SMiLE: Provably Enforcing Global Relational Properties in Neural Networks

Matteo Francobaldi[1], **Michele Lombardi**[1], **Andrea Lodi**[2]

[1]DISI, University of Bologna
[2]Jacobs Technion-Cornell Institute, Cornell Tech and Technion
{matteo.francobaldi2, michele.lombardi2}@unibo.it, andrea.lodi@cornell.edu

Abstract

Artificial Intelligence systems are increasingly deployed in settings where ensuring robustness, fairness, or domain-specific properties is essential for regulation compliance and alignment with human values. However, especially on Neural Networks, property enforcement is very challenging, and existing methods are limited to specific constraints or local properties (defined around datapoints), or fail to provide full guarantees. We tackle these limitations by extending SMiLE, a recently proposed enforcement framework for NNs, to support global relational properties (defined over the entire input space). The proposed approach scales well with model complexity, accommodates general properties and backbones, and provides full satisfaction guarantees. We evaluate SMiLE on monotonicity, global robustness, and individual fairness, on synthetic and real data, for regression and classification tasks. Our approach is competitive with property-specific baselines in terms of accuracy and runtime, and strictly superior in terms of generality and level of guarantees. Overall, our results emphasize the potential of the SMiLE framework as a platform for future research and applications.

Code — https://github.com/Francobaldi/SMiLE-AAAI26
Extended Version — https://arxiv.org/abs/2511.07208v1

Introduction

Artificial Intelligence (AI) has witnessed tremendous success in recent years, becoming a pervasive technology. This has been fueled in many domains by Machine Learning (ML) models such as Neural Networks (NN), and by purely data-driven algorithms. While highly effective in terms of accuracy, these methods have difficulties in providing satisfaction guarantees on additional properties, which is critical when regulatory compliance or reliance on expert knowledge are necessary. The former involves safety-critical or ethically sensitive scenarios, where legal frameworks impose specific requirements, such as robustness in autonomous driving (e.g., resisting adversarial attacks in traffic sign recognition) or fairness in automated hiring (e.g., avoiding gender bias in candidate selection). The latter applies to settings that, even though not necessarily critical, might still benefit from the incorporation of properties, like monotonicity in remaining useful life estimation (e.g., the condition of a machine can only deteriorate over time), or adherence to physical laws in scientific modeling (e.g., conserving mass or energy in fluid simulations).

In this work, we consider properties expressible as universally quantified implications between two predicates Q and R, specified over the input and output variables of a ML model $f\colon \mathcal{X} \to \mathcal{Y}$, that is:

$$\forall x_1, .., x_k \in \mathcal{X}, Q(x_1, .., x_k) \Rightarrow R(f(x_1), .., f(x_k)) \quad (1)$$

where each x_i represents a distinct input vector. We distinguish properties according to two criteria, *arity* and *scope*. The arity is *trace* if $k = 1$, *relational* otherwise. The scope is *local* if Q and R depend on data-based information, *global* if they apply also out-of-distribution. Our definition generalizes a broad range of requirements, such as adversarial (local) robustness ($\forall x \in \mathcal{X}\colon \|x - x_0\| \leqslant \delta \Rightarrow f(x) = f(x_0)$, with $x_0 \in D_{\text{train}}$), safety ($\forall x \in \mathcal{X}\colon x \in S_{\text{in}} \Rightarrow f(x) \in S_{\text{out}}$), and monotonicity ($\forall x', x'' \in \mathcal{X}\colon x' \leqslant x'' \Rightarrow f(x') \leqslant f(x'')$). Global relational properties generally allow to express more advanced requirements, but they are also more difficult to handle, as they require searching over the entire domain, as well as reasoning over multiple model evaluations (Banerjee, Xu, and Singh 2024; Blatter et al. 2022). Finally, a property can be enforced with or without *guarantees* on its validity, that is, on the non-existence of a counterexample where such property is violated. Guarantees are crucial in regulatory scenarios, while not always necessary in knowledge injection ones.

Enforcing Properties in ML Regardless of the arity and scope, enforcing properties in complex AI systems such as NNs is challenging due to the non-linear and high-dimensional nature of these models (Katz et al. 2017). The effort in this direction has been mostly spent on trace properties, in particular on adversarial robustness and safety, as surveyed by (Liu et al. 2021; Meng et al. 2022; Muhammad and Bae 2022). Recent years, however, have witnessed a shift of interest to relational ones, in particular fairness, robustness and monotonicity. Most of the proposed approaches embed carefully designed verifiers into training procedures, to promote property feasibility (Tumlin et al. 2024; Athavale et al. 2024; Benussi et al. 2022; Liu et al. 2020; Khedr and Shoukry 2023). Others modify the model architecture to make it systematically feasible (Kitouni, Nolte, and

Williams 2023; Nguyen et al. 2023; Runje and Shankaranarayana 2023; Leino, Wang, and Fredrikson 2021).

Despite the substantial progress achieved in the last decade, critical challenges remain, most notably the lack of guarantees and generality. Most of the existing methods improve the degree of property satisfaction, but without certifying its validity. Moreover, many of them are narrowly tailored to specific properties (e.g., adversarial robustness) or models (e.g., ReLU NNs). To the best of our knowledge, the only existing frameworks capable of providing guarantees while retaining generality are (Goyal, Dumancic, and Blockeel 2024) and (Francobaldi and Lombardi 2025). The two approaches, both limited to trace properties, are based on similar principles: during training they adjust the model weights to guarantee feasibility, in the former case by solving a Quantified SMT problem, in the latter by pairing counterexample generation and Projected Gradient Descent.

Contribution In this work, we extend the framework by (Francobaldi and Lombardi 2025), which was chosen due to the favorable trade-offs provided by its proposed architecture. This is referred to as SMiLE (Safe ML via Embedded overapproximation) and is obtained by augmenting an arbitrarily complex backbone network with a much simpler, trainable overapproximator: the former is used for inference, to enhance expressivity, the latter for enforcement, to improve scalability. Our main contribution is introducing support for relational properties in this framework, through an extensive re-elaboration of its key components. We evaluate the approach on 3 use cases: *monotonicity* on synthetic data, *robustness* on MNIST and *fairness* on Compas, Crime and Law. Across these benchmarks, our method can consistently achieve property guarantees, while remaining competitive with (and sometimes outperforming) property-specific baselines in terms of accuracy and runtime. Finally, we discuss several research directions opened up by our contributions.

Related Work

Two trends mainly arise from the AI literature accounting for properties: verification and enforcing methods. The former aim to formally certify the validity of properties in already-trained models, mostly by relying on optimization and searching techniques – such as Mixed-Integer Programming (MIP) or Satisfability Modulo Theory (SMT) – to seek counterexamples that falsify the assertion. The latter attempt to directly incorporate the desired property into the model's behavior, either by designing property-aligned architectures, or by using existing verifiers into counterexample-guided training loops, or into real-time defense mechanisms.

Most of the effort has been dedicated, thus far, to trace properties. Adversarial training methods, proposed for adversarial robustness (Madry et al. 2018; Zhang et al. 2019; Shafahi et al. 2019; Wang et al. 2020; Zhang et al. 2020b; Wong, Rice, and Kolter 2020; Kim, Lee, and Lee 2021; Bai et al. 2021), local fairness (Mohammadi, Sivaraman, and Farnadi 2023; Benussi et al. 2022), and local monotonicity (Liu et al. 2020), work by training the model over a combination of the original datapoints and their worst counterexamples. Other methods work by detecting and purifying, or rejecting, malicious inputs (Dhillon et al. 2018; Samangouei, Kabkab, and Chellappa 2018; Yang et al. 2019; Pang et al. 2018; Metzen et al. 2017; Xu, Evans, and Qi 2017), or by correcting the output to enforce constraint satisfaction (Wabersich and Zeilinger 2021; Yu, Xu, and Zhang 2022; Mohammadi, Sivaraman, and Farnadi 2023).

While the literature on trace properties is extensive, approaches capable of handling global relational properties have started to recently emerge. From a verification perspective, these methods rely on the same core idea: reducing the relational predicate to a trace one, by encoding multiple independent copies of the same model (product network) within a single verification instance (Banerjee, Xu, and Singh 2024; Tumlin et al. 2024; Athavale et al. 2024). The resulting verifiers can then be employed for property enforcement during training, as in (Benussi et al. 2022) and (Khedr and Shoukry 2023), which design MIP-based verifiers to promote fairness, and in (Liu et al. 2020), which proposes a MIP-based verifier to guarantee monotonicity. Another class of approaches modify the design of the network to make it systematically feasible. (Kitouni, Nolte, and Williams 2023; Nguyen et al. 2023; Runje and Shankaranarayana 2023) constrain the network weights or activations to provide monotonicity guarantees, while (Leino, Wang, and Fredrikson 2021) discourages robustness violations by introducing a violation class. Finally, our work also aligns with Neuro-symbolic AI, which integrates logical reasoning into NNs (Giunchiglia, Stoian, and Lukasiewicz 2022).

Many of these approaches are designed for specific properties, rely on strong architectural assumptions, e.g., small ReLU networks, or fail to provide satisfaction guarantees. To the best of our knowledge, only (Goyal, Dumancic, and Blockeel 2024) and (Francobaldi and Lombardi 2025) offer generality and guarantees, but only for trace properties, while no existing method can handle general relational ones. This work addresses these limitations by proposing a method to enforce generic relational properties in arbitrary networks, providing full satisfaction guarantees while remaining competitive with property-specific baselines.

Methodology

Relational Properties Although theoretically applicable to the general setting defined in Equation (1), in this work we formulate our method for a real-valued neural network $f\colon \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} = \times_{i=1}^{n}[l_i, u_i] \subseteq \mathbb{R}^n$ with $l_i \leqslant u_i\ \forall i$, and a global 2-arity property of the form

$$Q(x', x'') \equiv \ \underline{\delta}_i \leqslant x'_i - x''_i \leqslant \overline{\delta}_i, \forall i, \tag{2a}$$

$$R(f(x'), f(x'')) \equiv \ \underline{\epsilon} \leqslant f(x') - f(x'') \leqslant \overline{\epsilon}, \tag{2b}$$

for given input and output bounds $\underline{\delta}, \overline{\delta} \in \mathbb{R}^m$ and $\underline{\epsilon}, \overline{\epsilon} \in \mathbb{R}$. The most common relational properties targeted in the literature can be derived from Equation (2) by configuring its parameters, as shown in Table 1, where $\delta, \epsilon \geqslant 0$, $M \in \mathbb{R}^+$ is a sufficiently large value used to relax unnecessary bounds, while $\mathcal{P}, \mathcal{P}^c \subseteq [m]$ denote the set of sensitive features (protected for fairness and monotonic for monotonicity) and its complement, respectively. Intuitively: 1) in the robustness case, bounded input variations should translate into bounded

Property	$\underline{\delta}_{\mathcal{P}}$	$\underline{\delta}_{\mathcal{P}^c}$	$\overline{\delta}_{\mathcal{P}}$	$\overline{\delta}_{\mathcal{P}^c}$	$\underline{\epsilon}$	$\overline{\epsilon}$
Robustness	$-\delta$	$-\delta$	δ	δ	$-\epsilon$	ϵ
Fairness	$-M$	0	M	0	$-\epsilon$	ϵ
Monotonicity ($\uparrow$)	$-M$	0	0	0	$-M$	0

Table 1: Configuration of δ and ϵ for different properties.

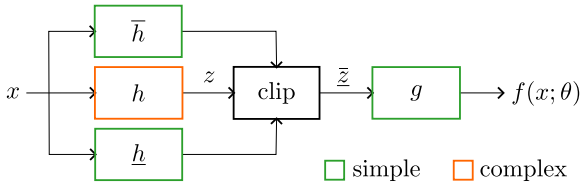

Figure 1: A depiction of the SMiLE architecture.

output variations; 2) in the fairness case, when only the protected features change, the corresponding output change should be limited; 3) in the monotonicity case, when the sensitive features increase, the output cannot decrease.

SMiLE SMiLE is an enforcement framework consisting of a verification-friendly neural architecture and a dedicated training algorithm (Francobaldi and Lombardi 2025). A SMiLE architecture can be built by decomposing a network f into an arbitrary embedding function h and a linear output function g, then by embedding a trainable overapproximator in between them, consisting of *lower and upper auxiliary models* $\underline{h}, \overline{h}$, and a *clipping operator* $\text{clip}(z; l; u) = \max(l, \min(u, z))$:

$$g(\bar{z}; \theta_g) \circ \text{clip}(z; \underline{h}(x; \theta_{\underline{h}}); \overline{h}(x; \theta_{\overline{h}})) \circ h(x; \theta_h). \quad (3)$$

The model structure, depicted in Figure 1, ensures that the input to the g function is contained into the bounding box

$$\overline{\underline{H}}(x; \theta_{\underline{h}}, \theta_{\overline{h}}) = [\underline{h}_1(x), \overline{h}_1(x)] \times \ldots \times [\underline{h}_n(x), \overline{h}_n(x)], \quad (4)$$

where n is the size of the embedding vector. Many neural architectures employed in AI research and real-world applications can be easily adapted to this structure, including classifiers, by focusing on their logit output.

The original SMiLE training algorithm relies on two main components: 1) a generator, which searches for a counterexample that violates the property, and 2) a projector, which minimally adjusts the model parameters to restore the counterexample feasibility. Training proceeds in two phases. In the first phase (simply referred to as training), the generator and projector routines are embedded into a standard gradient-based algorithm, and executed after each gradient step to maximize accuracy while maintaining feasibility. In the second phase (referred to as post-training), the update step is deactivated, and only the generation-projection loop is repeated to eliminate residual violations. If this process is successful, the resulting SMiLE model provably satisfies the desired property without requiring further intervention.

The key idea of the framework is to exploit the SMiLE architecture to speed up the generation and projection phase, which would be otherwise intractable for standard networks

Algorithm 1: TRAIN($L_s, \theta, \lambda_{\text{box}}, t, x, y$, SGD-pars)

Pretraining:
for pretraining step **do**
  $L \leftarrow \text{PRETRLOSS}(L_s, \theta, \lambda_{\text{box}}, x, y)$
  $\theta \leftarrow \text{PRIMALSTEP}(L)$
Training:
$x', x'', \bar{z}', \bar{z}'', \overline{\gamma}, S \leftarrow \text{GENERATE}(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, t)$
$\lambda_{\text{prop}} \leftarrow 0$
for training step **do**
  if $\text{RESOLVED}(x', x'', \bar{z}', \bar{z}'')$ **then**
    $x', x'', \bar{z}', \bar{z}'', \overline{\gamma}, S \leftarrow \text{GENERATE}(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, t)$
  $L \leftarrow \text{TRLOSS}(L_s, \theta, \lambda_{\text{box}}, \lambda_{\text{prop}}, x, y, x', x'', \bar{z}', \bar{z}'')$
  $\theta \leftarrow \text{PRIMALSTEP}(L)$
  $\lambda_{\text{prop}} \leftarrow \text{DUALSTEP}(L)$
Posttraining:
for posttraining step **do**
  $x', x'', \bar{z}', \bar{z}'', \overline{\gamma}, S \leftarrow \text{GENERATE}(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, t)$
  if $S =$ infeasible **then**
    `ViolBound` $\leftarrow 0$
    break
  $\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g \leftarrow \text{PROJECT}(x', x'', \bar{z}', \bar{z}'')$
`ViolBound` $\leftarrow \overline{\gamma}$
return θ, `ViolBound`

(Katz et al. 2017). When executing these steps, the method relaxes the arbitrarily complex constraints $z = h(x; \theta_h)$ into the simpler bounding box constraints $\underline{h}(x; \theta_{\underline{h}}) \leqslant \bar{z} \leqslant \overline{h}(x; \theta_{\overline{h}})$, while preserving the output ones $f(x, \theta) = g(\bar{z}; \theta_g)$. In other words, the complex h is ignored and only the much simpler $\underline{h}, \overline{h}$ and g are used. Since g is simple by construction, and $\underline{h}$ and $\overline{h}$ can be made simple by design, the complexity is *freely controllable*. Note that ignoring the exact component h in favor of the overapproximation computed by $\underline{h}$ and $\overline{h}$ preserves verification soundness (and hence feasibility), but may result in over-enforcement.

Reengineered Training The framework from (Francobaldi and Lombardi 2025) is limited to trace properties. While the main advantage of the SMiLE architecture (i.e., simplified verification) readily applies to relational properties, attempting to reuse its original training algorithm – with straightforward adaptations – proved entirely inadequate to the new setting from Equation (2). This motivated us to design a thoroughly reengineered training approach that is more general, scalable, and stable than its predecessor.

The procedure is described in Algorithm 1 and it is articulated in three sections, referred to as *pretraining*, *training*, and *posttraining*. In the pretraining phase, the architecture is trained via gradient descent (PRIMALSTEP) to approximately satisfy certain regularization properties that greatly reduce the chance of getting stuck in poor local optima. The training phase relies on Dual Ascent to reduce both estimation errors and property violations. This is done by iteratively: 1) generating pairs of examples, i.e., a counterexample, with large violation (GENERATE); if a counterexample from past iterations is available, we check its valid-

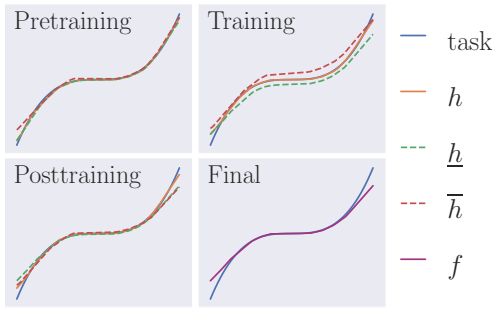

Figure 2: The training evolution of a SMiLE model f with latent dimension $n = 1$ and output function $g = \mathrm{id}$ for the task $y = x^3$, enforced with $|x' - x''| \leqslant 0.1 \implies |f(x') - f(x'')| \leqslant 0.5 \max_{x \in \mathcal{X}, \delta \leqslant 0.1} |x^3 - (x + \delta')^3|$, i.e., we want the model to be 0.5 times more robust than the task itself.

Algorithm 2: PRETRLOSS$(L_s, \theta, \lambda_{\text{box}}, x, y)$

$L_f \leftarrow L_s(y, f(x;\theta))$
$L_{\underline{h}} \leftarrow L_s(y, g(\underline{z};\theta_g) \circ \underline{h}(x;\theta_{\underline{h}}))$
$L_{\overline{h}} \leftarrow L_s(y, g(\overline{z};\theta_g) \circ \overline{h}(x;\theta_{\overline{h}}))$
$L_h \leftarrow L_s(y, g(z;\theta_g) \circ h(x;\theta_h))$
$L_{\overline{\text{box}}} \leftarrow \max(0, h(x;\theta_h) - \overline{h}(x,\theta_{\overline{h}}))$
$L_{\underline{\text{box}}} \leftarrow \max(0, \underline{h}(x;\theta_{\underline{h}}) - h(x;\theta_h))$
$L_{\text{acc}} \leftarrow L_f + L_{\underline{h}} + L_{\overline{h}} + L_h, \quad L_{\text{box}} \leftarrow L_{\overline{\text{box}}} + L_{\underline{\text{box}}}$
return $L_{\text{acc}} + \lambda_{\text{box}} L_{\text{box}}$

ity (RESOLVED) before generating a new one. 2) Using the counterexample to define a Lagrangian penalty term that is incorporated in the loss function (TRLOSS). 3) Performing a regular gradient descent step, followed by a gradient ascent step over the penalty term multiplier (DUALSTEP). The dual step increases the contribution of the property violation in the loss function, so that it is prioritized by the gradient descent process, with the aim to reach 0-violation. When this is not possible, in the posttraining phase we attempt to adjust the model weights so that no counterexample exists anymore. This process is also based on dual ascent, but implements a projection operator that does not depend on the training data. In what follows, we provide details on the design of the critical subroutines in the algorithm. The remaining subroutines are described in the paper appendix.

Pretraining Loss When adapting SMiLE for relational properties, we observed that traditional weight initialization strategies had a tendency to cause $\underline{h}$ and $\overline{h}$ to flip (the lower bound becomes higher than the upper bound). When this happens, the semantic of the clipping operator prevents most gradient components from being backpropagated, causing catastrophic training failure.

We address this issue by pretraining the model with a custom loss, outlined in Algorithm 2, and featuring two types of terms. On the one hand, the terms summed in L_{acc}, specified via a traditional loss function L_s (i.e., MSE or BCE), maximize the accuracy of *all* the possible output pathways in the SMiLE computation graph, i.e., $g \circ h$, $g \circ \underline{h}$, $g \circ \overline{h}$. On the other hand, L_{box} penalizes overapproximation degeneracy, occurring when $\underline{h}, \overline{h}$ and h violate the constraints $\underline{h}(x;\theta_{\underline{h}}) \leqslant h(x;\theta_h) \leqslant \overline{h}(x;\theta_{\overline{h}})$. Figure 2 shows the outcome of the pretraining phase for data generated according to a simple univariate function: it can be seen that all output pathways provide similar results, and no significant flipping occurs. The original SMiLE algorithm included only a basic pretraining step, where the entire architecture was optimized for accuracy, which regularly led to training failure in our preliminary experiments.

Generator Intuitively, the counterexample generator needs to search for *a pair* of input vectors violating the target property. In practice, we wish to use the SMiLE overapproximator to disregard the backbone network h and accelerate this process. Hence, we actually search for a pair of compound *input and embedding vectors* $(x', x''\overline{z}', \overline{z}'') \in \mathcal{X}^2 \times \underline{H}(x';\theta_{\underline{h}},\theta_{\overline{h}}) \times \overline{H}(x'';\theta_{\underline{h}},\theta_{\overline{h}})$. In other words, the embedding $\overline{z}', \overline{z}''$ are only required to be in the overapproximation boxes associated with x' and x''. This is done by solving the following problem:

$$\arg\max_{x', x'', \overline{z}', \overline{z}''} \ \gamma \tag{5a}$$
$$\text{s.t. } l \leqslant x', x'' \leqslant u \tag{5b}$$
$$\underline{\delta} \leqslant x' - x'' \leqslant \overline{\delta} \tag{5c}$$
$$\underline{h}(x';\theta_{\underline{h}}) \leqslant \overline{z}' \leqslant \max(\underline{h}(x';\theta_{\underline{h}}), \overline{h}(x';\theta_{\overline{h}})) \tag{5d}$$
$$\underline{h}(x'';\theta_{\underline{h}}) \leqslant \overline{z}'' \leqslant \max(\underline{h}(x'';\theta_{\underline{h}}), \overline{h}(x'';\theta_{\overline{h}})) \tag{5e}$$
$$y' = g(\overline{z}';\theta_g) \tag{5f}$$
$$y'' = g(\overline{z}'';\theta_g) \tag{5g}$$
$$\gamma \leqslant y' - y'' - \overline{\epsilon} + Mb \tag{5h}$$
$$\gamma \leqslant -y' + y'' + \underline{\epsilon} + M(1 - b) \tag{5i}$$
$$x', x'' \in \mathbb{R}^{B^c} \times \{0,1\}^B, \overline{z}', \overline{z}'' \in \mathbb{R}^n, \ y', y'' \in \mathbb{R} \tag{5j}$$
$$b \in \{0,1\}, \ \gamma \in \mathbb{R}_{>0} \tag{5k}$$

where $M \geqslant 0$ is a fixed big-M value, and $B, B^c \subseteq [m]$ respectively denote the set of binary variables and its complement, necessary when the learning task involves binary or one-hot encoded features.

Solving Problem (5) is dramatically easier than searching for counterexamples considering the actual backbone h, but can still be challenging, especially if moderately complex auxiliary models $\underline{h}, \overline{h}$ are used or if the model input is high-dimensional. We propose to speed up generation by observing that finding the most violated counterexample is not strictly necessary for enforcement to work. At the same time, however, complete search is needed (at least once) to determine feasibility – i.e. to check whether a counterexample exists at all. We meet these apparently contradicting needs by solving Problem (5) with a timeout extension scheme. Namely: 1) we start the solution process with a timeout; 2) if any counterexample is found or infeasibility is proven, we stop; 3) otherwise, we double the timeout and continue searching; 4) we proceed until the timeout reaches a maximum allowed value, at which point we stop. In the latter case, since we use Mathematical Programming as our solu-

Algorithm 3: TRLOSS$(L_s, \theta, \lambda_{box}, \lambda_{prop}, x, y, x', x'', \underline{z}', \underline{z}'')$

1: $L_{acc} \leftarrow L_s(y, f(x;\theta))$
2: $L_{\overline{box}} \leftarrow \max(0, h(x;\theta_h) - \overline{h}(x, \theta_{\overline{h}}))$
3: $L_{\underline{box}} \leftarrow \max(0, \underline{h}(x;\theta_{\underline{h}}) - h(x;\theta_h))$
4: $L_{box} \leftarrow L_{\overline{box}} + L_{\underline{box}}$
5: **if** $(x', x'', \underline{z}', \underline{z}'') \neq$ null **then**
6: $\quad y' \leftarrow$ CEPROP$(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, x', \underline{z}')$
7: $\quad y'' \leftarrow$ CEPROP$(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, x'', \underline{z}'')$
8: $\quad L_{prop} \leftarrow \max(0, \max(\underline{\varepsilon} - y' + y'', y' - y'' - \bar{\varepsilon}))$
9: **else**
10: $\quad L_{prop} \leftarrow 0$
11: **return** $L_{acc} + \lambda_{box} L_{box} + \lambda_{prop} L_{prop}$

Algorithm 4: PROJECT$(x', x'', \underline{z}', \underline{z}'', \theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g,$ SGD-pars$)$

1: $\theta^0_{\underline{h}} \leftarrow \theta_{\underline{h}},\ \theta^0_{\overline{h}} \leftarrow \theta_{\overline{h}},\ \theta^0_g \leftarrow \theta_g, \lambda_{prop} \leftarrow 0$
2: **for** projection step **do**
3: $\quad y' \leftarrow$ CEPROP$(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, x', \underline{z}')$
4: $\quad y'' \leftarrow$ CEPROP$(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, x'', \underline{z}'')$
5: $\quad L_{acc} \leftarrow \|\theta^0_{\underline{h}} - \theta_{\underline{h}}\|_2^2 + \|\theta^0_{\overline{h}} - \theta_{\overline{h}}\|_2^2 + \|\theta^0_g - \theta_g\|_2^2$
6: $\quad L_{prop} \leftarrow \max(0, \max(\underline{\varepsilon} - y' + y'', y' - y'' - \bar{\varepsilon}))$
7: $\quad L \leftarrow L_{acc} + \lambda_{prop} L_{prop}$
8: $\quad \theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g \leftarrow$ PRIMALSTEP(L)
9: $\quad \lambda_{prop} \leftarrow$ DUALSTEP(L)
10: **return** $\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g$

tion technology, we can still provide a bound on the maximum violation level for the network. The procedure is described in the supplemental material. The original generator from (Francobaldi and Lombardi 2025) was limited to trace properties and always relied on complete search, which we found to be impractical in our setting.

Training Loss During our training phase, we use the multi-component loss function described in Algorithm 3, which incorporates: 1) a term L_{acc} accounting for model accuracy; 2) a term L_{box} penalizing box degeneracy, analogously to Algorithm 2; 3) a term L_{prop} representing the degree of violation for the incumbent counterexample. At line 11, the multiplier λ_{box} for L_{box} is fixed, while the one for L_{prop} is trainable. During the primal gradient step, we differentiate w.r.t. all θ parameters and move opposite to the gradient (to reduce the loss value). During the dual gradient step, we differentiate w.r.t. λ_{box} and move in the same direction as the gradient, to increase the penalty in case of violations. Results from classical penalty methods ensure that the process asymptotically convergences to a local optimum.

Designing the L_{prop} term is challenging, as it should not only allow for backpropagation, but also be non-trivial to resolve. Specifically, in our considered setting we have that, due to the linearity of g, for any locally optimal counterexample the two embeddings always lie at one vertex of their bounding boxes: $\forall i \in [n],\ \underline{z}'_i = \underline{h}(x;\theta_{\underline{h}}) \vee \underline{z}'_i = \overline{h}(x;\theta_{\overline{h}})_i$, and similarly for $\underline{z}''$. In this situation, the counterexample could be resolved by making minor adjustments to the auxiliary models until either $\underline{z}'$ or $\underline{z}''$ lie outside the overapproximation box. In turn, this would lead to very frequent generator calls and impractically large training times.

We address these issues by applying an *abstraction step* to our counterexamples. Namely, we represent them through the associated active constraints from the overapproximation box, i.e., either $\underline{z}'_i = \underline{h}(x;\theta_{\underline{h}})_i$ or $\underline{z}'_i = \overline{h}(x;\theta_{\overline{h}})_i$. The CEPROP routine manages this step and is described in the supplemental material. An example outcome for the main training loop can be found in Figure 2: during the dual ascent process, the overapproximation defined by $\underline{h}, \overline{h}$ has expanded, allowing the backbone to more accurately approximate the underlying function in the regions where its shape is compatible with the target constraint (robustness).

The main training loop in (Francobaldi and Lombardi 2025) is based on Projected Gradient Descent (PGD), interleaving primal gradient steps with optimal projection steps limited to the output layer weights. The latter restriction proved enough to make the approach ineffective in the case of relational properties. In addition, our method reduces the zig-zag behavior typical of PGD, hence accelerating convergence, and requires significantly fewer generator calls.

Posttraining The main training loop is effective at improving both the accuracy and the degree of property satisfaction of the model. However, the need to manage those two, often conflicting goals prevents reaching guaranteed feasibility in most cases. Typically, at this stage only counterexamples associated to modest violation values remain, which we address in the posttraining phase.

For this phase, we use a projection operator implemented via gradient steps, detailed in Algorithm 4. Formally, we rely once again on dual ascent, but compared with the main training loop the accuracy term L_{acc} is not given by a data-driven loss, but rather consists of a squared L2 regularizer on the model weights. Upon termination, the loop returns the model weights θ, together with a certified upper bound on property violation `ViolBound`, whose value depends on the termination condition: if we manage to prove infeasibility in the generator, this value is zero and the model has satisfaction guarantees; otherwise, the bound can still serve as a form of partial certification. Figure 2 shows an example outcome for the post-training phase, where the overapproximation box has tightened again to prevent violations.

Experimentation

We evaluate our framework on three benchmarks: *monotonicity*, *fairness* and *robustness*, guided by the following research questions: *(Q1: Guarantees)* Can SMiLE consistently provide property satisfaction guarantees? *(Q2: Accuracy)* Is the accuracy of SMiLE competitive with state-of-the-art property-specific methods? *(Q3: Applicability)* Can SMiLE provide practically valuable outcomes? *(Q4: Generality)* Can SMiLE consistently enforce different properties on different neural architectures?

Our experiments are developed in Python: we implemented SMiLE by using Keras (Chollet et al. 2015), Pyomo (Hart, Watson, and Woodruff 2011; Bynum et al. 2021), OMLT (Ceccon et al. 2022) and Gurobi (Gurobi Optimiza-

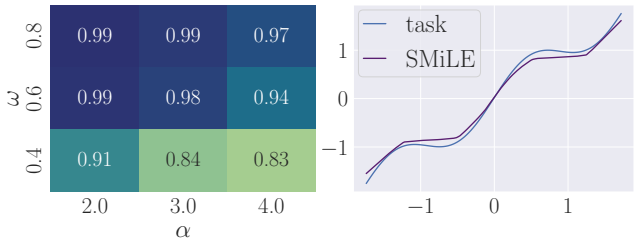

Figure 3: Accuracy on *monotonicity* (left) and an example of a monotonic estimation of a non-monotonic function (right).

tion, LLC 2024), while for the considered competitors we used the official code released by their authors.

Monotonicity We consider 9 synthetic tasks given by the function $y = x + \alpha \sin(\omega x)$, for $\alpha \in \{2, 3, 4\}$ and $\omega \in \{0.4, 0.6, 0.8\}$, which becomes increasingly non-monotonic as α and ω increase. On each task, we train a SMiLE model by enforcing non-decreasing monotonicity. Precisely, for any input pair x', x'' such that $x' \leqslant x''$, we force the corresponding outputs to satisfy $f(x') \leqslant f(x'')$. Figure 3 depicts a heatmap with the R^2 performance of our model (the higher, the better), and an example of a monotonic approximation of the non-monotonic function $\alpha = 2$ and $\omega = 0.6$, showing how our method can achieve acceptable results even when the property is substantially violated in the data. In all cases, 0-violation was reached after posttraining.

Robustness We consider the task of detecting the digit "zero" (binary classification) on the MNIST dataset (LeCun et al. 1998). We train 15 SMiLE models, each time by enforcing, on the model logit, a δ, ϵ-robustness property with $\delta \in \{0.010, 0.025, 0.050, 0.075, 0.100\}$ and $\epsilon \in \{0.75, 1.00, 1.25\}$. Precisely, for any input pair x', x'' such that $\|x' - x''\|_\infty \leqslant \delta$, we force the corresponding logits to satisfy $|f_{\text{logit}}(x') - f_{\text{logit}}(x'')| \leqslant \epsilon$.

The global robustness guarantees from our method can be used to implement a *constant-time* rejection-based defense against adversarial attacks. The defense checks whether the logit $f_{\text{logit}}(x)$, for a test input x, lies outside the rejection region $[-\epsilon, \epsilon]$. In this case, it issues a safety certificate: even if x was adversarially generated from a clean input as specified in the property, the attack would be unable to change the logit by more than ϵ, and hence to flip the prediction. In the opposite case, the defense raises a warning to the user.

We evaluate our framework against CROWN-IBP (Zhang et al. 2020a), which penalizes property violation by training the model against a convex combination of a standard and a robustness loss, computed via a sound but incomplete robustness verifier, integrating CROWN (Zhang et al. 2018) with IBP (Gowal et al. 2019). The method depends on both the perturbation δ and a convex combination hyperparameter λ, dynamically adjusted according to a scheduling strategy. At training time, CROWN-IBP enforces robustness only locally (i.e., around the training samples); at inference time, it provides a defense mechanism similar to the one described above, except that the rejection region needs to be computed for each sample by solving an overapproximation. We train CROWN-IBP for the same values of δ considered for SMiLE. Together with it, we also consider a base model unaware of the property to satisfy (Agnostic). In all approaches, we adopt a *deep convolutional neural network* for h, and *linear models* for $\underline{h}, \overline{h}$.

δ	*Model*	*Clean*	*PGD*	*Verified*
0.010	Agnostic	99.60	99.50	00.00
0.010	CROWNIBP	100.00	99.90	99.90
0.010	SMiLE	98.80	98.47	97.57
0.025	Agnostic	99.60	98.40	00.00
0.025	CROWNIBP	99.80	99.70	99.60
0.025	SMiLE	98.60	97.90	96.37
0.050	Agnostic	99.60	90.40	00.00
0.050	CROWNIBP	99.70	99.40	99.20
0.050	SMiLE	97.87	96.07	94.33
0.075	Agnostic	99.60	75.70	00.00
0.075	CROWNIBP	99.60	99.20	99.20
0.075	SMiLE	97.07	94.20	92.37
0.100	Agnostic	99.60	66.80	00.00
0.100	CROWNIBP	99.60	99.30	99.00
0.100	SMiLE	95.87	92.17	90.57

Table 2: Accuracy on *robustness*.

The predictive performance of the three competitors on the first 1,000 instances of the MNIST dataset is reported in Table 2, where the SMiLE results are aggregated across the different ϵ, and where *Clean*, *PGD* and *Verified* denote the percentage of correct predictions on the unperturbed test set, on the test set under a 100-step PGD attack, and the percentage of correct and verified predictions under any attack, respectively. We highlight, moreover, that all reported SMiLE models are guaranteed robust: their training successfully terminated with zero property violation, enabling the corresponding real-time defense. CROWN-IBP emerges as the most accurate model across all metrics. Agnostic maintains constant clean accuracy and zero verified one, being unaffected by δ and unable to certify, while its PGD accuracy drops significantly under stronger attacks, highlighting its vulnerability. SMiLE, on the other hand, while less accurate than the property-specific CROWN-IBP, exhibits a moderate decline in clean, verified, and PGD accuracy as attacks intensify, demonstrating its ability to produce verifiable and also significantly more robust networks than a baseline approach. Finally, while SMiLE may not match CROWN-IBP in terms of accuracy, it clearly outperforms it in terms of runtime: we report an average training time of 14,138s for CROWN-IBP, and of 7,717s for SMiLE; more importantly, at inference time, CROWN-IBP requires a total of 11.20s to both predict and verify on the test set, while SMiLE employs only 0.03s for the same procedure, comparable with the base model, which predicts in 0.02s but without certifying.

Fairness We consider 3 widely used datasets in the fairness literature: Compas (Angwin et al. 2016), Law (Wightman 1998) and Crime (Redmond and Baveja 2002), where

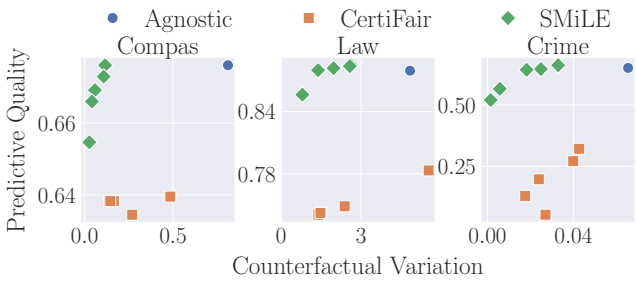

Figure 4: Accuracy and counterfactual variation on *fairness*.

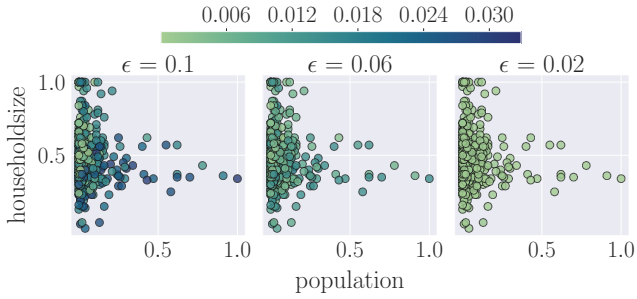

Figure 5: Counterfactual variation for different ϵ on Crime.

the task is to predict recidivism, law school admission and crime rate, respectively. The first two are binary classification tasks, while the third is a regression one. On each dataset, we train 5 SMiLE models, each time by enforcing, on the model output (logit), an ϵ-fairness property, with $\delta = 0$ and $\epsilon \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ for Compas, $\epsilon \in \{1, 2, 3, 4, 5\}$ for Law and $\epsilon \in \{0.02, 0.04, 0.06, 0.08, 0.10\}$ for Crime. Precisely, for any input pair x', x'' such that $|x'_i - x''_i| \leqslant \delta \ \forall i \neq p$ (i.e., x', x'' are mutually counterfactual), we force the corresponding outputs (logits) to satisfy $|f(x') - f(x'')| \leqslant \epsilon$, where the protected variable p is always *race*. We evaluate our framework against CertiFair (Khedr and Shoukry 2023), which penalizes property violation by training the model against a convex combination of a standard and a fairness loss, computed via a sound but incomplete fairness verifier. The method depends on a combination hyperparameter λ, while is independent of the counterfactual perturbation δ. At training time, similarly to us, our competitor enforces fairness globally (i.e., on the entire input space); at inference time, differently from us, it lacks property satisfaction guarantees on individual inputs. We train CertiFair for 5 values of λ, calibrated to span the reasonable hyperparameter space for each dataset: $\lambda \in \{0.025, 0.050, 0.075, 0.100, 0.125\}$ for Compas and Law, and $\lambda \in \{0.08, 0.09, 0.10, 0.11, 0.12\}$ for Crime. Together with it, we consider again Agnostic. In all approaches, we adopt a *deep feedforward neural network* for h, and 1-hidden-layer *ReLU neural networks* for $\underline{h}, \overline{h}$.

We report the results of the experiment in Figure 4, where each dot represents a model evaluated along two dimensions, *Predictive Quality* and *Counterfactual Variation*: the former corresponds to R^2 or accuracy (the higher, the better), the latter to the maximum absolute difference in model output (logit) on any pair of counterfactual samples from the test set, quantifying unfairness (the lower, the better), where the counterfactual of a sample is obtained by flipping its protected attribute. The figure clearly demonstrates the superiority of our method across all datasets: SMiLE models are consistently positioned to the left of Agnostic models, indicating a substantially higher degree of fairness compared to the baseline approach, as well as above and generally to the left of CertiFair models, showing that our method achieves higher accuracy while guaranteeing an equivalent or superior level of fairness relative to its competitor. We highlight that the reported SMiLE models achieve again zero property violation, meaning that besides decreasing the counterfactual variation in-distribution (on the test set), we are also able to guarantee a variation upper bound (i.e., the enforced ϵ) out of distribution. Finally, Figure 5 shows how the counterfactual variation of SMiLE (color gradient) decreases with ϵ, over the pair of counterfactual samples (dots) from the Crime test set, projected on two non-protected features.

Discussion *(Q1: Guarantees)* Each SMiLE model in our computational study successfully converged to zero property violation, demonstrating that, in practice, our framework can consistently provide satisfaction guarantees. *(Q2: Accuracy)* While SMiLE is outperformed by CROWN-IBP on *robustness*, it shows high accuracy on *monotonicity* and *fairness*, where it matches Agnostic and surpasses CertiFair, highlighting its competitive predictive capabilities. *(Q3: Applicability)* On *robustness* and *fairness*, SMiLE is able to produce more robust and fair networks, as well as to enable a very fast real-time defense against adversarial attacks, yielding practically valuable outcomes. *(Q4: Generality)* The strong results achieved across three relational properties (monotonicity, robustness, and fairness), and for different neural architectures (convolutional and feedforward), demonstrate the broad generality of SMiLE.

Conclusion

We extended the property-enforcement SMiLE framework, designed for trace properties, to also support relational ones, by retaining its verification-friendly architecture, while completely reengineering its training algorithm. Precisely, we designed a procedure consisting of 3 phases: *pretraining*, which suitably initializes the model to prevent poor local optima; *training*, which maximizes accuracy while encouraging feasibility via a counterexample-based Dual Gradient Ascent; and *posttraining*, which eliminates residual violations by iteratively generating and resolving counterexamples. We tested our framework on *monotonicity*, *robustness* and *fairness*, for regression and classification tasks, demonstrating its ability to provide satisfaction guarantees for generic relational properties and networks, while remaining competitive with property-specific methods. Our contributions open up several research directions, such as further extending the SMiLE framework to *functional properties*, constraining input and output together into the same predicates, or adapting it to inputs of variable size, via Transformers or Graph Neural Networks.

Acknowledgements

The project leading to this application has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101070149.

References

Angwin, J.; Larson, J.; Mattu, S.; and Kirchner, L. 2016. Machine bias. *ProPublica*.

Athavale, A.; Bartocci, E.; Christakis, M.; Maffei, M.; Nickovic, D.; and Weissenbacher, G. 2024. Verifying Global Two-Safety Properties in Neural Networks with Confidence. In *Computer Aided Verification: 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24–27, 2024, Proceedings, Part II*, 329–351. Springer-Verlag.

Bai, T.; Luo, J.; Zhao, J.; Wen, B.; and Wang, Q. 2021. Recent Advances in Adversarial Training for Adversarial Robustness. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 4312–4321. International Joint Conferences on Artificial Intelligence Organization.

Banerjee, D.; Xu, C.; and Singh, G. 2024. Input-Relational Verification of Deep Neural Networks. *Proc. ACM Program. Lang.*, 8(PLDI).

Benussi, E.; Patane, A.; Wicker, M.; Laurenti, L.; and Kwiatkowska, M. 2022. Individual Fairness Guarantees for Neural Networks. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, 651–658. International Joint Conferences on Artificial Intelligence Organization.

Blatter, L.; Kosmatov, N.; Prevosto, V.; and Le Gall, P. 2022. Certified Verification of Relational Properties. In ter Beek, M. H.; and Monahan, R., eds., *Integrated Formal Methods*, 86–105. Springer International Publishing.

Bynum, M. L.; Hackebeil, G. A.; Hart, W. E.; Laird, C. D.; Nicholson, B. L.; Siirola, J. D.; Watson, J.-P.; and Woodruff, D. L. 2021. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, third edition.

Ceccon, F.; Jalving, J.; Haddad, J.; Thebelt, A.; Tsay, C.; Laird, C. D.; and Misener, R. 2022. OMLT: Optimization & Machine Learning Toolkit. *Journal of Machine Learning Research*, 23(349): 1–8.

Chollet, F.; et al. 2015. Keras. https://keras.io. Accessed: 2025-11-15.

Dhillon, G. S.; Azizzadenesheli, K.; Lipton, Z. C.; Bernstein, J.; Kossaifi, J.; Khanna, A.; and Anandkumar, A. 2018. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*.

Francobaldi, M.; and Lombardi, M. 2025. SMLE: Safe Machine Learning via Embedded Overapproximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 26, 27286–27294.

Giunchiglia, E.; Stoian, M. C.; and Lukasiewicz, T. 2022. Deep Learning with Logical Constraints. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 5478–5485. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Gowal, S.; Dvijotham, K.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T.; and Kohli, P. 2019. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models. arXiv:1810.12715.

Goyal, K.; Dumancic, S.; and Blockeel, H. 2024. Deepsade: Learning neural networks that guarantee domain constraint satisfaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 12199–12207.

Gurobi Optimization, LLC. 2024. Gurobi Optimizer Reference Manual.

Hart, W. E.; Watson, J.-P.; and Woodruff, D. L. 2011. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3(3): 219–260.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, 97–117. Springer.

Khedr, H.; and Shoukry, Y. 2023. CertiFair: a framework for certified global fairness of neural networks. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

Kim, H.; Lee, W.; and Lee, J. 2021. Understanding catastrophic overfitting in single-step adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 9, 8119–8127.

Kitouni, O.; Nolte, N.; and Williams, M. 2023. Expressive monotonic neural networks. *arXiv preprint arXiv:2307.07512*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Leino, K.; Wang, Z.; and Fredrikson, M. 2021. Globally-robust neural networks. In *International Conference on Machine Learning*, 6212–6222. PMLR.

Liu, C.; Arnon, T.; Lazarus, C.; Strong, C.; Barrett, C.; and Kochenderfer, M. J. 2021. Algorithms for Verifying Deep Neural Networks. *Foundations and Trends in Optimization*, 4(3-4): 244–404.

Liu, X.; Han, X.; Zhang, N.; and Liu, Q. 2020. Certified monotonic neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

Meng, M. H.; Bai, G.; Teo, S. G.; Hou, Z.; Xiao, Y.; Lin, Y.; and Dong, J. S. 2022. Adversarial robustness of deep neural networks: A survey from a formal verification perspective. *IEEE Transactions on Dependable and Secure Computing*.

Metzen, J. H.; Genewein, T.; Fischer, V.; and Bischoff, B. 2017. On Detecting Adversarial Perturbations. In *International Conference on Learning Representations*.

Mohammadi, K.; Sivaraman, A.; and Farnadi, G. 2023. FETA: Fairness Enforced Verifying, Training, and Predicting Algorithms for Neural Networks. In *Proceedings of the 3rd ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*, EAAMO '23. Association for Computing Machinery.

Muhammad, A.; and Bae, S.-H. 2022. A survey on efficient methods for adversarial robustness. *IEEE Access*, 10: 118815–118830.

Nguyen, A.-P.; Moreno, D. L.; Le-Bel, N.; and Rodríguez Martínez, M. 2023. MonoNet: enhancing interpretability in neural networks via monotonic features. *Bioinformatics advances*, 3(1).

Pang, T.; Du, C.; Dong, Y.; and Zhu, J. 2018. Towards Robust Detection of Adversarial Examples. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Redmond, M.; and Baveja, A. 2002. Communities and Crime Data Set. https://archive.ics.uci.edu/ml/datasets/communities+and+crime. UCI Machine Learning Repository.

Runje, D.; and Shankaranarayana, S. M. 2023. Constrained monotonic neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Samangouei, P.; Kabkab, M.; and Chellappa, R. 2018. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. In *International Conference on Learning Representations*.

Shafahi, A.; Najibi, M.; Ghiasi, M. A.; Xu, Z.; Dickerson, J.; Studer, C.; Davis, L. S.; Taylor, G.; and Goldstein, T. 2019. Adversarial training for free! *Advances in neural information processing systems*, 32.

Tumlin, A. M.; Manzanas Lopez, D.; Robinette, P.; Zhao, Y.; Derr, T.; and Johnson, T. T. 2024. FairNNV: The Neural Network Verification Tool For Certifying Fairness. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF '24, 36–44. Association for Computing Machinery.

Wabersich, K. P.; and Zeilinger, M. N. 2021. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129: 109597.

Wang, Y.; Zou, D.; Yi, J.; Bailey, J.; Ma, X.; and Gu, Q. 2020. Improving Adversarial Robustness Requires Revisiting Misclassified Examples. In *International Conference on Learning Representations*.

Wightman, L. F. 1998. LSAC National Longitudinal Bar Passage Study. Technical report, Law School Admission Council, Newtown, PA.

Wong, E.; Rice, L.; and Kolter, J. Z. 2020. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*.

Xu, W.; Evans, D.; and Qi, Y. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*.

Yang, Y.; Zhang, G.; Katabi, D.; and Xu, Z. 2019. ME-Net: Towards Effective Adversarial Robustness with Matrix Estimation. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Yu, H.; Xu, W.; and Zhang, H. 2022. Towards safe reinforcement learning with a safety editor policy. *Advances in Neural Information Processing Systems*, 35: 2608–2621.

Zhang, D.; Zhang, T.; Lu, Y.; Zhu, Z.; and Dong, B. 2019. You only propagate once: Accelerating adversarial training via maximal principle. *Advances in neural information processing systems*, 32.

Zhang, H.; Chen, H.; Xiao, C.; Gowal, S.; Stanforth, R.; Li, B.; Boning, D. S.; and Hsieh, C. 2020a. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018. Efficient neural network robustness certification with general activation functions. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, 4944 – 4953. Curran Associates Inc.

Zhang, J.; Xu, X.; Han, B.; Niu, G.; Cui, L.; Sugiyama, M.; and Kankanhalli, M. 2020b. Attacks which do not kill training make adversarial learning stronger. In *International conference on machine learning*, 11278–11287.