks," in *10th International Symposium on NASA Formal Methods*, ser. Lecture Notes in Computer Science, vol. 10811. Springer, 2018. doi: 10.1007/978-3-319-77935-5_9 pp. 121–138.

[42] M. Fischetti and J. Jo, "Deep neural networks and mixed integer linear optimization," *Constraints An Int. J.*, vol. 23, no. 3, pp. 296–309, 2018. doi: 10.1007/s10601-018-9285-6

[43] V. Tjeng, K. Y. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *7th International Conference on Learning Representations*. OpenReview.net, 2019.

[44] J. P. Vielma, "Mixed integer linear programming formulation techniques," *SIAM Rev.*, vol. 57, no. 1, pp. 3–57, 2015. doi: 10.1137/130915303

[45] R. Bunel, J. Lu, I. Turkaslan, P. H. S. Torr, P. Kohli, and M. P. Kumar, "Branch and bound for piecewise linear neural network verification," *J. Mach. Learn. Res.*, vol. 21, pp. 42:1–42:39, 2020.

[46] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Formal security analysis of neural networks using symbolic intervals," in *27th USENIX Security Symposium*. USENIX Association, 2018, pp. 1599–1614.

[47] M. Sotoudeh, Z. Tao, and A. V. Thakur, "Syrenn: A tool for analyzing deep neural networks," *Int. J. Softw. Tools Technol. Transf.*, vol. 25, no. 2, pp. 145–165, 2023. doi: 10.1007/s10009-023-00695-1

[48] A. Albarghouthi and K. L. McMillan, "Beautiful interpolants," in *25th International Conference on Computer Aided Verification*, ser. Lecture Notes in Computer Science, vol. 8044. Springer, 2013. doi: 10.1007/978-3-642-39799-8_22 pp. 313–329.

[49] W. Ruan, M. Wu, Y. Sun, X. Huang, D. Kroening, and M. Kwiatkowska, "Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance," in *Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 5944–5952.

[50] T. Baluta, Z. L. Chua, K. S. Meel, and P. Saxena, "Scalable quantitative verification for deep neural networks," in *43rd IEEE/ACM International Conference on Software Engineering*. IEEE, 2021. doi: 10.1109/ICSE43902.2021.00039 pp. 312–323.

[51] P. Yang, R. Li, J. Li, C. Huang, J. Wang, J. Sun, B. Xue, and L. Zhang, "Improving neural network verification through spurious region guided refinement," in *27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, vol. 12651. Springer, 2021, pp. 389–408.

[52] E. L. Malfa, M. Wu, L. Laurenti, B. Wang, A. Hartshorn, and M. Kwiatkowska, "Assessing robustness of text classification through maximal safe radius computation," in *Findings of the Association for Computational Linguistics*, ser. Findings of ACL, vol. EMNLP 2020. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.266 pp. 2949–2968.

[53] E. L. Malfa and M. Kwiatkowska, "The king is naked: On the notion of robustness for natural language processing," in *Thirty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, 2022, pp. 11 047–11 057.

[54] E. Benussi, A. Patanè, M. Wicker, L. Laurenti, and M. Kwiatkowska, "Individual fairness guarantees for neural networks," in *Thirty-First International Joint Conference on Artificial Intelligence*. ijcai.org, 2022. doi: 10.24963/ijcai.2022/92 pp. 651–658.

[55] R. Michelmore, M. Wicker, L. Laurenti, L. Cardelli, Y. Gal, and M. Kwiatkowska, "Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control," in *IEEE International Conference on Robotics and Automation*. IEEE, 2020. doi: 10.1109/ICRA40945.2020.9196844 pp. 7344–7350.

[56] M. Wicker, L. Laurenti, A. Patane, and M. Kwiatkowska, "Probabilistic safety for bayesian neural networks," in *Thirty-Sixth Conference on Uncertainty in Artificial Intelligence*, ser. Proceedings of Machine Learning Research, vol. 124. AUAI Press, 2020, pp. 1198–1207.

[57] M. Wicker, L. Laurenti, A. Patane, N. Paoletti, A. Abate, and M. Kwiatkowska, "Certification of iterative predictions in bayesian neural networks," in *Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, ser. Proceedings of Machine Learning Research, vol. 161. AUAI Press, 2021, pp. 1713–1723.

[58] P. Gourdeau, V. Kanade, M. Kwiatkowska, and J. Worrell, "On the hardness of robust classification," in *Advances in Neural Information Processing Systems*, 2019, pp. 7444–7453.

[59] ——, "On the hardness of robust classification," *Journal of Machine Learning Research*, vol. 22, 2021.

[60] ——, "Sample complexity bounds for robustly learning decision lists against evasion attacks," in *Thirty-First International Joint Conference on Artificial Intelligence*. ijcai.org, 2022. doi: 10.24963/ijcai.2022/419 pp. 3022–3028.

[61] ——, “When are local queries useful?” in *Advances in Neural Information Processing Systems*, 2022.