ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 38

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

SMILE: SAFE MACHINE LEARNING VIA EMBEDDED OVERAPPROXIMATION

Presentata da: Matteo Francobaldi

Coordinatore Dottorato

Paola Salomoni

Supervisore

Michele Lombardi

Co-supervisore

Paolo Torroni

Esame finale anno 2026

Acknowledgments

To my supervisor Michele Lombardi, for his invaluable guidance, for everything he has taught me over these years, and for the doors he has opened for me, most notably the research visit in New York, where I lived one of the happiest and most enriching chapters of my life. Thank you, Michi!

To all members of my research group in Bologna, for offering me a supportive and exciting environment, and for making me feel at home from the very beginning.

To Andrea Lodi, for welcoming me into his group at Cornell, and for the inspiring discussions that greatly contributed to my professional growth.

To the TUPLES project, for supporting this work and for offering me the opportunity to take part in the many stimulating meetings across Europe.

To my community of beautiful souls spread across the world, from Capistrello to Bologna, and onward to Berlin, New York, San Francisco, and Paris, for the joy, warmth, and energy they have brought into my life.

Last but certainly not least, to my parents, for their unconditional love and affection, and for supporting and trusting me in all my choices. Everything I am, and everything I hope to become, is rooted in you!

Abstract

The 21st century has witnessed the resurgence of Artificial Intelligence, mostly driven by a shift from symbolic methods, dominant in the previous decades, to sub-symbolic techniques, fueled by the rapid growth of data availability and computational power. Neural Networks, in particular, have achieved unprecedented results in a variety of domains, such as computer vision and natural language processing.

However, beyond the general excitement, the widespread adoption of AI has also raised concerns on its trustworthiness and reliability, as reflected in the emerging landscape of legal frameworks regulating this technology, pioneered by the EU AI Act. In safety-critical or socially sensitive scenarios, these regulations demand the satisfaction of specific properties, like robustness or fairness, to ensure AI alignment with human values and expectations. Even in non-critical settings, the satisfaction of formal requirements, like monotonicity or physical laws, may be desired to align these systems with the operational domain, hence to foster interpretability and usability.

Unfortunately, while highly effective in terms of accuracy, purely data-driven AI remains inherently unable to formally guarantee the satisfaction of additional properties, due to its dependence on data, often scarce and noisy, and to the heuristic nature of the most common training algorithms. This has led, in recent years, to the development of methodologies to train property-aware systems, which however remain limited to specific properties and architectures, given the complexity of the problem.

In this thesis, we take a step in this direction by introducing SMiLE (Safe Machine Learning via Embedded Overapproximation), a novel framework to enforce generic properties into arbitrary neural models, built upon the integration of sub-symbolic learning with symbolic reasoning. Across a wide spectrum of properties and learning tasks, we demonstrate the ability of SMiLE to match property-specific baselines in terms of accuracy, while providing advantages in terms of generality and guarantees.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivations

Data-driven approaches, at the dawn of the 21st century, sparked a renaissance in Artificial Intelligence (AI), following the stagnation and skepticism of the previous two decades. Neural Networks (NNs), in particular, revitalized in the early 2010s by the surge in data availability and computational power, have achieved unprecedented success across a diverse range of domains, marking the beginning of what is widely regarded as a new spring for the discipline.

Convolutional Neural Networks [LBBH98a, Kri23] have enabled breakthroughs in computer vision, while Recurrent Neural Networks [Elm90, MSO24] have driven advances in natural language processing. The more recent Transformers [VSP+17] have reshaped the AI landscape: GPT-3 [BMR+20, Kal24], with its unprecedented capabilities in generating coherent and persuasive text, has revolutionized human-computer interaction, while Vision Transformers [DBK+21, PHC+25] and Conformers [GQC+20] have pushed the state of the art in image classification and speech recognition, respectively. In game AI, progress in reinforcement learning has enabled systems such as AlphaGo [SHM+16] and OpenAI Five [BBC+19] to master complex tasks and environments. Finally, learning approaches have also had a profound impact on science and engineering, from drug discovery [BA25] to physics [MGC+25], from energy management [SDAM24] to manufacturing [GKM+24].

With growing AI capabilities, however, come increased risks, which motivate the intensifying global effort in regulating this technology, with the EU AI Act [Eur24] pioneering this direction. In safety-critical or ethically sensitive scenarios, in particular, these regulations demand the satisfaction of specific properties, such as robustness in autonomous driving (e.g., resisting adversarial attacks in traffic

sign recognition) or fairness in automated hiring (e.g., avoiding racial bias in candidate selection), necessary to foster *AI alignment* with human values and societal expectations. Even non-critical settings may benefit from the satisfaction of formal properties, such as monotonicity in remaining useful life estimation (e.g., the health condition of a machine can only deteriorate over time), or physical laws in scientific modeling (e.g., conserving mass or energy in fluid simulations), desired to align these systems with the operational domain.

Unfortunately, purely data-driven AI is designed to solely optimize prediction quality. Moreover, it heavily relies on data, often scarce and noisy, as well as on heuristic algorithms, unable to ensure convergence to optimal solutions. This is why these methods, beyond statistical accuracy, are inadequate to provide formal guarantees on additional properties. Believing that AI can keep evolving through data alone is, in our opinion, a flawed perspective. Data is unfortunately not enough! There is an urgent need to develop hybrid methodologies that, by combining subsymbolic learning with symbolic reasoning, are able to enforce user-defined properties into data-driven systems.

These considerations apply to a wide range of model classes, including more traditional non-neural approaches, such as kernel-based [BGV92, LB08] and tree-based [Bre01, NTF22] methods, which remain widely used in practice. The focus of this thesis, however, is on NNs.

Enforcing properties in such systems is a challenging task. In fact, even verifying the validity of properties in NNs has been shown to be NP-hard [KBD+17], while enforcing them is even harder [ML23]. Some approaches act before training, such as [QRLB20] and [GKR24]: the former promotes invariance through data augmentation, the latter encodes equivariance into the model architecture. Others operate during training, such as [SFMVdB20] and [MA25]: the former encourages monotonicity through regularization, the latter enforces safety via output projection. Finally, a third line of approaches intervene after training, such as [MSF23] and [RK22]: the former ensures fairness via output purification, the latter restores robustness through model correction.

Despite the substantial progress achieved over the past decade, the current landscape of methods still exhibits significant gaps, most notably in terms of *generality* and *guarantees*. Many approaches are tailored to specific properties, such as local robustness against adversarial perturbations, monotonicity constraints, or fairness criteria, making them difficult to adapt to other requirements. Others depend on restrictive architectural assumptions, such as small ReLU feedforward

networks, limiting their applicability to modern, large-scale models. Finally, most of the existing methods are only designed to encourage property feasibility, without providing formal satisfaction guarantees, which is particularly problematic in safety-critical or ethically sensitive applications, where even rare violations can have severe consequences.

In this thesis, we contribute to close these gaps by introducing *SMiLE* (*Safe Machine Learning via Embedded Overapproximation*), a framework to enforce generic properties into arbitrary neural models, able to provide full satisfaction guarantees, while remaining competitive with property-specific baselines.

1.2 Contributions

This thesis lies at the intersection between Machine Learning (ML) and Combinatorial Optimization (CO), where the latter, thanks to its inherent ability to deal with constraints, serves as a powerful technology to inject properties into the former, representing our ultimate ambition. Specifically, to this aim, we provide the following contributions.

Taxonomies. We review real-world enforcement scenarios that motivate our effort, and we classify them by the nature of the desired property – *domain alignment* and *human alignment*. We then formalize properties as implications between a set of inputs and their corresponding outputs, and we categorize them by two mathematical criteria: the size of this set – *trace* if it contains at most one element, *relational* otherwise – and their dependence on data – *local* if defined around specific datapoints, *global* otherwise. Finally, we review existing approaches for property verification and enforcement, the former classified by the adopted algorithmic paradigm – *optimization and searching*, *reachability analysis* and *heuristic attacks* – the latter by the stage of the ML pipeline in which they operate – *preprocessing*, *inprocessing* and *postprocessing*.

Methodology. We propose a novel property-enforcement framework, SMiLE, representing the main contribution of this thesis, and consisting of two core components: a particular *neural architecture* and a dedicated *property-aware training*. The SMiLE architecture augments a standard neural network with an overapproximation mechanism, designed to facilitate both property verification and enforcement. The SMiLE training augments a standard gradient-based approach with a CO-based enforcement mechanism, and is provided in two versions, specialized on trace and relational properties, respectively

Applications. We present an extensive computational study, which demonstrates the competitiveness and versatility of SMiLE across a diverse range of both trace and relational use cases. For the former, we consider *safety* in function approximation, *stability* in multi-step time series forecasting, and *exclusiveness* in multi-label classification. For the latter, we adopt *monotonicity* in function approximation, *robustness* in digit recognition, and *fairness* in recidivism prediction, student admission, and crime estimation.

1.3 Outline

This thesis is structured as follows. Chapter 2 introduces the foundational notions and terminology underpinning this work. Chapter 3 formalizes the problem, motivates our study, and reviews the relevant literature, while proposing taxonomies for scenarios, properties, verifiers and enforcers. Chapter 4 presents our property-enforcement framework. Chapters 5 and 6 demonstrate its application to trace and relational properties, respectively. Finally, Chapter 7 concludes the thesis by summarizing the main contributions, and outlining a possible roadmap for future research directions.

Chapter 2

Background

In this chapter, we introduce the background notions and terminology of Machine Learning (ML) and Combinatorial Optimization (CO), which constitute the foundation of this work. The presented framework employs CO techniques to enforce formal properties in ML systems, thereby positioning this thesis within the cross-fertilization of these two fields. We begin by formalizing the supervised learning problem, the paradigm for which SMiLE is currently designed. We then introduce neural networks, the primary focus of our study, together with the gradient descent algorithm, which underpins the property-aware training we propose. Finally, we review the fundamental concepts of CO, as well as the practice of optimizing over data-based systems, which we extensively adopt to design our property-enforcement mechanism.

2.1 Machine Learning

Machine Learning is the study of algorithms that use experience, in the form of data, to attain and improve the ability to perform certain tasks automatically, that is, without being explicitly programmed. Depending on the learning paradigm and the form of data used, machine learning is traditionally divided into three broad categories: *Supervised Learning*, *Unsupervised Learning* and *Reinforcement Learning*.

Even though our methodology can be extended to unsupervised and, in particular, reinforcement learning tasks, at the moment it is only designed for the supervised learning paradigm, which we formalize in the following section.

2.1.1 Supervised Learning

Supervised learning is the problem of inferring the relation between two variables from a sample set of observations representing it. Formally, the supervised learning framework consists of the following components.

- An input space $\mathcal{X}$, whose elements are called *instances*. Each instance in $\mathcal{X}$ is represented by a vector $x = (x_1, \ldots, x_n)$, whose entries are said *features*.
- An output space $\mathcal{Y}$, whose elements are called *labels*. The space $\mathcal{Y}$ can be either a finite set of *classes* or a continuous set: in the former case the problem is called *classification*, in the latter it is referred to as *regression*.
- An unknown relationship between the two spaces, that we model via a *joint probability distribution* $\mathcal{P} = \mathcal{P}(x, y)$ over the space $\mathcal{X} \times \mathcal{Y}$. Precisely, $\mathcal{P}$ can be seen as composed of two parts: a *marginal distribution* $\mathcal{P}_x$ over $\mathcal{X}$, which determines how likely it is to encounter any input x, and a *conditional distribution* $\mathcal{P}((x, y)|x)$ over $\mathcal{Y}$, which specifies the probability that the input x is labeled with the output y.
- A *dataset* $\mathcal{D} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{i=N} \subseteq \mathcal{X} \times \mathcal{Y}$, providing an empirical description of the input-output relation. The pairs in $\mathcal{D}$ are sampled from the distribution $\mathcal{P}$, that is, $\left(x^{(i)}, y^{(i)}\right) \sim \mathcal{P}$ for all $i \in [N]$.
- A *model space* $\Omega = \{f : \mathcal{X} \longrightarrow \mathcal{Y}\}$, defining the form of the model assumed to estimate the relation between $\mathcal{X}$ and $\mathcal{Y}$. This space, especially in statistical learning literature, is also known as *hypothesis space*.
- A sample-level *loss function* $L : \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}_+$, used as a measure of success for the learning algorithm, that is, to quantify how well the model predicts the target for a given instance: for $(x, y) \sim \mathcal{P}$, $L(f(x), y)$ measures how "far" the estimation $f(x)$ is from the correct label y.

The goal of a supervised learning algorithm is to approximate the relation between $\mathcal{X}$ and $\mathcal{Y}$, that is, to produce a model f such that $f(x) \approx y$ for each pair $(x, y) \sim \mathcal{P}$. Precisely, this goal is formalized as the following unconstrained optimization problem:

$$\arg\min_{f \in \Omega} L_{\mathcal{P}}(f) = \arg\min_{f \in \Omega} \mathbb{E}_{(x,y)\sim\mathcal{P}}[L(f(x), y)], \tag{2.1}$$

where the distribution-based model-level loss $L_{\mathcal{P}}(f)$ is called *expected error*.

This goal, however, is not achievable directly, since the distribution $\mathcal{P}$ is unknown and inaccessible. Therefore, it is attained empirically by using the available information $\mathcal{D}$, that is, by solving the following problem:

$$\arg\min_{f\in\Omega} L_{\mathcal{D}}(f) = \arg\min_{f\in\Omega} \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L\left(f(x), y\right). \tag{2.2}$$

where the data-based model-level loss $L_{\mathcal{D}}(f)$ is said *empirical error*, and the process of solving (2.2) is called *learning*.

The formulation provided above highlights the central challenge of any ML task: *generalization*. Rather than performing perfectly on the observed instances, the aim of ML is to perform well on new, previously unseen ones. In other words, by observing past data, we try to make predictions about future data. For this reason, we also refer to a learning model f as a *predictor*, and to the estimated value $f(x)$ as the *prediction* of f for $x \in \mathcal{X}$. In practical development, a standard ML pipeline involves the following steps: collect a dataset $\mathcal{D}$ representative of the learning task, define a model space Ω, decide a proper loss function L, finally design a procedure to solve the corresponding optimization problem. In particular, the dataset $\mathcal{D}$ is split into two disjoint subsets: one is used for *training* the model, the other for *testing* its generalization abilities. Accordingly, the two subsets are called *train set* and *test set*, and denoted as $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$, respectively. Moreover, the error of the model on $\mathcal{D}_{train}$ is called *train error*, the one on $\mathcal{D}_{test}$ is said *test error*. For a more extensive discussion on the general learning theory, we refer the reader to the popular textbook "Understanding Machine Learning: From Theory to Algorithms" [SSBD14].

The method presented in this thesis is designed for a specific class of ML models, which we introduce in the next section.

2.1.2 Neural Networks

An *Artificial Neural Network*, or briefly *Neural Network* (NN), is a computational model loosely inspired by the human brain. The study of these models, and their use in solving learning problems, delineates an entire subfield of machine learning, called *Deep Learning*. The starting point of this research area dates back to 1943, when McCulloch and Pitts published the first mathematical formulation of a biological neuron [MP43]. The first learnable artificial neuron, called *Perceptron*, was instead proposed in 1958 by Rosenblatt [Ros58], and consisted of a binary classifier capable of distinguishing between two categories through a linear separating boundary.

The modern artificial neuron generalizes Rosenblatt's idea, and represents the basic unit of state-of-the-art networks. Formally, a neuron is a vector-to-scalar function $\phi\colon \mathbb{R}^n \to \mathbb{R}$, that combines a linear transformation, parameterized by *weights* $w \in \mathbb{R}^n$ and *bias* $b \in \mathbb{R}$, with a (typically) non-linear *activation function* σ:

$$y = \sigma(w \cdot x + b) \tag{2.3}$$

The parameters $\theta = (w, b)$ are usually optimized through a learning process, while the activation σ represents a hyperparameter, responsible for injecting non-linearity into the model [Dat20]. Figure 2.1 provides a visual representation of a neuron, while Figure 2.2 depicts some of the most commonly used activations.

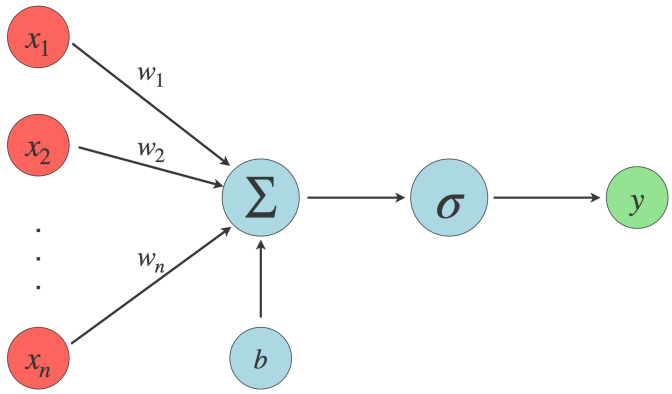


Figure 2.1: An artificial neuron.

A *neural network* can then be obtained by stacking multiple neurons into a sequence of layers, resulting in a vector-to-vector function $f\colon \mathcal{X} \to \mathcal{Y}$, typically represented as a computational graph. Formally:

$$x^{(d)} = f(x;\theta) = f_d(x^{(d-1)};\theta_d) \circ \cdots \circ f_2(x^{(1)};\theta_2) \circ f_1(x^{(0)};\theta_1) \tag{2.4}$$

Here, the input vector $x^{(0)}$ is propagated through the graph into intermediate *embeddings* (or latent vectors) $x^{(i)}$, for $i = 1, \ldots, d-1$, to finally produce the output $x^{(d)}$. In particular, f_1 is called the *input layer*, f_d is said the *output layer*, while the internal layers are referred to as *hidden layers*. The number of layers in the network, called *depth*, the size of each layer, referred to as the layer's *width*, and the connectivity among neurons, that is, the way in which information flows through the graph, define the *architecture* of the network (in other words, the topology of the computational graph). The connectivity pattern, in particular, is the key distinguishing element among the different classes of neural networks. In

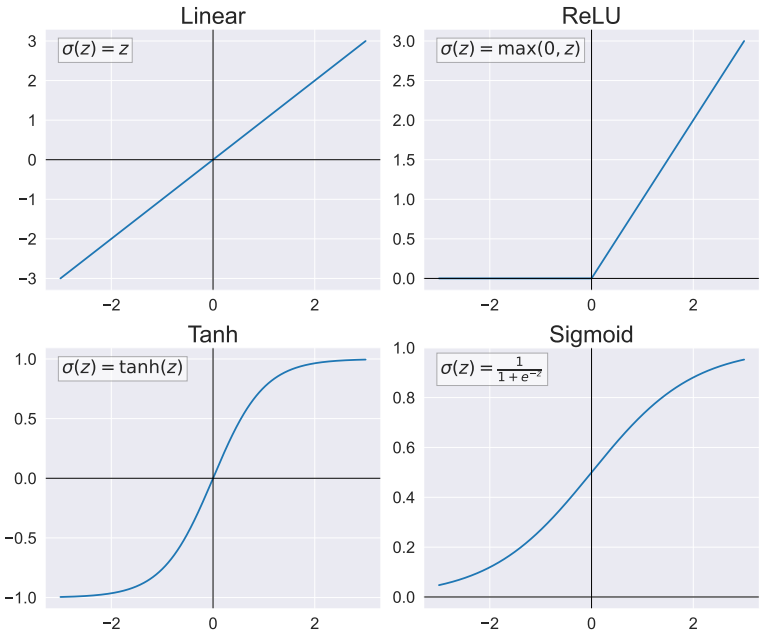


Figure 2.2: Popular activation functions.

what follows, we briefly introduce the most common types of architectures, which are also the ones that we adopt in our computational study.

Feedforward Neural Networks (FNNs) Feedforward Neural Networks, also known as Multi-Layer Perceptrons (MLPs) and representing the simplest and most classical form of networks, process information unidirectionally from the input to the output, without cycles or feedback, through densely connected layers, that is, each neuron in each layer is connected to all neurons from the previous layer. Formally, the l-th dense layer transforms $x^{(l-1)}$ into $x^{(l)}$ as follows:

$$x^{(l)} = \sigma(W^{(l)}x^{(l-1)} + b^{(l)}) \tag{2.5}$$

where $\theta_l = (W^{(l)}, b^{(l)})$ represents the layer parameters, while σ is a component-wise activation function. The width of the layer n_l, in this case, is usually meant as the number of neurons (i.e., the number of rows of the matrix $W^{(l)}$). Figure 2.3 depicts an example of an FNN.

Convolutional Neural Networks (CNNs) Convolutional Neural Networks, like FNNs, process information forward, but unlike FNNs, they include at least one convolutional layer, where each neuron is only connected to a small local region of the previous layer (*sparse connectivity*), and where subsets of neurons

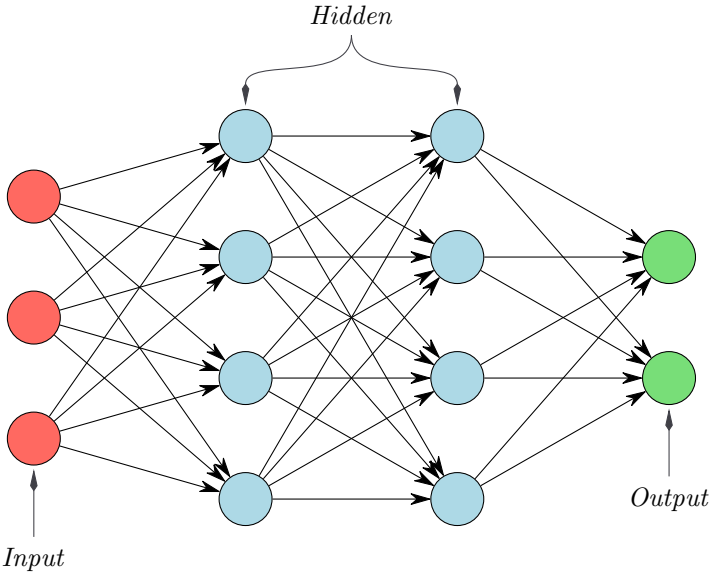


Figure 2.3: An FNN with 2 hidden layers.

within the layer share the same parameters (*parameter sharing*). In particular, a convolutional layer works by sliding a *filter*, or *kernel*, over the input, computing a weighted sum of the input values overlapped at each location. This operation is called *convolution*, and can be defined, for a 2-dimensional input $x \in \mathbb{R}^{h \times w \times c}$ with multiple channels, and for a filter $W \in \mathbb{R}^{h' \times w' \times c}$ with $h' \leqslant h$ and $w' \leqslant w$, as:

$$(W * x)_{i,j} = \sum_{p=1}^{h'} \sum_{q=1}^{w'} \sum_{k=1}^{c} W_{p,q,k}\, x_{i+p-1,\, j+q-1,\, k} \tag{2.6}$$

for $1 \leqslant i \leqslant h - h' + 1$ and $1 \leqslant j \leqslant w - w' + 1$. Then, the l-th convolutional layer transforms $x^{(l-1)}$ into $x^{(l)}$ as follows:

$$x^{(l)}_{k,i,j} = \sigma\Big((W^{(l)}_k * x^{(l-1)})_{i,j} + b^{(l)}_k\Big), \quad k = 1, \ldots, n_l, \tag{2.7}$$

where $\theta_l = (W^{(l)}_k, b^{(l)}_k)_{k=1}^{k=n_l}$ represents the layer-specific parameters, while σ is a component-wise activation function. For this layer, the width is commonly considered as the number of filters n_l. Figure 2.4 provides a visual intuition of a convolutional layer. The convolutional structure of these networks enables the extraction of local and hierarchical features (edges, shapes, objects), making them capable of capturing spatial dependencies. This is why CNNs work particularly well on spatial data, hence for tasks such as image recognition and computer vision. Note that, in general, convolutional layers can also include additional hyperparameters such as *padding* and *stride*, which control the spatial size of the output and

the movement of the filter over the input. Moreover, they are typically followed by *pooling* layers, which replace the output at a certain location with a summary statistics (e.g., max or mean) computed over its neighborhood, in order to improve the robustness and computational efficiency of the transformation. For a more detailed discussion of CNNs, we refer to [GBC16, LBBH98a].

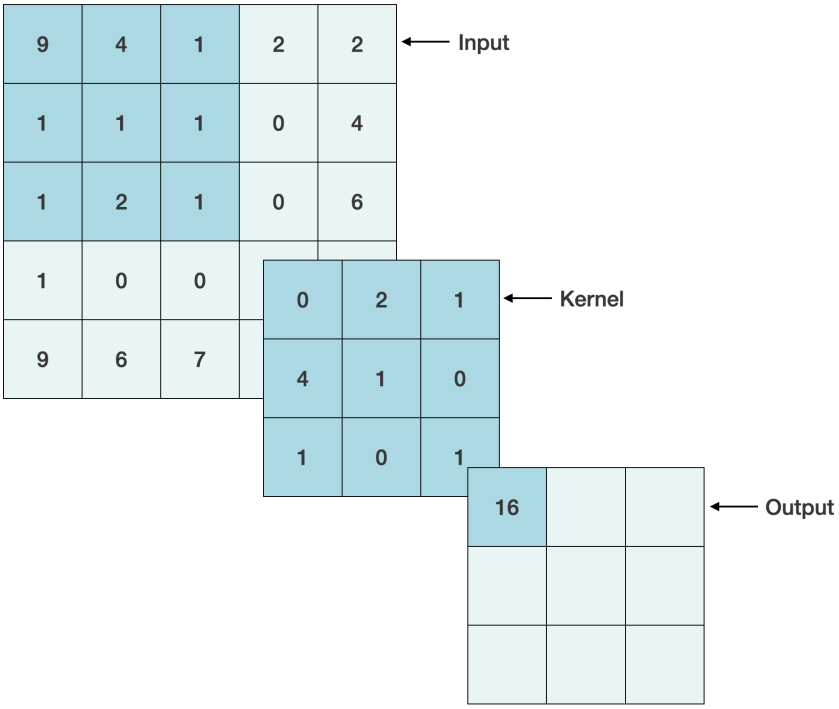


Figure 2.4: A convolution applied to a 1-channel input.

Recurrent Neural Networks (RNNs) Unlike FNNs and CNNs, Recurrent Neural Networks include *cycles* in a least one of their layers, so that the hidden state of this layer, at a given time step, is fed back to the same layer at the next step, providing the network with a form of memory. Similar to CNNs, the weights are shared across time steps, enabling the same transformation to be applied recurrently. Formally, at each time step t, the l-th recurrent layer transforms the output from the previous layer $x_t^{(l-1)}$, and its hidden state from the previous step $h_{t-1}^{(l)}$, as follows

$$h_t^{(l)} = \sigma_h\left(W_{xh}^{(l)} x_t^{(l-1)} + W_{hh}^{(l)} h_{t-1}^{(l)} + b_h^{(l)}\right) \tag{2.8}$$

$$x_t^{(l)} = \sigma_y\left(W_{hy}^{(l)} h_t^{(l)} + b_y^{(l)}\right) \tag{2.9}$$

where $\theta_l = (W_{xh}^{(l)}, W_{hh}^{(l)}, W_{hy}^{(l)}, b_h^{(l)}, b_y^{(l)})$ represents the layer-specific parameters, while σ_h and σ_y are component-wise activation functions. For this layer, the width n_l usually represents the dimension of the hidden state (i.e., the memory capacity). Figure 2.5 depicts an *unfolded* recurrent layer, highlighting the sequential nature of the computation, and the parameter sharing across the different steps. The recurrent structure of these networks enables information to persist across time, making them capable of capturing temporal dependencies. This is why RNNs work particularly well on sequential data, hence for tasks such as natural language processing and time series forecasting. Several specialized variants of recurrent layers have been proposed to address limitations of the basic formulation, such as vanishing and exploding gradients. Notable examples include the Long Short-Term Memory (LSTM) [HS97] and the Gated Recurrent Unit (GRU) [CvMBB14], which introduce gating mechanisms to better capture long-range dependencies. A detailed discussion of RNNs can be found in [GBC16].

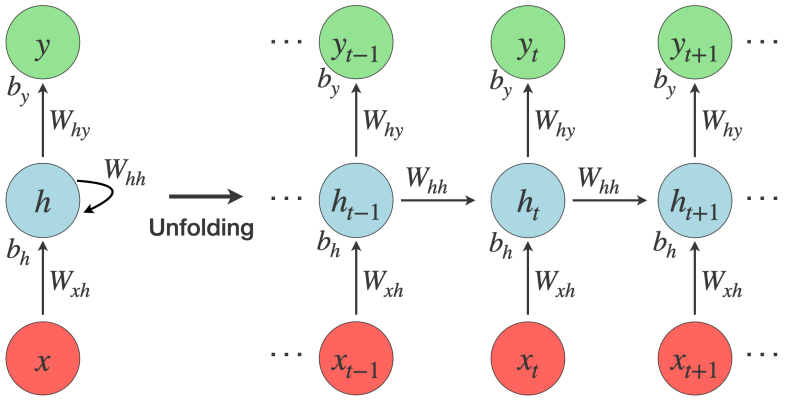


Figure 2.5: An unfolded recurrent layer.

Beyond the network types discussed above, other two notable architectures have emerged in recent years: *Graph Neural Networks* and *Transformers*, for which, however, we only provide an intuitive description, given that they are not directly adopted in this thesis.

Transformers First proposed in the seminal paper "Attention is All You Need" [VSP+17], Transformers have quickly gained tremendous popularity, and today they dominate the state of the art across various domains, particularly language

modeling, translation, and sequence prediction, as they are primarily designed to deal with sequential data. At the core of Transformers is the self-attention mechanism, which enables the model to process an input sequence as a whole, rather than token by token as in standard RNNs. In each token representation, the model encodes both the information of the token itself and the contextual influence of all other tokens in the sequence. This allows the model to capture long-range dependencies efficiently, and represents the foundation for the success of Transformers in tasks such as language modeling, translation, and sequence prediction. The seminal paper by [VSP+17] provides a detailed description of the Transformer architecture.

Graph Neural Networks Graph Neural Networks (GNNs) are neural architectures designed to operate on graph-structured data, where each input is a set of nodes connected by edges. Each node is represented by a feature vector, which is updated by iteratively aggregating information from neighboring nodes. This allows GNNs to capture both local and global graph structure, making them effective for tasks such as node classification, link prediction, and graph-level regression, in domains ranging from social networks to molecular modeling. For an extensive description of GNNs, we refer to [SGT+09].

We conclude this brief overview of neural networks by emphasizing that, while most existing property-enforcement approaches rely on restrictive assumptions on the chosen neural architecture or activation function, our method is largely independent of both. Specifically, the proposed framework is fully compatible with any activation function, and with any neural architecture that works with fixed input dimension. This includes FNNs, CNNs and RNNs, but excludes Transformers and GNNs. Extending the framework to these currently unsupported architectures represents a promising direction for future research.

In the next section, we describe the standard algorithm to train neural models, which in this thesis we complement with an enforcement mechanism, designed to make the process property-aware.

2.1.3 Gradient Descent

Training a neural network is, conceptually, not different from training other types of machine learning models, that is, we need to apply the learning paradigm described in Section 2.1.1: collect the data, define a model space, choose a loss function, solve the resulting optimization problem. However, differently from simpler systems, such as *linear models*, in deep learning algorithms the error function

is usually non-convex, due to the nonlinearity of the network, and the parameter space is extremely high-dimensional, due to its typically large scale. For these reasons, NNs are trained via heuristic methods that search for a good, yet suboptimal, solution.

One of the most popular class of optimizers for neural networks is represented by the family of *Gradient Descent* (GD) algorithms. Algorithm 1 provides the pseudocode of the general GD version: for each epoch, the algorithm propagates a mini-batch of input instances through the network to obtain the corresponding loss value; it then computes the gradient of this value with respect to the network parameters; it finally updates these parameters in the opposite direction of this gradient to reduce the error. In particular, the backward pass (line 5) is performed via *backpropagation*, an algorithm specifically designed to efficiently differentiate over neural architectures [RHW86].

Algorithm 1 SGD

1: **Input:** initial parameters θ, training data $\mathcal{D}_{\text{train}}$, loss L, batch size b, epochs e, learning rate η
2: **for** epoch from 1 to e **do**
3: **for** mini-batch $\mathcal{B} \subseteq \mathcal{D}_{\text{train}}$ of size b **do**
4: **Forward pass:** $L_{\mathcal{B}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{(x,y)\in\mathcal{B}} L(f(x;\theta), y)$
5: **Backward pass:** $g \leftarrow \nabla_\theta L_{\mathcal{B}}$
6: **Update:** $\theta \leftarrow \theta - \eta\, g$
7: **end for**
8: **end for**
9: **return** θ

Note that, according to the batch size, different terminologies are commonly adopted for Algorithm 1. Classical GD corresponds to the case in which $b = |\mathcal{D}|$, where each update is computed from the exact full gradient. We instead call the algorithm *Stochastic Gradient Descent* (SGD) when $b = 1$, where the parameters are updated after evaluating the gradient on a single data point, which speeds up the computations, but also introduces significant stochasticity. In practical applications, purely full-batch or single-sample updates are rarely used. Instead, mini-batch gradient descent, with $1 < b < |D|$, still referred to as SGD, is employed to strike a balance between computational efficiency and gradient accuracy. Moreover, beyond these basic schemes, several advanced optimizers have been proposed to improve convergence and robustness, such as Adam [KB14], which combines

momentum and adaptive learning techniques, and represents the default choice for most applications. A clear overview of the different types of gradient-based algorithms is provided by [Rud17].

As a main contribution of this thesis, we extend the GD algorithm with a mechanism that accounts for the property to satisfy. To design such mechanism, in particular, we rely on a class of mathematical programming techniques, which we introduce in the next section.

2.2 Combinatorial Optimization

Combinatorial Optimization (CO) is a cornerstone of Operations Research, Analytics and Computer Science, where it is used as a versatile decision-making tool for a large variety of applications, ranging from renewable energies [KRBA16] to cancer detection [LZMW09], from logistics [BS07] to project scheduling [BHL+10], among many others.

A CO problem is the problem of optimizing a real-valued function c over a discrete or semi-discrete set of points $\mathcal{F}$, and can be formulated as follows:

$$\begin{array}{ll} \min & c(x) \\ \text{s.t.} & x \in \mathcal{F}. \end{array} \tag{2.10}$$

where the function c is called the *objective* of the problem, while the set $\mathcal{F}$ is referred to as the *feasible space*, and its points are called *feasible solutions*. Such a solution x^* is said *optimal* (or an *optimum*) if $c(x^*) = \min_{x \in \mathcal{F}} c(x)$, where $*$ in the superscript is used to denote optimality. The particular form taken by the objective and the feasible space defines different classes of problems, and diversifies CO into many subfields, such as Mixed-Integer Programming (MIP) or Constraint Programming (CP).

From a complexity theory perspective, most of these problems fall into the *NP-Hard* computational class [Kar72], due to their discrete and non-convex nature, which makes a brute-force search for an optimal solution intractable in practice. Despite their complexity, however, today many of these problems can be realistically solved, thanks to decades of effort spent by the CO community in designing ever more sophisticated techniques, such as *Branch and Bound* (B&B), *Cutting Planes* and *Primal Heuristics*. Briefly, B&B [LD60] represents the backbone of modern solvers, and consists of a tree-search algorithm that explores the solution space of an optimization problem, while pruning nodes that can not contain optimal solutions, as identified through bounds computed on the objective

function. Cutting planes [Gom58, MMWW02] strengthen linear relaxations by iteratively adding valid inequalities (*cuts*), which eliminate infeasible fractional solutions without excluding any feasible integer ones. Finally, primal heuristics [Ber06, Ber14] are approximate methods that quickly generate feasible, though potentially suboptimal, solutions, useful to improve bounds and to prune the tree, hence to accelerate the optimization process.

The profound impact of CO on a wide range of domains has raised the commercial interest around the field. Today, many competitive CO software products are released both for commercial purposes, such as Gurobi [Gur] or Xpress [Xpr], and for academic uses, like SCIP [SCI] or Google OR-Tools [Goo].

Combinatorial optimization plays a crucial role in this thesis. As formalized in the next chapter, indeed, properties can be interpreted as requirements, and requirements naturally translate into constraints. Unlike standard ML, which primarily tackles unconstrained problems (Equation (2.2)) through approximate, heuristic procedures (Algorithm 1), CO is inherently designed to address constrained optimization in a complete, exact fashion, which makes it particularly well-suited to the objective of this thesis: guaranteeing property satisfaction in data-based systems. In particular, we build upon a practice that has recently emerged in the literature, described in the next section.

2.2.1 Optimizing over Machine Learning Models

Optimizing over a ML model essentially means turning its input and output variables, or its weights, into the decision variables of an optimization process. This practice consists in encoding a fully-fledged ML model into the language of a chosen CO technology. Examples can be found in [FJ18, AHM+20, TKTM21], which provide MIP formulations of neural networks and decision trees, or in [BLM15, LG16], which instead propose CP embeddings of the same ML models.

Multiple reasons motivate this class of methodologies, such as using ML to approximate the missing components of a partially specified CO problem [LM18], compressing NNs to enable their deployment under limited computational resources [SKR20], or bounding and counting their linear regions to investigate their complexity [STR18].

Optimizing over neural networks is, however, a highly challenging task. The large size of these models exponentially increases the dimension of the hosting CO formulation and, more critically, their strong non-linearity exacerbates its discreteness and non-convexity (non-linearity is typically encoded linearly through

discrete variables). Despite the availability of open-source libraries offering well-designed approaches for encoding neural models within CO frameworks, like OMLT [CJH+22], ENTMOOT [TKM+21], and EML [LMB17], the optimization process becomes intractable once these models reach the level of complexity necessary in real-world applications.

In this thesis, we are primarily interested in the use of optimization techniques over neural networks to rigorously verify and enforce the satisfaction of properties. In the methodological sections, in particular, we show how we mitigate the complexity of this task through a specifically designed neural architecture.

Chapter 3

Data is not Enough!

In this chapter, we present the motivations underlying this thesis, formalize the addressed problem, and review existing competitors. We begin by proposing a taxonomy of real-world scenarios in which incorporating external requirements into data-driven models is desirable or even mandatory. We then define formal properties and classify them from a mathematical standpoint. Subsequently, we formulate the property verification problem, discuss its complexity, and introduce a taxonomy of existing verifiers. Finally, we move the discussion to enforcement methods: analogously to verification, we formulate the enforcement problem, analyze its complexity, and propose a taxonomy of existing enforcers, which constitute our direct competitors. We emphasize, however, that the proposed taxonomies are intended solely to support the arguments of this thesis and should not be regarded as comprehensive surveys of the literature.

3.1 A Taxonomy of Scenarios

Data-driven approaches, at the beginning of the 21st century, sparked a renaissance in AI, after the disillusionment that had characterized the field during the previous two decades. Particularly neural networks, since the early 2010s, fueled by the exponential growth of data availability and computing power, have achieved unprecedented results across a wide variety of domains, initiating a new AI spring.

Convolutional NNs [LBBH98a, Kri23] have played a crucial role in computer vision, where architectures such as VGG [SZ15] and ResNet [HZRS16] have significantly outperformed traditional handcrafted approaches, enabling breakthroughs in domains such as medical imaging [LKB$^+$17] and autonomous driving [B$^+$16]. At the same time, recurrent NNs [Elm90, MSO24] and their gated variants, like LSTMs [HS97] and GRUs [CvMBB14], have powered advances in natural language

processing [YHPC18] and time series forecasting [QSC+17]. Building on these foundations, the introduction of the Transformer architecture [VSP+17] has reshaped the AI landscape, thanks to its superior performance and scalability. GPT-3 [BMR+20, Kal24], in particular, has marked a major milestone, demonstrating unprecedented capabilities in generating coherent, contextually relevant, and often persuasive text, revolutionizing human-computer interaction. More recently, the transformer-based Vision Transformers [DBK+21, PHC+25] and Conformers [GQC+20] have significantly advanced the state of the art in image classification and speech recognition, respectively. Finally, learning approaches have also had a profound impact on science and engineering, from drug discovery [BA25] to physics [MGC+25], from energy management [SDAM24] to manufacturing [GKM+24].

However, despite the general excitement surrounding it, data-driven AI exhibits, in many scenarios, systematic limitations. First of all, ML inherently requires high-quality data, and data, unfortunately, is not always available, as it may be costly to collect, restricted due to privacy concerns, or rare for certain phenomena. Second, the ever-increasing complexity of neural models underscores the need for their interpretability and explainability. Finally, there is a large variety of scenarios in which AI models are expected to satisfy constraints dictated by natural laws, or to comply with legal frameworks regulating their behavior.

When it comes to guaranteeing the satisfaction of formal properties, in particular, purely data-driven approaches prove to be completely inadequate. Indeed, even by assuming optimal conditions for the learning setup, namely, clean and abundant data, well representing both the task to perform and the property to satisfy, and even by assuming the optimal outcome of the learning process, that is, the gradient descent algorithm converges to a global optimum, the trained neural network would only be able to provide statistical guarantees on the input region covered by the data, while it would not offer any guarantee out of distribution. The problem degenerates quickly if we observe that the assumed conditions rarely hold: optimality in the gradient descent solution is unrealistic, given the high dimensionality of neural network parameter spaces, as well as the heuristic nature of the algorithm, while data, as already remarked, is usually noisy and scarce (especially with respect to the input dimensionality, like in computer vision tasks), and in many cases non-representative of the property, or even in conflict with it.

Beyond training, a purely data-based approach, from a property-satisfaction perspective, is also inappropriate for evaluation. Data-dependent metrics, indeed, only provide statistical information over a finite test set, and are therefore insufficient to evaluate the behavior of the model in rare but critical input regions. In

particular, sample-based predictive performance, representing the primary evaluation criterion in standard ML pipelines, is inadequate in the property-aware learning setting: a model may achieve high accuracy while still violating critical constraints. Consequently, property satisfaction must be regarded as a first-class evaluation criterion, and measured via suitable data-independent metrics.

Believing that AI can keep evolving through data alone is, in our opinion, a flawed perspective. Data is unfortunately not enough! Only by combining subsymbolic learning with symbolic knowledge and principles, we can ensure that AI is both effective and reliable, which is of paramount importance for its widespread adoption, particularly in high-stakes applications.

In what follows, we review and classify several scenarios where the data-based model is required to align with principles that cannot be captured from data alone. The nature of this alignment represents the main dimension of our taxonomy: we distinguish between *domain alignment*, where the model needs to conform to the laws and mechanics of the environment where it operates, and *human alignment*, where it must adhere to the values and norms of humans and societies.

3.1.1 Domain Alignment

This class of methods builds upon the paradigm of *Physics-Informed Machine Learning* (PIML), originally introduced to incorporate physical laws, typically expressed as partial differential equations, into data-driven systems [RPK19], and today adopted, beyond physics, to integrate knowledge from a wide range of domains. In some cases, the original PIML terminology is preserved, while in others it is adapted to the specific source of knowledge, or even to the specific principle embedded, hence resulting, for example, in *Medical-* [SBM24], *Chemistry-* [ZMN22], *Biology-* [WWZ+23], *Linguistics-* [SLX+24], *Symbolic-Reasoning-* [MZB+23], or *Gibbs–Duhem-*[RFLM23] *Informed ML*. In general, domain knowledge encompasses scientific laws and principles established through centuries of study in the natural sciences, insights and expertise accumulated by practitioners and professionals, as well as intuitions and commonsense derived from the everyday human interaction with the world.

In *physics*, examples include the use of Lagrangian mechanics for robot control [LRP19], stability properties for drone landing [SSO+19], or Navier–Stokes equations for fluid flow surrogate modeling [SGPW20]. In *geophysics* and environmental sciences, consistency between lake energy and heat fluxes has been incorporated into lake temperature monitoring [JWK+20], the Eikonal equation has been used to inform earthquake hazard assessment [bWAH+21], while flow and

transport constraints have been enforced into groundwater pollution emulation [MWK+23]. In *chemistry*, Fermi-Dirac statistics has been adopted in predicting the dissociation curves of the nitrogen molecule and hydrogen chain [PSMF20], monotonicity has been incorporated into oxygen solubility estimation [MIM+18], while rotational and translational equivariance have been embedded into molecular structure generation [TSK+18] and molecular dynamics simulation [ZHW+18]. In *materials science*, the acoustic wave equation has been employed for surface breaking crack quantification [SLB+20], geometrical symmetries for metal plate damage localization [ARI+25], while molecular dynamics for material viscosity estimation [CSKea24]. In *cosmology*, permutation invariance has been considered to estimate the red-shift of galaxies [ZKR+17], and the Schrödinger–Poisson equations to simulate gravitational collapse of fuzzy dark matter [MT25]. In *medicine*, the Pennes bioheat equation has been used for breast cancer monitoring [PRK23], while the Eikonal equation for cardiac activation mapping [SCYP+20]. In *epidemiology*, infectious disease forecasting has been informed with general epidemic dynamics [RCR+23], Susceptible-Infectious-Recovered (SIR) mechanics [MSR+24], or Covid-19 compartmental constraints [QMB+25]. In *vision*, the integration of rotational and translational invariance or equivariance has been proposed for multiple image classification tasks [CW16, DFK16, LYLC18, WGTB17] In *manufacturing*, finally, monotonicity has been used for Remaining Useful Life (RUL) estimation [AVS+22], while general industrial system dynamics for predicting the thermal power of industrial Air Handling Units (AHUs) [IRD+25]. Table 3.1 provides a concise overview of the scenarios discussed above, whereas [vRMB+23] offers a comprehensive survey of the field, which the authors designate under the unified terminology of *Informed Machine Learning*.

Regardless of the source of knowledge, domain alignment is primarily pursued to enhance the interpretability and usability of the model, by ensuring that it operates consistently with its environment, as well as to improve its predictive performance and robustness, especially when data is scarce or noisy. Finally, beyond these practical motivations, this line of research is also guided by a principle of parsimony: *if relevant knowledge is available, why re-learn it from scratch?*

3.1.2 Human Alignment

If domain alignment teaches AI to understand the environment in which it operates, human alignment makes it aware of the humans it interacts with, where "humans" here may refer to individuals, social groups, or humanity as a whole. Human-aligned properties include safety practices, ethical principles and social expectations, especially arising in safety-critical and ethically sensitive domains.

In *healthcare*, approaches have been proposed, for example, to increase adversarial robustness in brain tumor [YF25] and retinal Optical Coherence Tomography (OCT) [HYL+19] classification, as well as in lung and skin lesion segmentation [LPZ21], to enforce safety constraints in prostate cancer diagnosing [JVW+25], to promote group fairness, specifically equalized odds, in Substance Use Disorder (SUD) treatment success estimation [WY25] and in dermatology disease identification [SYY+23], or to guarantee privacy in pneumonia detection [JAK25]. In *recruitment*, different trustworthiness properties have been addressed in candidate ranking, such as equality of opportunity and demographic parity [GAK19], disparate treatment and disparate impact [SJ18], and explainability [HCR+22]. In *finance*, individual fairness has been considered for income estimation [BPW+22] and credit approval [KS23], adversarial robustness, safety, and privacy for fraud detection [FMK+21, ADAA25, ZHD+23], and explainability for credit assessment [dLMVW22]. In *justice*, methods have been designed to promote opportunity equality and demographic parity in predicting recidivism scores [WVP18], individual fairness in estimating the criminal potential of communities [LGW19], while explainability in legal case matching [YSX+22]. In *education*, different notions of fairness have been considered in predicting the academic success of students to decide their admissibility, such as equalized odds and demographic parity [KKKS25], or intersectional fairness [PDD+25], while explainability has been studied in the development of systems to provide students with career feedback [SRD+25]. In *automotive*, finally, examples include methods to improve adversarial robustness in traffic sign recognition [HH25] and traffic estimation and routing [SMLD18], and approaches to guarantee safety in lane keeping [PAU24] and emergency braking [SQL+24]. An overview of the mentioned human-alignment scenarios is provided in Table 3.2, while more extensive surveys can be found in [LWF+22] and [JQC+25].

Regardless of the type of enforced property, human alignment aims to make AI models more interpretable and trustworthy, by ensuring that their outputs are fair, safe, and reliable, even in rare or out-of-distribution situations, as well as compliant with the landscape of both existing and emerging regulations. Trustworthiness is desirable for societal acceptance, as it fosters confidence in the technology, and enables innovation and positive impact without fear or resistance. Compliance is instead mandatory in regulated scenarios, where AI models, before deployment, must undergo thorough assessment to ensure that they meet the required legal standards.

An autonomous driving system, for example, must adhere to ISO 26262 [ISO18], the international standard for functional safety of electrical and electronic systems in road vehicles, and especially to its extension, Safety Of The Intended Function-

ality (ISO/PAS 21448, SOTIF) [ISO19], recently proposed to address the emerging AI functionalities, that is, to regulate scenarios in which hazards may arise even when the system operates as intended, for example due to unexpected environmental changes. A diagnostic AI software, as another example, must comply with IEC 62304 [Int06], the international standard governing the software development lifecycle for medical devices, and if deployed in Europe, also with the EU Medical Device Regulation (MDR) [Eur17], an additional set of requirements applied to the European market.

Beyond sector-specific legal frameworks, recent years have witnessed an intensifying global effort in regulating AI more broadly, with particular emphasis on fostering trustworthiness in its design and ensuring responsibility in its use. The European Union has pioneered this direction by introducing the EU AI Act [Eur24], which categorizes AI systems by risk level, and prescribes corresponding compliance obligations for all AI actors operating within its jurisdiction. In the United States, even though the regulatory landscape is more fragmented, and a comprehensive AI legislation is still missing, several bills have been proposed to fill this gap, both at federal level, such as the Artificial Intelligence Initiative Act [Uni20], and especially at state level, where these bills have been, in some cases, signed into law, like in Colorado [Col24] and California [Cal24]. The US National Institute of Standards and Technology (NIST), moreover, formalized a set of guidelines and recommendations, known as the NIST AI Risk Management Framework [Nat23], to promote the incorporation of trustworthiness considerations into the design and evaluation of AI products, even though this framework is intended solely for voluntary use. Another example, again with a non-mandatory nature, is represented by the AI Promotion Act [Int25] issued by Japan, which establishes social principles for human-centered AI, emphasizing respect for transparency, societal well-being and human dignity. At the international level, finally, a step towards regulatory AI was taken by the Organization for Economic Cooperation and Development (OECD), which proposed an intergovernmental set of non-binding recommendations, called the OECD AI Principles [OEC19], to promote innovative, trustworthy AI that respects human rights and democratic values.

These efforts reflect a widely recognized imperative: with growing AI capabilities, come increased risks, which extend across industries and national borders, and which highlight the urgency of designing AI methodologies that are ethically aligned and demonstrably compliant.

In the following sections, we discuss two cross-cutting considerations arising in property enforcement scenarios: consistency with data and validity guarantees.

Domain	Property	Application Example	Reference
Physics	Lagrangian Mechanics	Robot Control	[LRP19]
	Stability (Lipschitz Bounds)	Drone Landing	[SSO+19]
	Navier–Stokes Equations	Fluid Flow Surrogate Modeling	[SGPW20]
Geophysics	Energy Conservation	Lake Temperature Monitoring	[JWK+20]
	Eikonal Equation	Earthquake Hazard Assessment	[bWAH+21]
	Flow/Transport Constraints	Groundwater Pollution Emulation	[MWK+23]
Chemistry	Fermi–Dirac Statistics	N_2-H-Dissociation Curves Prediction	[PSMF20]
	Monotonicity	Oxygen Solubility Estimation	[MIM+18]
	Symmetry	Molecular Structure Generation	[TSK+18]
	Symmetry	Molecular Dynamics Simulation	[ZHW+18]
Materials Science	Acoustic Wave Equation	Surface Crack Quantification	[SLB+20]
	Symmetry	Metal Plate Damage Localization	[ARI+25]
	Molecular Dynamics	Material Viscosity Estimation	[CSKea24]
Cosmology	Geometrical Symmetries	Galaxy Red-Shift Estimation	[ZKR+17]
	Schrödinger–Poisson Equations	Dark Matter Collapse Simulation	[MT25]
Medicine	Pennes Bioheat Equation	Breast Cancer Monitoring	[PRK23]
	Eikonal Equation	Cardiac Activation Mapping	[SCYP+20]
Epidemiology	Epidemic Dynamics	Infectious Disease Forecasting	[RCR+23]
	SIR Mechanics	Infectious Disease Forecasting	[MSR+24]
	Covid-19 Compartmental Constraints	Infectious Disease Forecasting	[QMB+25]
Vision	Symmetry	Image Classification	[CW16]
	Symmetry	Image Classification	[DFK16]
	Symmetry	Image Classification	[LYLC18]
	Symmetry	Image Classification	[WGTB17]
Manufacturing	Monotonicity	RUL Estimation	[AVS+22]
	Industrial System Dynamics	AHU Thermal Power Estimation	[IRD+25]

Table 3.1: Examples of domain-alignment scenarios.

Domain	Property	Application Example	Reference
Healthcare	Adversarial Robustness	Brain Tumor Classification	[YF25]
	Adversarial Robustness	Retinal OCT Classification	[LPZ21]
	Adversarial Robustness	Lung/Skin Lesion Segmentation	[HYL+19]
	Safety	Prostate Cancer Diagnosis	[JVW+25]
	Equalized Odds	SUD Treatment Success Estimation	[WY25]
	Equalized Odds	Dermatology Disease Classification	[SYY+23]
	Privacy	Pneumonia Detection	[JAK25]
Recruitment	Equal Opportunity – Demographic Parity	LinkedIn Talent Ranking	[GAK19]
	Disparate Treatment – Disparate Impact	Candidate Ranking	[SJ18]
	Explainability	Candidate Ranking	[HCR+22]
Finance	Individual Fairness	Income Estimation	[BPW+22]
	Individual Fairness	Credit Approval	[KS23]
	Adversarial Robustness	Fraud Detection	[FMK+21]
	Safety	Fraud Detection	[ADAA25]
	Privacy	Fraud Detection	[ZHD+23]
	Explainability	Credit Assessment	[dLMVW22]
Justice	Equal Opportunity – Demographic Parity	Recidivism Scoring	[WVP18]
	Individual Fairness	Community Crime Estimation	[LGW19]
	Explainability	Legal Case Matching	[YSX+22]
Education	Equalized Odds – Demographic Parity	Academic Success Estimation	[KKKS25]
	Intersectional Fairness	Academic Success Estimation	[PDD+25]
	Explainability	Student Feedback Providing	[SRD+25]
Automotive	Adversarial Robustness	Traffic Sign Recognition	[HH25]
	Adversarial Robustness	Traffic Estimation – Routing	[SMLD18]
	Safety	Lane Keeping	[PAU24]
	Safety	Emergency Braking	[SQL+24]

Table 3.2: Examples of human-alignment scenarios.

3.1.3 Property-Data Consistency

A crucial consideration arises when incorporating properties into data-driven models: the consistency between the property to enforce and the data representing the task to learn. Training a property-aware model is, essentially, a multi-objective problem, aiming at maximizing predictive performance while minimizing property violation.

Now, these two objectives are, in many cases, naturally aligned. Consider, for example, the task of estimating the remaining useful life of a machine, while forcing the learning model to behave monotonically with respect to time. Table 3.1 reports, under the manufacturing domain, an example of this task [AVS+22], representing a typical application in predictive maintenance, where the goal is to anticipate the failure of a production component. In this case, the task itself, and in particular the data representing it, can be assumed to be compatible with the property to incorporate: reasonably, the health conditions of a machine, over time, can only deteriorate, never improve. For this reason, a monotonicity-aware model, once successfully trained, is expected to predict more accurately than a monotonicity-agnostic one. In fact, in a similar scenario, property enforcement is performed precisely in the hope of improving the accuracy of the resulting model, especially in regime of data scarcity or noise.

However, there are cases in which the same assumption can not be made. As an example, consider the task of training a model to decide whether a job applicant is eligible for an interview, while enforcing fairness constraints with respect to race or gender. A task of this type, for which concrete references can be found in Table 3.2 under the recruitment domain [GAK19, SJ18, HCR+22], represents a typical practice in human resources departments. In this scenario, the data describing the task is very likely incompatible with the enforced fairness requirement: the data reflects human behavior, and humans, unfortunately, can be racist and sexist. This is the typical situation in which we aim at replicating the data, but only partially: we seek to capture useful information, while diverging from discriminatory patterns. This is, in the end, the final goal of the AI fairness literature, that is, preventing the learning model from capturing the biases hidden in the data. This is why, in this scenario, a fairness-aware model, trained and tested on the biased available dataset, should not be expected to outperform its fairness-agnostic counterpart. In contrast, in this case, a trade-off should be accepted: increased fairness can only be achieved at the cost of an accuracy drop.

The predictive performance of a property-aware, data-driven model, in other words, directly correlates with the consistency between the enforced property and

the training data, as represented in Figure 3.1: the higher the latter, the higher the former.

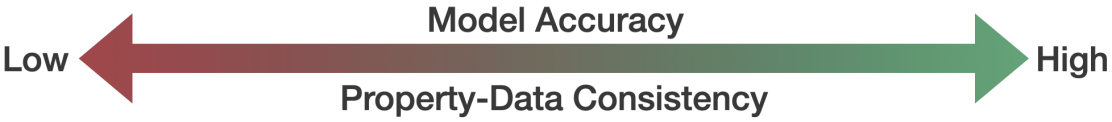


Figure 3.1: Correlation between model accuracy and property-data consistency.

Intuitively, and in line with the provided examples, property-data inconsistency occurs more frequently in human-alignment scenarios than in domain-alignment ones. In domain alignment, both the data from which the model learns, and the external knowledge with which the model is informed, derive from the domain in which the system is deployed. In human alignment, conversely, data and property may originate from different sources, not necessarily aligned with each other: the operational domain for the former, humans and societies for the latter.

The framework presented in this thesis is designed to handle both property-data consistent and inconsistent scenarios.

3.1.4 Property-Validity Guarantees

Besides property-data consistency, a second critical aspect should be considered when dealing with an enforcement scenario: the required level of property satisfaction guarantees.

There are cases in which such guarantees, although desirable, are not crucial. Consider again the RUL estimation example with monotonicity constraints discussed above. In this context, minor violations of the monotonicity property does not necessarily compromise the usefulness of the model, which may still be able to produce accurate and actionable predictions. Larger deviations could prevent the model from anticipating an imminent breakdown, thereby necessitating rescheduled maintenance and resulting in a waste of resources. The impact of such violations, in any case, remains limited to non-critical operational consequences. Given that a lack of guarantees does not necessarily prevent the adoption of the model, property enforcement, in such scenarios, can be eventually relaxed into a form of informing.

In other cases, however, full property satisfaction guarantees are crucial. In the job application scenario, even a sporadic violation of the fairness requirement may, for example, favor a male or White candidate over an equivalently qualified female

or Black competitor, which constitutes an ethically unacceptable behavior, with a substantial impact on individuals and societies. Beyond ethical considerations, moreover, a model designed to operate in this scenario, before being deployed, is likely to be evaluated for compliance with existing regulations, which may mandate formal fairness guarantees in its behavior. Therefore, in these cases, property enforcement must be rigorously performed until all residual violations are eliminated.

The computational cost of the enforcement procedure, in other words, directly relates to the required degree of guarantees on property satisfaction, as depicted in Figure 3.2: the higher the latter, the higher the former.

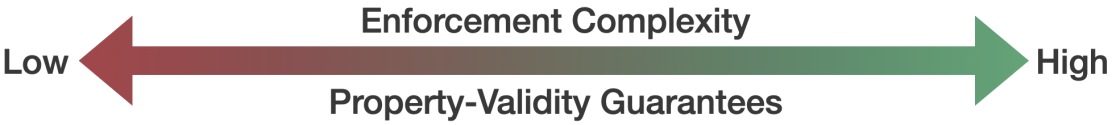


Figure 3.2: Correlation between enforcement complexity and property guarantees.

Property-validity guarantees are usually essential in human-alignment tasks, whereas they can often be relaxed in domain-alignment scenarios. In domain alignment, the primary objective of enforcement is model accuracy, which may benefit from the incorporation of properties even if their satisfaction is not fully guaranteed. In human alignment, in contrast, the main goal is to uphold human rights and safety, which must always be satisfied, in particular when deployment occurs in regulated contexts.

The emphasis in this work, in terms of guarantees, is in actually enforcing the property in the behavior of the model, rather than simply informing it about the desired requirement.

In the following section, we provide a formal definition of property, then we start delving into the technical details of the addressed problem.

3.2 A Taxonomy of Properties

The AI alignment literature has represented knowledge and requirements in a large variety of forms. In this work, we consider properties expressible as implications between two predicates Q and R, universally quantified over the input space of a ML model $f\colon \mathcal{X} \to \mathcal{Y}$:

$$\forall x_1, \ldots, x_k \in \mathcal{X}, \;\; Q(x_1, \ldots, x_k) \implies R(f(x_1), \ldots, f(x_k)) \tag{3.1}$$

where each x_i represents a distinct input vector. We distinguish properties according to two criteria: *arity* and *scope*. The arity of a property consists in the number of its variables k: properties of arity at most 1 are commonly referred to as *trace*, while those of arity larger than 2 are said to be *relational*. The scope of a property is *local* if the property is "anchored" in a finite set of data points, that is, some of the variables are *grounded*, so that Q or R are only defined over neighborhoods of them; *global* otherwise. Global relational properties generally allow to express more advanced requirements, but they are also the most difficult to handle, since they require reasoning over the entire model domain, and over multiple model executions, as clarified in the next section.

Equation (3.1) generalizes a broad range of requirements, as outlined in Table 3.3, where we define the most common properties considered in the literature, expressible through our formalism. As shown in the table, some properties can be specified, eventually with some adaptations, in multiple learning settings (e.g., injectivity, invariance and equivariance can be defined for any learning problem), while others only make sense in specific paradigms (e.g., inclusiveness and exclusiveness are only defined in multi-class classification). Moreover, we note that the arity of a property can always be reduced by grounding some of its variables at specific datapoints (dotted variables), so to obtain its adversarial or point-wise version, eventually ignoring the quantifier and the input predicate Q (when the arity is decreased to 0).

Boundedness constrains the output into a box $[l, u]$, for any input (e.g., medical dosage must be within a safe therapeutic range). *Stability* forces a bound Δ on the distance between two consecutive components of the output vector, again for any input (e.g., a multi-step time-series forecaster should not change its prediction drastically from one timestep to the next). *Safety* (linear) generalizes both boundedness and stability, by constraining the output into a polyhedron $Cf(x) \leqslant d$, whenever the input is constrained into a polyhedron $Ax \leqslant b$ (e.g., if an intruding plane is approaching from the right, the flight controller should diverge to the left). *Robustness* forces a bound $\epsilon \geqslant 0$ on the variation of the output, provided a bound $\delta \geqslant 0$ on the variation of the input (e.g., small input changes should produce small output changes). *Fairness* works similarly, except that the input variation is only computed over the non-sensitive features, i.e., over the complement $\mathcal{S}^c$ of the set of sensitive features $\mathcal{S}$ (e.g. two equivalently qualified job applicants should be treated similarly, regardless of their race), thus it can be considered as a generalization of robustness. *Exclusiveness*, in multi-label classification, encodes forbidden pairs (e.g., a patient can not be prescribed with two mutually contraindicated drugs), *Inclusiveness*, in the same learning paradigm, captures co-occurring features (e.g.,

a dog is also an animal). Both exclusiveness and inclusiveness, in particular, can be considered as combinatorial properties. *Monotonicity* (non-increasing) requires that the output does not increase whenever the sensitive components $\mathcal{S}$ of the input do (e.g., the predicted remaining useful life of a machine should not increase over time). *Injectivity* requires that different inputs are assigned with different predictions, and it is particularly desirable as it enables invertibility (e.g., in physics problems). *Invariance* enforces that the output remains unchanged under symmetric transformations of the input (e.g., rotating or translating an image does not alter its output). *Equivariance* requires that the output changes in a predictable or consistent manner under symmetric transformations of the input (e.g., rotating or translating an image produces a correspondingly rotated or translated output), thus it can be seen as a generalization of invariance. Injectivity, invariance and equivariance represent mathematical requirements, with the latter two properties encoding symmetry-preserving relations, this is why they are also called symmetries.

Note that our definition of a property, and consequently the compatibility of the presented enforcement framework, although reasonably general, does not fully cover the vast variety of specifications considered in the AI alignment literature. Equation (3.1), for example, is not particularly suited to encode physics knowledge represented in the form of differential equations, describing relations between functions and their spatial or temporal derivatives, such as the Eikonal equation [SCYP+20, bWAH+21], a non-linear first-order partial differential equation, and the Navier-Stokes equations [SGPW20], a system of non-linear second-order partial differential equations, both reported in Table 3.1. Another class of requirements, that are not captured by Equation (3.1), are those specified over distributions of inputs and outputs, like the different types of group fairness listed in Table 3.2: equalized odds [WY25, SYY+23], equal opportunity [GAK19, WVP18], disparate impact [SJ18], and demographic parity [KKKS25]. Finally, Equation (3.1) does not generalize to those properties that, rather than characterizing the behavior of the model, describe its interaction with the user, or its use of data: examples from Table 3.2 include explainability [HCR+22, SRD+25, YSX+22, dLMVW22] for the former, and privacy [JAK25, ZHD+23] for the latter. Extending the property compatibility of our framework, in particular to the so-called *functional properties*, i.e., those in which one of the two predicates in Equation (3.1) involves both inputs and outputs, capturing for example differential equations, represents one promising future direction of the present work.

In the following section, we formalize the property verification problem, in particular for NNs, then we discuss its complexity and review the existing literature.

$\mathcal{Y}$	**Property**	**Quantifier**	**Q**	**R**	**Arity**	**Scope**
$\mathbb{R}^m$	Pw. Boundedness	-	$\top$	$l \leqslant f(\dot{x}) \leqslant u$	0	local
	Pw. Stability	-	$\top$	$\bigwedge_{i\in[m-1]} \lvert f(\dot{x})_i - f(\dot{x})_{i+1}\rvert \leqslant \Delta$	0	local
	Pw. Safety	-	$\top$	$Af(\dot{x}) \leqslant b$	0	local
	Pw. Robustness/Fairness	-	$\top$	$\lVert f(\dot{x}) - \ddot{y}\rVert_p \leqslant \epsilon$	0	local
	Adv. Robustness	$\forall x \in \mathcal{X}$	$\lVert x - \dot{x}\rVert_p \leqslant \delta$	$\lVert f(x) - \dot{y}\rVert_p \leqslant \epsilon$	1	local
	Adv. Fairness	$\forall x \in \mathcal{X}$	$\lVert x_{\mathcal{S}^c} - \dot{x}_{\mathcal{S}^c}\rVert_p \leqslant \delta$	$\lVert f(x) - \dot{y}\rVert_p \leqslant \epsilon$	1	local
	Boundedness	$\forall x \in \mathcal{X}$	$\top$	$l \leqslant f(x) \leqslant u$	1	global
	Stability	$\forall x \in \mathcal{X}$	$\top$	$\bigwedge_{i\in[m-1]} \lvert f(x)_i - f(x)_{i+1}\rvert \leqslant \Delta$	1	global
	Safety	$\forall x \in \mathcal{X}$	$Ax \leqslant b$	$Cf(x) \leqslant d$	1	global
	Robustness	$\forall x', x'' \in \mathcal{X}$	$\lVert x' - x''\rVert_p \leqslant \delta$	$\lVert f(x') - f(x'')\rVert_p \leqslant \epsilon$	2	global
	Fairness	$\forall x', x'' \in \mathcal{X}$	$\lVert x'_{\mathcal{S}^c} - x''_{\mathcal{S}^c}\rVert_p \leqslant \delta$	$\lVert f(x') - f(x'')\rVert_p \leqslant \epsilon$	2	global
$[m]$	Pw. Robustness/Fairness	-	$\top$	$f(\ddot{x}) = \dot{y}$	0	local
	Adv. Robustness	$\forall x \in \mathcal{X}$	$\lVert x - \dot{x}\rVert_p \leqslant \delta$	$f(x) = \dot{y}$	1	local
	Adv. Fairness	$\forall x \in \mathcal{X}$	$\lVert x_{\mathcal{S}^c} - \dot{x}_{\mathcal{S}^c}\rVert_p \leqslant \delta$	$f(x) = \dot{y}$	1	local
	Robustness	$\forall x', x'' \in \mathcal{X}$	$\lVert x' - x''\rVert_p \leqslant \delta$	$f(x') = f(x'')$	2	global
	Fairness	$\forall x', x'' \in \mathcal{X}$	$\lVert x'_{\mathcal{S}^c} - x''_{\mathcal{S}^c}\rVert_p \leqslant \delta$	$f(x') = f(x'')$	2	global
$\{0,1\}^m$	Pw. Exclusiveness	-	$\top$	$\bigwedge_{(i,j)\in\mathcal{S}} f(\dot{x})_i + f(\dot{x})_j \leqslant 1$	0	local
	Pw. Inclusiveness	-	$\top$	$\bigwedge_{(i,j)\in\mathcal{S}} f(\dot{x})_i \leqslant f(\dot{x})_j$	0	local
	Exclusiveness	$\forall x \in \mathcal{X}$	$\top$	$\bigwedge_{(i,j)\in\mathcal{S}} f(x)_i + f(x)_j \leqslant 1$	1	global
	Inclusiveness	$\forall x \in \mathcal{X}$	$\top$	$\bigwedge_{(i,j)\in\mathcal{S}} f(x)_i \leqslant f(x)_j$	1	global
$\mathbb{R}$	Pw. Monotonicity ($\downarrow$)	-	$\top$	$f(\dot{x}) \leqslant f(\ddot{x})$	0	local
	Adv. Monotonicity ($\downarrow$)	$\forall x \in \mathcal{X}$	$x_{\mathcal{S}} \geqslant \dot{x}_{\mathcal{S}} \wedge x_{\mathcal{S}^c} = \dot{x}_{\mathcal{S}^c}$	$f(x) \leqslant f(\dot{x})$	1	local
	Monotonicity ($\downarrow$)	$\forall x', x'' \in \mathcal{X}$	$x'_{\mathcal{S}} \geqslant x''_{\mathcal{S}} \wedge x'_{\mathcal{S}^c} = x''_{\mathcal{S}^c}$	$f(x') \leqslant f(x'')$	2	global
Any	Pw. Injectivity	-	$\top$	$f(\dot{x}) \neq f(\ddot{x})$	0	local
	Pw. Invariance/Equivariance	-	$\top$	$f(\dot{x}) = f(\ddot{x})$	0	local
	Adv. Injectivity	$\forall x \in \mathcal{X}$	$x \neq \dot{x}$	$f(x) \neq f(\dot{x})$	1	local
	Injectivity	$\forall x', x'' \in \mathcal{X}$	$x' \neq x''$	$f(x') \neq f(x'')$	2	global
	Invariance	$\forall x', x'' \in \mathcal{X}$	$x'' = \pi(x')$	$f(x'') = f(x')$	2	global
	Equivariance	$\forall x', x'' \in \mathcal{X}$	$x'' = \pi_{in}(x')$	$f(x'') = \pi_{out}(f(x'))$	2	global

Table 3.3: Commonly adopted properties, expressed according to our formalism.

3.3 A Taxonomy of Verifiers

Historically, the AI alignment community has focused on deciding the validity of properties. This extensive branch of the literature traces its genesis to a paper published by Alan Turing in 1949, titled *Checking a Large Routine* [Tur49], where the author addresses the following question: *how can we prove that a program does what it is supposed to do?* Then, he provides a proof of correctness for a program implementing the factorial function, that is, he proves that his piece of code always terminates, and always produces the factorial of its input.

Formally, in the ML setting, given a model f and a property as defined in Equation (3.1), the corresponding *verification problem* can be formulated as:

$$\exists x_1, \ldots, x_k \in \mathcal{X} \colon \; Q(x_1, \ldots, x_k) \wedge \neg R(f(x_1), \ldots, f(x_k)) \tag{3.2}$$

A verification problem, in other words, is a decision problem, which consists either in providing a certificate of property violation $(x_1^*, \ldots, x_k^*)$, called *counterexample*, or in proving that no such certificate exists. Figure 3.3 depicts a counterexample for the safety property $\forall x \in \mathcal{X},\ x \in \mathcal{Q} \implies f(x) \in \mathcal{R}$: the point x^*, falling within the hazardous input region $\mathcal{Q} = \{x \in \mathcal{X} \colon Ax \leqslant b\}$, is propagated outside the safe output region $\mathcal{R} = \{y \in \mathcal{Y} : Cy \leqslant d\}$.

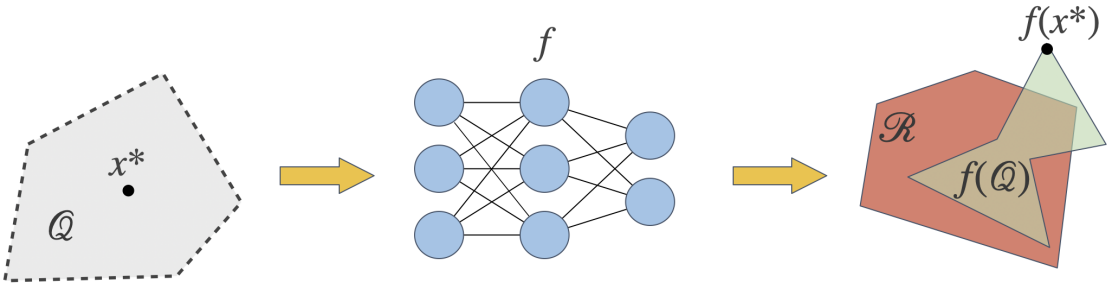


Figure 3.3: A counterexample for a safety property.

The difficulty of Equation (3.2) depends on several factors. First, it is determined by the degree of the predicates Q and R: non-linear predicates are evidently harder to verify than linear ones. Second, it is influenced by the typically infinite and high-dimensional input domain $\mathcal{X}$, especially if the property is global: the larger this domain, the broader the search space for the counterexample. Third, it is crucially impacted by the arity of the property: the greater the arity, the larger the number of decision variables of the problem. Finally, and most importantly, it is affected by the complexity of the model to verify: in particular, when f is a ReLU FNN, and Q and R are conjunctions of linear constraints (i.e. a safety property), Equation (3.2) has been shown to be NP-hard [KBD+17]. We can deduce,

in turn, that the problem is at least NP-hard for all networks and properties that subsumes this case.

An algorithm designed to solve a verification problem is called a *verifier*, and can be distinguished according to two criteria: *soundness* and *completeness*. A verifier is sound if every property it declares as valid is indeed valid, and complete if it declares every valid property as valid. Soundness and completeness can be achieved by reasoning exactly over all possible input-output patterns of the model, which guarantees a correct verdict for each property. Computationally speaking, however, this is not always feasible, for the aforementioned reasons. The only way to handle the complexity of this problem, when it becomes intractable, is to resort to approximate reasoning, hence inevitably sacrificing either soundness or completeness. Unsoundness arises through underapproximation, which may produce false positives: the property is declared valid, but actually it is not. Incompleteness instead occurs via overapproximation, which may produce false negatives, i.e., spurious counterexamples: the property is declared invalid, but actually it is.

The verification community has produced a plethora of verifiers for ML models, particularly for NNs. The great majority of them, however, are limited to trace properties, while relational verifiers have been emerging only recently. In the following, we outline the most representative NN verifiers, without aiming for a comprehensive review, which is beyond the scope of this thesis. Specifically, we identify three main lines of work within the verification literature: *optimization and searching*, *reachability analysis* and *heuristic attacks*.

Optimization and searching approaches essentially embed the fully-fledged ML model into a CO model, by encoding it into the language of the chosen technology, such as Mixed-Integer Programming (MIP), Boolean Satisfiability (SAT), or Satisfiability Modulo Theory (SMT). In other words, they turn the input and output variables of the ML model into the decision variables of a CO model, which is then solved either via a specifically designed algorithm, or through a general-purpose solver. These methods typically offer both soundness and completeness, as long as the model encoding is kept exact, and no early stopping criterion is adopted for the optimization/searching procedure. Representative examples include, for trace properties, the SAT-based EVV [JR20], the SMT-based Reluplex [KBD+17] and Marabou [WIZ+24], and the MIP-based MIPVerify [TXT17] and ILP [BIL+16], while for relational properties, the MIP-based CertiFair [KS23].

Reachability analysis algorithms analyze the layer-by-layer propagation of a set of inputs through a neural network, in order to reconstruct the set of reach-

able outputs, hence to check whether any of them falls into an unsafe region. By maintaining an outerapproximation of the input set during the propagation, these methods offer increased scalability, but at the price of completeness, even though some methods manage to preserve it by reconstructing the output exactly. Examples in this class include, for trace properties, AI^2 [GMDC$^+$18], MaxSens [XTJ18], PRIMA [MMS$^+$22], DeepZ [SGM$^+$18] and ExactReach [XTRJ18], while for relational ones, FairNNV [TMLR$^+$24].

Finally, heuristic attacks consist in approximate procedures to generate counterexamples, typically based on gradient descent. These methods offer completeness and high scalability, but do not provide soundness. While they have been widely applied to trace properties, with examples including FGSM [GSS14], DeepFool [MDFF16], C&W [CW17], PGD [MMS$^+$19], and HCLU [GPSB19], to the best of the author's knowledge, no analogous approach have been developed for relational ones. Note that the formal verification literature typically assumes soundness, that is, it does not consider heuristic attacks as proper verifiers, but merely as testing methods.

Beyond these approaches, there are those that combine the three paradigms in a hybrid fashion, such as NeVer2 [DGP$^+$24] and α, β-Crown [XZW$^+$21, WZX$^+$21] for trace properties, and RaVeN [BXS24] and RACoon [BS24] for relational ones.

A comprehensive survey on NN verification can be found in [LAL$^+$21], while a comparison among the state-of-art verifiers is provided by the annual competition VNN-COMP [BBJW24, KLB$^+$25], currently at the 6$^{\text{th}}$ edition. These surveys are, however, both limited to trace properties. In particular, again to the best of the author's knowledge, no comprehensive survey has been proposed, thus far, for relational verifiers, evidently because this line of research is still its early stage. The methods discussed above are outlined in Table 3.4.

In this thesis, we design a sound but incomplete optimization-based verifier, which is able to handle both trace and relational properties, and which we execute as a subroutine of our property-enforcement algorithm.

3.4 A Taxonomy of Enforcers

While the AI alignment community has long devoted substantial effort to verifying the validity of properties, recent years have witnessed a shift of interest toward actually enforcing them. Verification is, by nature, a passive practice: it enables confident deployment when the property is satisfied, but it offers no remedy when

Approach	Example	Sound	Complete	Arity	Reference
Optimization & Searching	EVV	•	•	1	[JR20]
	Reluplex	•	•	1	[KBD+17]
	Marabou	•	•	1	[WIZ+24]
	MIPVerify	•	•	1	[TXT17]
	ILP	•	○	1	[BIL+16]
	Certifair	•	○	2	[KS23]
Reachability Analysis	AI^2	•	○	1	[GMDC+18]
	MaxSens	•	○	1	[XTJ18]
	PRIMA	•	○	1	[MMS+22]
	DeepZ	•	○	1	[SGM+18]
	ExactReach	•	•	1	[XTRJ18]
	FairNNV	•	○	2	[TMLR+24]
Heuristic Attacks	FGSM	○	•	1	[GSS14]
	DeepFool	○	•	1	[MDFF16]
	C&W	○	•	1	[CW17]
	PGD	○	•	1	[MMS+19]
	HCLU	○	•	1	[GPSB19]
Hybrid	NeVer2	•	○	1	[DGP+24]
	α, β-Crown	•	○	1	[WZX+21]
	RaVeN	•	○	2	[BXS24]
	RACoon	•	○	2	[BS24]

Table 3.4: Representative NN verifiers.

the property is instead violated. Enforcement, on the other hand, is a proactive approach: it forces property satisfaction, by actively eliminating potential violations.

Formally, given a model space Ω, a loss function L, and a dataset $\mathcal{D}$, as defined in Equation (2.2), as well as a property as defined in Equation (3.1), the corresponding *enforcement problem* can be formulated as follows:

$$\arg\min_{f \in \Omega} L_{\mathcal{D}}(f) \tag{3.3a}$$

$$\text{s.t. } \forall x_1, \ldots, x_k \in \mathcal{X}, \;\; Q(x_1, \ldots, x_k) \implies R(f(x_1), \ldots, f(x_k)) \tag{3.3b}$$

An enforcement problem, in other words, is a constrained optimization problem, consisting in finding a model f^* that (i) minimizes the training error on the given data (*accuracy*), while (ii) satisfying the desired property (*feasibility*).

If verifying the validity of a property in a network is hard, enforcing the property into such network is, unfortunately, even harder. Precisely, [ML23] proves

that enforcing a trace property (specifically adversarial robustness around a single example) into a neural network is Σ_2^P-hard, where $\Sigma_2^P \supsetneq$ NP, assuming the Polynomial Hierarchy conjecture [Toc76], i.e., a generalization of the P$\neq$NP conjecture. In other words, enforcing is, in general, *strictly harder* than verifying, unless P=NP. The intuition underlying this claim is the following: in order to enforce, one needs to attack!

To see why, let us consider the decision version of Equation (3.3), obtained by ignoring the optimality requirement, that is:

$$\exists f \in \Omega : \ \forall x_1, \ldots, x_k \in \mathcal{X}, \ \ Q(x_1, \ldots, x_k) \implies R(f(x_1), \ldots, f(x_k)) \tag{3.4}$$

The computational asymmetry between the two problems evidently arises by comparing Equation (3.2) with Equation (3.4). Verification is an *existential* problem, formulated as an $\exists$ query: find *one* counterexample that violates the property. Enforcement, on the other hand, is a *universal* problem, expressible as an $\exists\forall$ query: find a model for which *no* counterexample exists that violates the property. In other words, solving an enforcement problem requires iteratively solving a nested verification one: during the search, each candidate solution f needs to be verified, until a feasible one f^* is found (if it exists). Figure 3.4 provides an illustration of an enforcement process. This computational dependency helps explain why, in comparison to the verification literature, the enforcement one is currently much narrower: before starting to address enforcement problems, we firstly needed to develop powerful verifiers.

An algorithm designed to solve Equation (3.3) is called an *enforcer*, and can be classified according to its *strength* with respect to the enforced property, for which we distinguish three standard levels: *guaranteed*, *certified* and *heuristic*. We say that an enforcer is guaranteed if it produces unattackable models: any *complete* verifier, executed on this model, would fail to return a counterexample. An enforcer is certified if it produces models equipped with an online defense procedure: at inference time, for any given input, the method issues, together with the corresponding prediction, either a certificate of property satisfaction, or a warning of potential violation. Finally, an enforcer is heuristic if it only provides statistical guarantees on property satisfaction: the produced models respect the given property on a certain percentage of test instances, but without supplying their predictions with formal guarantees or input-specific certifications. Heuristic guarantees are weaker than certifications, which are in turn weaker than formal guarantees: heuristic enforcers increase the likelihood of property satisfaction, certified enforcers offer a mechanism to detect and reject potential violations, guaranteed enforcers eliminate such violations completely.

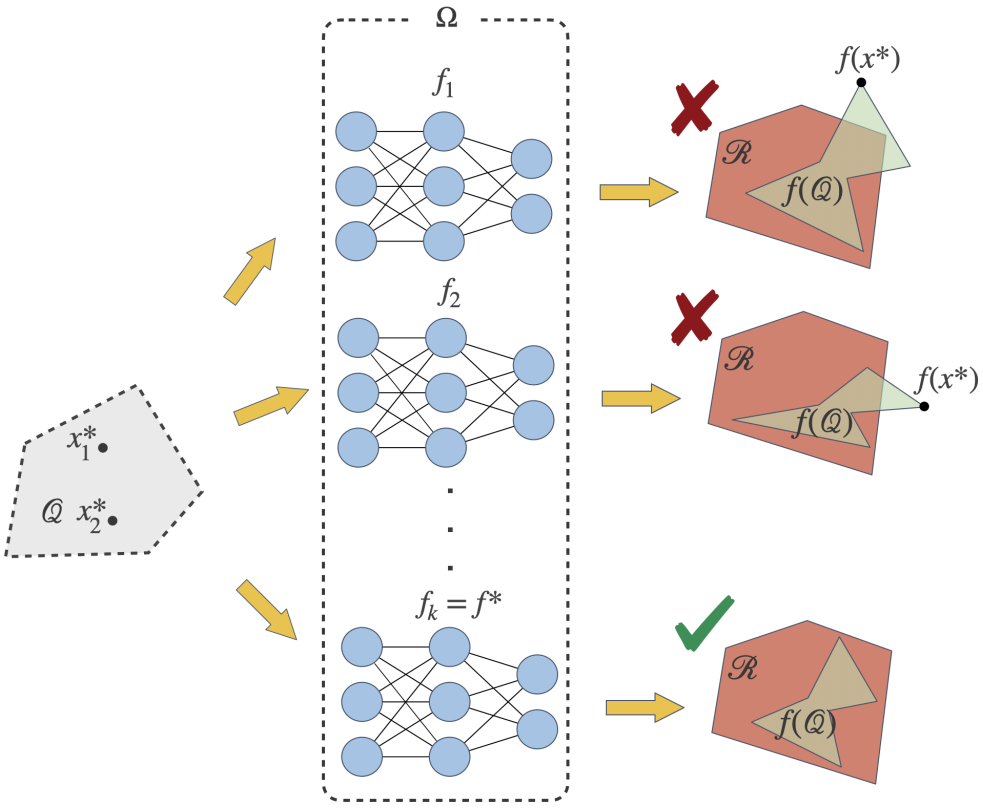


Figure 3.4: An illustration of an enforcement process.

Note that there are cases in which formal guarantees are excessively restrictive and can lead to degenerate solutions. For example, in a classification setting, guaranteeing adversarial robustness around every input is unrealistic: no classifier can be adversarially robust everywhere, unless it is constant, i.e., it always outputs the same class. This is because a classifier prediction changes abruptly at the decision boundary, meaning that, for any input sufficiently close to this boundary, there always exists an admissible perturbation that moves such input to the other side (if the input space is continuous), rendering the model attackable. For this reason, in such contexts, certifications represent the strongest realistic form of enforcement. Moreover, even when theoretically possible, a strong enforcement may not be feasible in practice, given the complexity of the problem. The only way to handle this complexity, when it becomes intractable, is to resort either to underenforcement or to overenforcement: both approaches increase tractability, the former by sacrificing property guarantees or certifications, the latter by jeopardizing model expressivity.

Similarly to the verification community, enforcement research has mostly focused on trace properties, while relational ones have started to attract attention

only recently. In the following, we present an overview of enforcement methods for NNs. It should be noted, however, that this discussion does not aim to provide a comprehensive survey of the field, which is beyond the scope of this thesis. Specifically, we distinguish three main enforcement paradigms, according to the stage of the ML pipeline in which Equation (3.3b), i.e., the feasibility condition of the problem, is handled: *Preprocessing*, *Inprocessing* and *Postprocessing*. Within this scope, the model space Ω is meant as the space of network parameters θ.

Preprocessing. These methods address property constraints before starting the actual training, by enforcing them through *data transformations* or *architecture design*. In this way, they ignore Equation (3.3b) in the enforcement problem, and reduce the constrained training to a standard one, solvable via gradient descent. With data transformations, we refer to those methods that encode the property into the dataset $\mathcal{D}$, by means of transformations applied to the raw input or output, which produce a new dataset $\mathcal{D}' = \{(\tau_{\text{in}}(x), \tau_{\text{out}}(y)),\ \forall (x, y) \in \mathcal{D}\}$, with $\tau_{in} \colon \mathcal{X} \to \mathcal{X}$ and $\tau_{out} \colon \mathcal{Y} \to \mathcal{Y}$. Such transformations are used either to augment the data to reinforce property representation, in which case $\mathcal{D}$ is replaced with $\mathcal{D} \cup \mathcal{D}'$, or to sanitize it by removing property violations, in which case $\mathcal{D}$ is replaced with $\mathcal{D}'$. Data augmentation is proposed in DA [QRLB20] and SAT [ZQMQ20], which expands the training data with rotated, translated and scaled input instances, to promote adversarial invariance and adversarial robustness on the training set, respectively. Data purification is instead used in Massaging [KC12], which encourages adversarial fairness, again on the training instances, by relabeling the unfair ones, and in OB [CZ24], which enforces the same property by means of orthogonal-to-bias transformations, designed to eliminate the influence of a group on sensitive variables. Data transformation methods generate heuristic enforcers, only able to provide statistical guarantees. Architecture design methods compile the property directly into the network architecture, that is, they design f so that it is feasible for each parameterization $\theta \in \Omega$. These methods are able to guarantee the feasibility of the property, such as TFENNs [GKR24] and DREN [LYLC18] for equivariance, GloRo Nets [LWF21] for robustness, and ConstrMonoNet [RS23] for monotonicity.

Inprocessing. These approaches perform property enforcement at training time, that is, they address Equation (3.3) directly. We further distinguish two lines of work within this class: *regularization* and *projection*. The core idea underlying regularization is to define a function $\nu \colon \Omega \times \mathcal{X} \to \mathbb{R}_+$, to measure the property violation of the network parameterization $\theta \in \Omega$ on a given combination of variables $x_1, \ldots, x_k \in \mathcal{X}$, hence to turn the nested verification problem, i.e., Equation (3.3b), into an optimization one, consisting in finding a combination $x_1^*, \ldots, x_k^*$ that max-

imizes such violation (i.e., the worst counterexample). Consequently, the constrained optimization problem, defined in Equation (3.3), can be turned into the following unconstrained one:

$$\arg\min_{\theta \in \Omega} \; L_{\mathcal{D},\text{task}}(\theta) + \lambda L_{\text{prop}}(\theta) \tag{3.5a}$$

$$\text{with:} \; L_{\text{prop}}(\theta) = \max_{x_1,\ldots,x_k \in \mathcal{X}} \nu(\theta; x_1, \ldots, x_k) \tag{3.5b}$$

where $L_{\mathcal{D},\text{task}}$ denotes the standard data-dependent loss, the *regularizer* L_{prop} represents the maximum violation, while the parameter $\lambda \geqslant 0$ is used to control the effect of the regularization. Note that the property loss can also be data-dependent, e.g., for local properties, in which case we denote it with $L_{\mathcal{D},\text{prop}}$. Equation (3.5) can be solved via standard gradient descent, or via Dual Ascent, which consists in turning λ into a trainable variable, hence in performing, at each iteration, a gradient descent step to update θ, and a gradient ascent one to update λ. Note that, however, in order to adopt a gradient-based approach, the regularizer L_{prop} needs to be differentiable. The strength of the enforcement, in these approaches, mostly depends on the verification technology adopted to solve the inner maximization problem: an unsound verifier would make the resulting enforcement heuristic, while a complete verifier would be potentially able to provide certifications or formal guarantees, depending on the property to enforce, as well as on the ability of the training algorithm to nullify the violation by the end of the process, i.e., to converge to $L_{\text{prop}} = 0$. A notable example for this paradigm is represented by the adversarial training (AT) [MMS+18], which enforces adversarial robustness around the training data, typically by computing property violation via an adversarial attack, hence resulting in a heuristic enforcer. Similarly to AT, CROWN-IBP [ZCX+20] computes robustness violation on the training instances, but differently from it, it adopts a complete verifier, which can be executed at both training and inference time, hence certifiably enforcing adversarial robustness around any input instance. On the same line of CROWN-IBP, certified enforcement is provided by MonoNet [LHZL20] and COMET [SFMVdB20] for adversarial monotonicity, and by IFNN [BPW+22] for adversarial fairness. CertiFair [KS23] and ART [LZSJ19], instead, not only adopt complete verification for the inner problem, but they run it globally, with the former providing certified guarantees for fairness, the latter formal ones for safety. Projection approaches work instead by enforcing feasibility via projection mechanisms, operating either on the output space $\mathcal{Y}$, or on the parameter one Ω. Enforcing via output projection means appending a *neural projector* $\Pi_{\mathcal{Y}}$ to the last layer of the network, which corrects any infeasible output by projecting it onto the feasible region, hence allowing to ignore the property constraints, i.e., Equation (3.3b), in the enforcement problem. The challenge for these methods resides in designing $\Pi_{\mathcal{Y}}$ so that it is differentiable, in order to enable a gradient-based

training, and efficient, in order to keep the process tractable. These approaches are able to produce guaranteed enforcers, such as HardNet [MA25] for safety, C-HMCNN [GL21] for inclusiveness, and ENFORCE [LS25] for any trace property. Enforcing through weight projection consists instead in designing a *weight projector* Π_Ω, which corrects any infeasible configuration θ to restore the feasibility of the resulting model f. This projector is then executed as a subroutine of gradient descent after each parameter update. Π_Ω does not need to be differentiable, but it obviously needs to be tractable. Similarly to output projection methods, also these approaches are able to provide formal guarantees on the enforced property. A notable example is DeepSaDe [GDB24], compatible with any trace property.

Postprocessing These methods enforce the desired property at post-training phase, either via *input/output purification* or through *model repair*, that is, they train via standard gradient descent, hence they handle the constraints in Equation (3.3b) after training. Purification approaches work either right before the inference process, by purifying the input in order to decrease the chance for the model to produce an infeasible prediction, or right after it, by correcting such prediction to restore its feasibility. Examples of pre-inference heuristic enforcers include, for adversarial robustness, feature distillation (FD) [LLL+19], which attempts to suppress eventual adversarial perturbations in any input by means of JPEG compression and quantization, PixelDefend [SKN+18], which moves the malicious input towards the distribution seen in the training data, and Randomized Smoothing [CRK19], which provides probabilistic guarantees on the feasibility of the output, by considering multiple noisy copies of the input. A post-inference guaranteed enforcer is instead FETA [MSF23], which identifies a counterexample for the given input, hence corrects the model output accordingly, in order to guarantee adversarial fairness. Model repair techniques work instead by minimally changing the weights of a trained network in order to restore its feasibility on a given finite set of violated inputs. Examples are represented by 3M-DNN [RK22] and DDNN [ST21], both able to provide formal guarantees, the former for pointwise robustness, the latter also for adversarial one.

An illustration of our taxonomy of enforcers is depicted in Figure 3.5, while an overview of the enforcers discussed above is provided in Table 3.5, where the "strength" column indicates whether the enforcer is heuristic (○○), certified (●○), or guaranteed (●●).

The table highlights the current gaps within the property enforcement literature, precisely: the *lack of generality*, with respect to both property and model compatibility, and the *lack of guarantees*. Many of the existing enforcers, indeed,

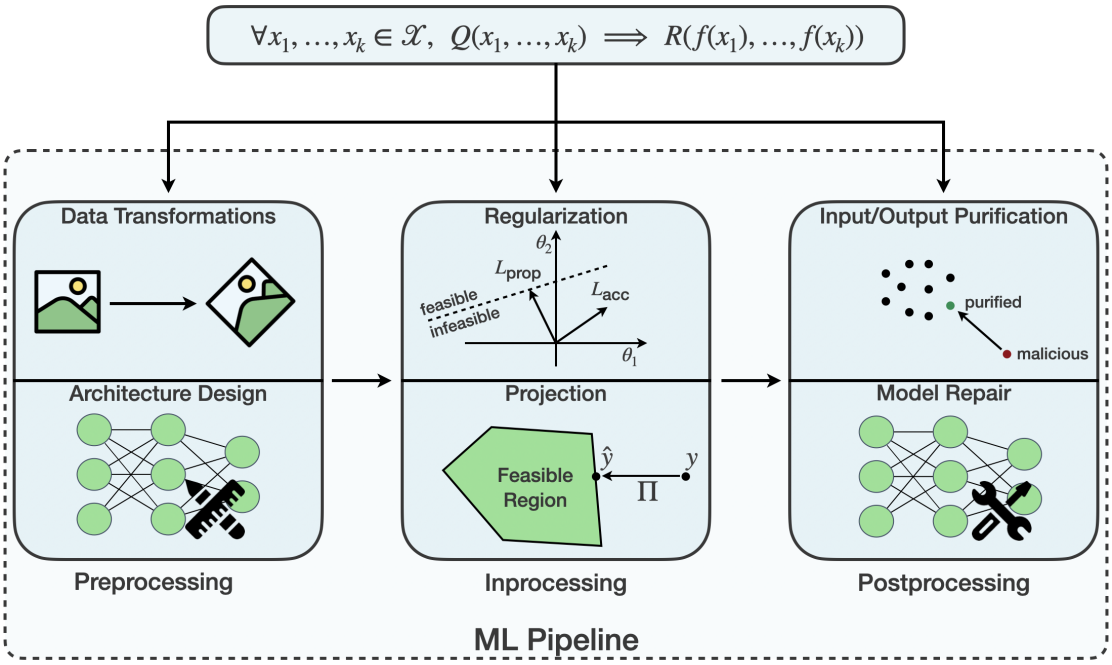


Figure 3.5: Our taxonomy of enforcers.

are only able to enforce specific properties, mostly trace ones, by relying on strong architectural assumptions, e.g., small ReLU networks, and most of them fail to provide formal satisfaction guarantees.

With this thesis, we contribute to bridge these gaps by proposing a framework, called *SMiLE* (*Safe Machine Learning via Embedded Overapproximation*), capable of enforcing generic trace and relational properties into arbitrary neural networks, providing full satisfaction guarantees, while remaining competitive with property-specific baselines. Our framework, formalized in the next chapter, consists of two main components: (i) a verification-friendly neural architecture, designed to simplify the verification problem in Equation (3.2), and (ii) a dedicated training algorithm, developed to address the enforcement property in Equation (3.3), which relies on weight projection techniques for trace properties, and on regularization schemes for relational ones. This is why, within the enforcement taxonomy provided in this section, our enforcer can be seen as hybrid preprocessing and inprocessing approach.

Stage	Approach	Example	Property	Model	Strength	Reference
Preprocessing	Data Transformation	DA	Adv. Invariance	Any	○○	[QRLB20]
		SAT	Adv. Robustness	Any	○○	[ZQMQ20]
		Massaging	Adv. Fairness	Any	○○	[KC12]
		OB	Adv. Fairness	Any	○○	[CZ24]
	Architecture Design	TFENNs	Equivariance	FNN/RNN	●●	[GKR24]
		DREN	Equivariance	ReLU CNN	●●	[LYLC18]
		GloRo Nets	Robustness	Any	●●	[LWF21]
		ConstrMonoNet	Monotonicity	ReLU FNN	●●	[RS23]
Inprocessing	Regularization	AT	Adv. Robustness	Any	○○	[MMS+18]
		CROWN-IBP	Adv. Robustness	Any	●○	[ZCX+20]
		MonoNet	Adv. Monotonicity	ReLU FNN	●○	[LHZL20]
		COMET	Adv. Monotonicity	ReLU FNN	●○	[SFMVdB20]
		IFNN	Adv. Fairness	FNN	●○	[BPW+22]
		CertiFair	Fairness	ReLU FNN	●●	[KS23]
		ART	Safety	ReLU FNN	●●	[LZSJ19]
	Projection	HardNet	Safety	Any	●●	[MA25]
		C-HMCNN	Inclusiveness	Any	●●	[GL21]
		ENFORCE	Any Trace	Any	●●	[LS25]
		DeepSade	Any Trace	Any	●●	[GDB24]
Postprocessing	Input/Output Purification	FD	Adv. Robustness	Any	○○	[LLL+19]
		PixelDefend	Adv. Robustness	Any	○○	[SKN+18]
		RandSmooth	Adv. Robustness	Any	○○	[CRK19]
		FETA	Adv. Fairness	Any	●●	[MSF23]
	Model Repair	DDNN	Pw./Adv. Robustness	FNN	●●	[ST21]
		3M-DNN	Pw. Robustness	FNN	●●	[RK22]

Table 3.5: Representative NN enforcers.

Chapter 4

Methodology

In this chapter, we introduce the main contribution of this thesis: a property enforcement framework, called *SMiLE* (*Safe Machine Learning via Embedded Overapproxiamtion*), consisting of a verification-friendly neural architecture and two dedicated property-aware training algorithms, specialized to trace and relational properties, respectively. In particular, we start by presenting the SMiLE *architecture*, showing how it can be derived from any standard neural network, and discussing its advantages from a property verification and enforcement perspective. We then introduce the SMiLE *trace enforcer*: we formalize its two key algorithmic components, namely, a *counterexample generator* and a *weight projector*, then we combine them into the complete enforcement framework. Finally, following a similar scheme, we formalize the SMiLE *relational enforcer*: we introduce the corresponding projector and generator, then the resulting enforcement framework.

4.1 Architecture

Let $f\colon \mathcal{X} \to \mathcal{Y}$ be neural network of depth d, as defined in Equation (2.4). A SMiLE architecture can be built from f as follows.

We start by viewing f as a composition of an *embedding function* $h\colon \mathcal{X} \to \mathbb{R}^{n_{d-1}}$, also referred to as the *backbone*, which maps the input x to an *embedding* z into a *latent space* $\mathbb{R}^{n_{d-1}}$, and a *linear output function* $g\colon \mathbb{R}^{n_{d-1}} \to \mathcal{Y}$, also referred to as the *head*, which propagates such embedding to the output y. Formally:

$$f = \overbrace{f_d}^{g} \circ \overbrace{f_{d-1} \circ \cdots \circ f_1}^{h} \tag{4.1}$$

or equivalently,

$$y = f(x;\theta) = g(z;\theta_g) \circ h(x;\theta_h) \tag{4.2}$$

where $\theta = (\theta_h, \theta_g)$ represents the parameter vector of f, composed by the parameters of the sub-networks h and g. This decomposition ensures that, while the backbone h can be arbitrarily complex, as it may include any number of layers, the head g is instead simple by construction, since it always consists of the last linear layer. This decomposition, depicted in Figure 4.1, is both natural and common for many neural architectures, including classifiers, in which case the output y is meant as a logit vector.

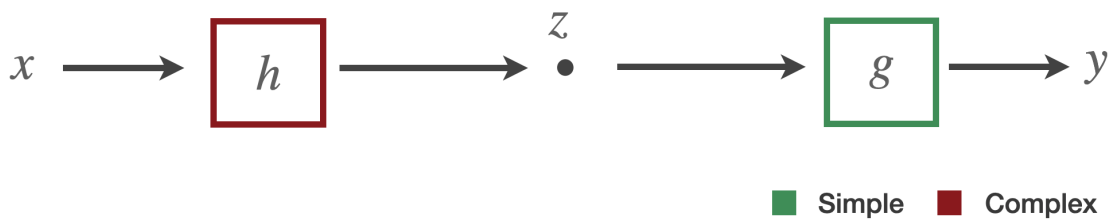


Figure 4.1: Our decomposition of a standard network f.

Then, we embed a trainable *overapproximator* into f, consisting of two differentiable *auxiliary models* $\underline{h}\colon \mathcal{X} \to \mathbb{R}^{n_{d-1}}$ and $\overline{h}\colon \mathcal{X} \to \mathbb{R}^{n_{d-1}}$, and a *clipping operator* $\text{clip}\colon \mathbb{R}^{n_{d-1}} \to \mathbb{R}^{n_{d-1}}$, defined component-wise as:

$$\text{clip}(z; l, u) = \max(\min(z, u), l) \tag{4.3}$$

with $l, u \in \mathbb{R}^{n_{d-1}}$. Precisely, the overapproximator operates within the latent space $\mathbb{R}^{n_{d-1}}$, by clipping the embedding z, produced for each input x, in between the lower and upper bounds predicted by $\underline{h}$ and $\overline{h}$. The resulting SMiLE version of f can then be written as:

$$y_{\text{smile}} = f_{\text{smile}}(x; \theta_{\text{smile}}) = g(\overline{\underline{z}}; \theta_g) \circ \text{clip}(z; \underline{h}(x; \theta_{\underline{h}}), \overline{h}(x; \theta_{\overline{h}})) \circ h(x; \theta_h) \tag{4.4}$$

with $\overline{\underline{z}}$ denoting the *clipped embedding*, while $\theta_{\text{smile}} = (\theta_h, \theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g)$ representing the parameter vector of f_{smile}, composed by the parameters of the sub-networks $h, \underline{h}, \overline{h}$ and g. This overapproximation mechanism ensures that, for each input x, the corresponding clipped embedding $\overline{\underline{z}}$ lies inside the bounding box

$$\overline{\underline{H}}(x; \theta_{\underline{h}}, \theta_{\overline{h}}) = [\underline{h}_1(x), \overline{h}_1(x)] \times \ldots \times [\underline{h}_{n_{d-1}}(x), \overline{h}_{n_{d-1}}(x)] \tag{4.5}$$

before it is propagated through g to produce the output y_{smile}. An illustration of a SMiLE architecture is provided in Figure 4.2.

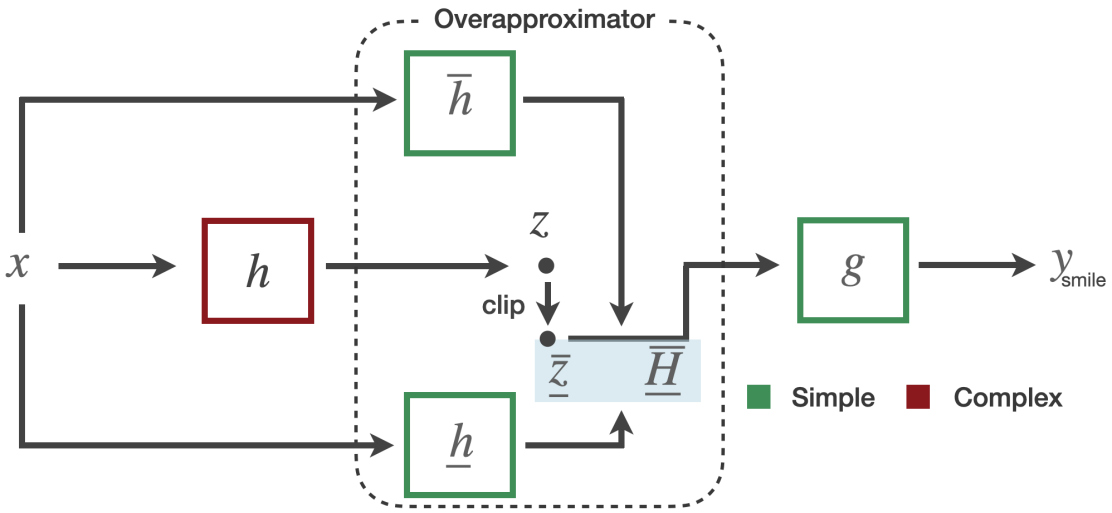


Figure 4.2: A SMiLE architecture.

Intuitively, the auxiliary models attempt to locate the embedding z within the latent space $\mathbb{R}^{n_{d-1}}$, by enclosing it into the corresponding box $\overline{\underline{H}}(x; \theta_{\underline{h}}, \theta_{\overline{h}})$. The clipping operator serves instead as a safety mechanism: it forces z into such box, to fix the cases in which $\underline{h}$ and $\overline{h}$ mislocate it. In other words, if the auxiliary models succeed in accomplishing their task, i.e., $z \in \overline{\underline{H}}(x; \theta_{\underline{h}}, \theta_{\overline{h}})$, then the clipping operator becomes neutral, i.e., $z = \overline{\underline{z}} = \text{clip}(z; \underline{h}(x; \theta_{\underline{h}}), \overline{h}(x; \theta_{\overline{h}}))$, and consequently $y = y_{\text{smile}}$. This is why we require these two models to be trainable: ideally, at training time, $\underline{h}$ and $\overline{h}$ learn to bound z, while h learns to place it within the bounds, thus adapting to each other, and reducing the effect of the introduced auxiliary architecture on the predictive performance of the model.

Moreover, to further mitigate the potential accuracy loss caused by the embedded overapproximation, we may employ a *selector* at the input level of the architecture, which checks the validity of the input predicate Q and, in case of a negative outcome, deactivates $\underline{h}$ and $\overline{h}$, and substitutes the head g with an alternative head $g_{\neg}$, specialized to handle the inputs unaffected by the property. In other words, the selector allows to bypass the overapproximation mechanism when this is not necessary. Formally, the selector-augmented SMiLE architecture can be described as:

$$y_{\text{smile}} = f_{\text{smile}}(x; \theta_{\text{smile}}) = \begin{cases} g(\overline{\underline{z}}; \theta_g) \circ \text{clip}(z; \underline{h}(x; \theta_{\underline{h}}), \overline{h}(x; \theta_{\overline{h}})) \circ h(x; \theta_h) & \text{if } Q(x) \\ g_{\neg}(z; \theta_{g_{\neg}}) \circ h(x; \theta_h) & \text{if } \neg Q(x) \end{cases} \tag{4.6}$$

This selection mechanism is particularly suitable for trace properties with a non-trivial input predicate, i.e., those with $Q(x) \neq \top$, where such predicate splits the

input space into two disjoint regions. For relational properties, instead, the selector is generally not appropriate, given that, in these cases, the predicate $Q(x_1, \ldots, x_k)$, rather than the input space $\mathcal{X}$, partitions $\mathcal{X}^k$. Figure 4.3 provides a representation of a SMiLE architecture with an input selector.

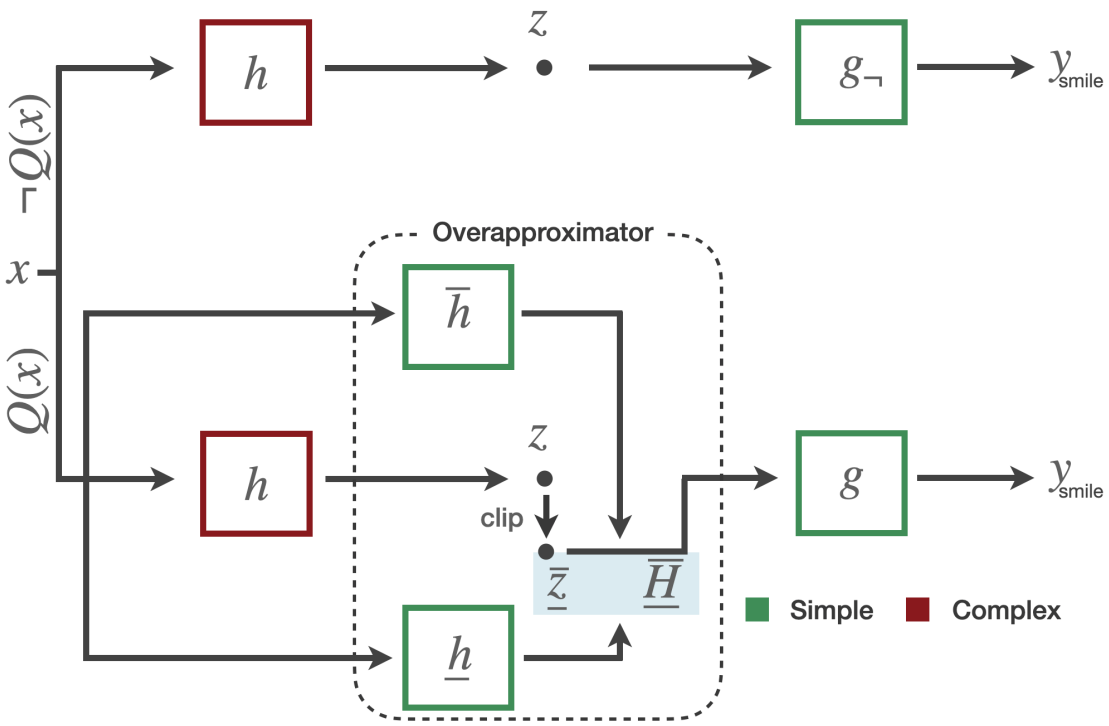


Figure 4.3: A selector-augmented SMiLE architecture.

The convenience of adopting a SMiLE architecture, from a verification and enforcement perspective, arises from the following observations.

1. The auxiliary models $\underline{h}$ and $\overline{h}$ are fully customizable, as long as they remain differentiable. Their structure, in other words, represents a hyperparameter of the SMiLE architecture.

2. These models do not require high expressivity, since they do not need to compute z exactly (which is the role of h), but to only roughly locate it within the latent space. These models, in other words, can be chosen to be arbitrarily simple. This is why, in Figures 4.2 and 4.3, they are depicted in green.

Building upon these two observations, we develop the core idea underlying the SMiLE architecture: while the complex module h, at inference time, is useful to

make the model expressive, at verification or enforcement time, it can be deactivated to decrease the computational burden of both problems.

In the next two sections, we demonstrate how we leverage this idea to design efficient algorithms that enforce trace and relational properties into a SMiLE model, respectively. Note that, in the remainder of this thesis, unless otherwise clarified, we assume a standard SMiLE architecture without a selection mechanism, as described in Equation (4.4) and represented in Figure 4.2. We observe, indeed, that the enforcement algorithms presented in the following sections, although formalized for a selector-free architecture, readily apply also to a selector-equipped one. From now on, moreover, to simplify the notation, we denote a SMiLE model, and its output, simply as f and y, hence we omit the subscript $\cdot_{\text{smile}}$.

4.2 Trace Enforcement

In this section, we present our SMiLE enforcement algorithm for trace properties, that is, the method that we design to solve Equation (3.3), when $k \leqslant 1$.

We address the problem by adopting a Projected Gradient Descent (PGD) scheme [PB+14], a variant of the standard gradient descent algorithm, obtained by pairing each weight update with a counterexample-based weight projection, that is, a mechanism that corrects the weights of the model to restore the feasibility of a given collection of counterexamples, with respect to the given property. Our trace enforcement algorithm, in other words, consists of two main algorithmic principles: a weight *projector* and a counterexample *generator*. We describe these two subroutines in Section 4.2.1 and Section 4.2.2, respectively, before combining them into the global algorithm outlined in Section 4.2.3.

4.2.1 Projector

Lef $f : \mathcal{X} \to \mathcal{Y}$ be a SMiLE architecture. The PGD algorithm pairs the standard update step in SGD (Algorithm 1) with a projection step, defined as:

$$\arg\min_{\hat{\theta}} \|\hat{\theta} - \theta\|_2^2 \tag{4.7a}$$

$$\text{s.t. } R(f(x;\hat{\theta})) \quad \forall x \in \mathcal{X} \colon Q(x) \tag{4.7b}$$

Intuitively, we seek the smallest parameter adjustment that guarantees the satisfaction of the desired property. As discussed in Section 3.3 and Section 3.4, this problem is challenging, due to the universal quantification in Equation (4.7b), as

well as the high non-linearity and large size of modern neural models.

To decrease the difficulty of the problem, we can rely on the SMiLE architecture, that is, we can re-write Equation (4.7) as follows

$$\arg\min_{\hat{\theta}_g} \|\hat{\theta}_g - \theta_g\|_2^2 \tag{4.8a}$$

$$\text{s.t. } R(g(\underline{\overline{z}}; \hat{\theta}_g)) \quad \forall x \in \mathcal{X} \colon Q(x), \forall \underline{\overline{z}} \in \underline{\overline{H}}(x; \theta_{\underline{h}}, \theta_{\overline{h}}) \tag{4.8b}$$

We adopt, in other words, the idea stated in Section 4.1: we relax the arbitrarily complex constraints $z = h(x; \theta_h)$ into the simpler bounding box constraints $\underline{\overline{z}} \in \underline{\overline{H}}(x; \theta_{\underline{h}}, \theta_{\overline{h}})$, which allows to eliminate the dependence of the projection operator on the complex backbone h. More precisely, the projection is only performed in the parameter space of g, while the auxiliary parameters $\theta_{\underline{h}}$ and $\theta_{\overline{h}}$ are solely used to obtain the bounding box $\underline{\overline{H}}$. The adopted relaxation removes the computational flow from the input x to its embedding z and, finally, to its clipped embedding $\underline{\overline{z}}$, turning $\underline{\overline{z}}$ into a free variable of the problem. This, on one hand, gives rise to a conservative projection, potentially leading to an overenforcement of the desired property: in Equation (4.8), property feasibility is enforced not only over the input region satisfying Q, like in the exact projection in Equation (4.7), but also over all corresponding bounding boxes. On the other hand, this makes the projection step considerably simpler to solve, as it now involves only simple functions and much fewer parameters. The problem remains challenging, however, since it contains an infinite number of constraints.

In order to remove the universal quantifier, we adopt a *lazy* (or *delayed*) *constraint generation* approach, consisting in reducing the action of the projection to a finite collection of counterexamples $\mathcal{C}$. Precisely, we replace Equation (4.8) with:

$$\arg\min_{\hat{\theta}_g} \|\hat{\theta}_g - \theta_g\|_2^2 \tag{4.9a}$$

$$\text{s.t. } R(g(\underline{\overline{z}}^*; \hat{\theta}_g)) \quad \forall (x^*, \underline{\overline{z}}^*) \in \mathcal{C} \tag{4.9b}$$

This counterexample-based projection is not only independent of h, but also of $\underline{h}$ and $\overline{h}$: for improved efficiency, and without loss of generality, we restrict our attention to the value of the clipped embedding $\underline{\overline{z}}^*$ of each counterexample $(x^*, \underline{\overline{z}}^*)$, hence ignoring x^*. We encapsulate Equation (4.9) into a routine called TProject, described in Algorithm 2, which returns a feasible $\hat{\theta}_g^*$, by minimally perturbing a given θ_g to restore its feasibility with respect to all counterexamples queued in $\mathcal{C}$, simultaneously. An illustration of a SMiLE architecture at projection time is provided in Figure 4.4, where the removed components are depicted in grey.

Algorithm 2 TPROJECT

1: **Input:** init param θ_g, counterexample queue $\mathcal{C}$
2: $\hat{\theta}_g^* \leftarrow$ solve Equation (4.9) with parameter θ_g and queue $\mathcal{C}$
3: **return** $\hat{\theta}_g^*$

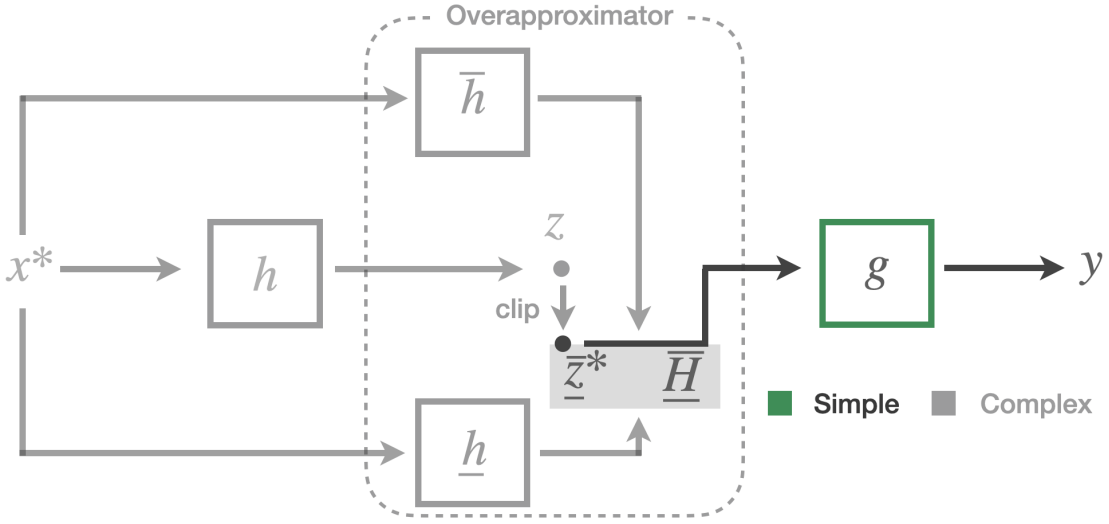


Figure 4.4: A SMiLE architecture at projection time.

4.2.2 Generator

In order to design an efficient counterexample generator for a SMiLE model, to support the projection operator described in the previous section, we replace the verification problem formalized in Equation (3.2) with:

$$\exists x \in \mathcal{X},\ \overline{z} \in \underline{\overline{H}}(x; \theta_{\underline{h}}, \theta_{\overline{h}}) \colon Q(x) \wedge \neg R(g(\overline{z}; \theta_g)) \tag{4.10}$$

In other words, we adopt again the idea stated in Section 4.1: we deactivate h, in order to get rid of its complexity, hence to increase the tractability of the problem. Similarly to the projection, the overapproximation turns the exact verification into a conservative one: if no counterexample is found, then the SMiLE model is guaranteed to satisfy the desired property, but if one is found, then the test is inconclusive, as this may be a false counterexample. In other words, we trade completeness for tractability. The precision of the test, in particular, depends on the size of the bounding box $\underline{\overline{H}}(x; \theta_{\underline{h}}, \theta_{\overline{h}})$, that is, on how tightly the auxiliary models overapproximate the embedding z, for each input x. The quality of the overapproximation, together with an approach designed to increase it, is investigated in Section 5.3.3. An illustration of a SMiLE architecture at projection time

is instead provided in Figure 4.5, where the removed components are depicted in gray.

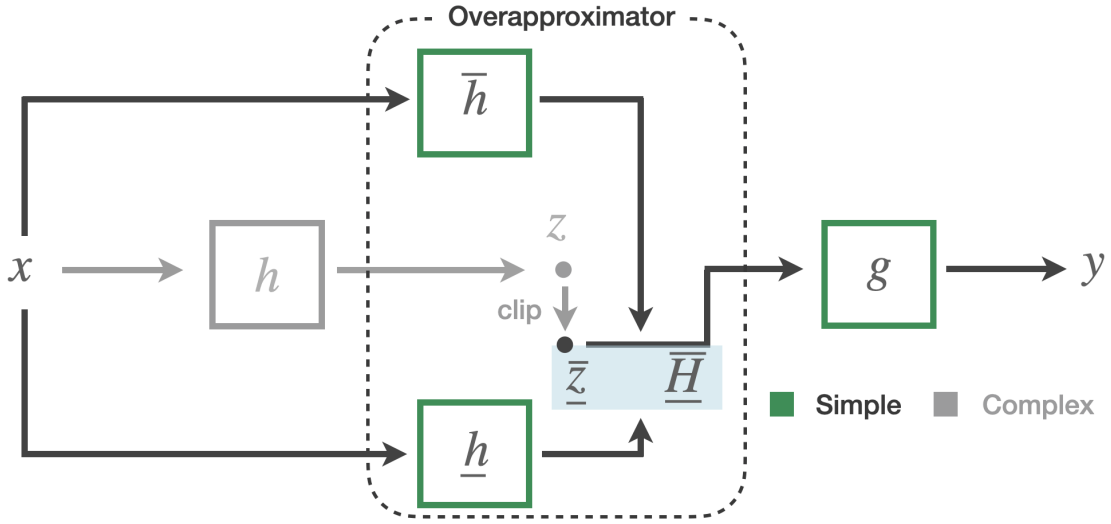


Figure 4.5: A SMiLE architecture at verification time.

Now, in principle, solving Equation (4.10) would be enough to obtain a counterexample. In practice, however, that leaves no control on which kind of counterexample is obtained: if this is too weak, the subsequent projection process could become trivial. Unfortunately, defining "strong" counterexamples is non-trivial, especially without making assumptions on the nature of the property to be satisfied. In particular, since we treat Q and R as logical predicates, they have no associated continuous measure of constraint violation.

We argue that strong counterexamples are those requiring large parameter adjustments for their resolution. Hence, given any pair $(x, \overline{z})$ satisfying Q, we propose to define its strength as a counterexample by means of the ℓ^1-norm:

$$\|\hat{\theta}_g^* - \theta_g\|_1 \tag{4.11}$$

where θ_g is the current weight vector for the g function, and $\hat{\theta}_g$ is defined as:

$$\hat{\theta}_g^* = \arg\min_{\hat{\theta}_g} \left\{ \|\hat{\theta}_g - \theta_g\|_2^2 \text{ s.t. } R(g(\overline{z}; \hat{\theta}_g)) \right\} \tag{4.12}$$

For a pair $(x, \overline{z})$ that already satisfies the property, we have $\hat{\theta}_g^* = \theta_g$, while we have $\|\hat{\theta}_g^* - \theta_g\|_1 > 0$ for a true counterexample. In other words, we are using our

projection operator as a mechanism to compute a counterexample strength. In particular, in Equation (4.11), we use the ℓ^1-norm, rather than the ℓ^2, for its lower computational complexity.

When used into a counterexample generation process, however, this mechanism leads to an intractable bi-level optimization problem. This is why we reduce Equation (4.12) to a translation-based projection, that is, we restrict the action of the projection to the offset:

$$g(\overline{\underline{z}}; \hat{\theta}_g) = \theta_{g,1:n_{d-1}} \overline{\underline{z}} + (\theta_{g,0} + \nu) = g(\overline{\underline{z}}; \theta_g) + \nu \tag{4.13}$$

where $\theta_{g,1:n_{d-1}} \in \mathbb{R}^{n_d \times n_{d-1}}$, while $\theta_{g,0}, \nu \in \mathbb{R}^{n_d}$. We now frame the problem of generating a strong counterexample as that of finding a pair $x, \overline{\underline{z}}$ whose resolution requires the largest translation:

$$\begin{aligned}
&\arg\max_{x,\overline{\underline{z}}} \|\nu^*\|_1 && \text{(4.14a)}\\
&\text{s.t.: } x \in \mathcal{X} && \text{(4.14b)}\\
&\quad Q(x) && \text{(4.14c)}\\
&\quad \overline{\underline{z}} \geqslant \underline{h}(x; \theta_{\underline{h}}) && \text{(4.14d)}\\
&\quad \overline{\underline{z}} \leqslant \underline{h}(x; \theta_{\underline{h}}) + M(1-p) && \text{(4.14e)}\\
&\quad \overline{\underline{z}} \leqslant \overline{h}(x; \theta_{\overline{h}}) + Mp && \text{(4.14f)}\\
&\quad \nu^* = \arg\min_{\nu} \{\|\nu\|_2^2 \text{ s.t. } R(g(\overline{\underline{z}}; \theta_g) + \nu)\} && \text{(4.14g)}
\end{aligned}$$

where Equations (4.14d) to (4.14f), for a sufficiently large $M > 0$, linearize

$$\underline{h}(x; \theta_{\underline{h}}) \leqslant \overline{\underline{z}} \leqslant \max(\underline{h}(x; \theta_{\underline{h}}), \overline{h}(x; \theta_{\overline{h}})) \tag{4.15}$$

through big-M constraints, modeling eventual bounding box degeneracy, occurring when, along any ith-axis, the two bounds flip, i.e., $\overline{h}(x; \theta_{\overline{h}})_i < \underline{h}(x; \theta_{\underline{h}})_i$. Precisely, if the box is non-degenerate, then any choice of p_i ensures that $\overline{\underline{z}}_i$ is chosen in between the bounds, i.e., $\underline{h}(x; \theta_{\underline{h}})_i \leqslant \overline{\underline{z}}_i \leqslant \overline{h}(x; \theta_{\overline{h}})_i$, while in case of degeneracy, the only choice for the optimization engine to prevent infeasibility is to fix $p_i = 1$, which in turn fixes $\overline{\underline{z}}_i = \underline{h}(x; \theta_{\underline{h}})_i$, consistently with the behavior of the model specified by the clipping operator in Equation (4.3), hence correctly modeling the box constraints in both cases. Moreover, $\|\nu^*\|_1$ and $\|\nu\|_2^2$ are equivalent to $\|\hat{\theta}^*_{g,0} - \theta_{g;0}\|_1$ and $\|\hat{\theta}_{g,0} - \theta_{g,0}\|_2^2$, since we are restricted to translation.

While translation alone might be insufficient to solve the projection from Equation (4.9), here we are interested in resolving a single pair $x, \overline{\underline{z}}$. This can always be done by translation if the R predicate defines a non-empty space. Note that

Equation (4.14g) is significantly easier to solve than Equation (4.12), given that, in many cases, it admits a closed-form solution, as shown in the property-specific groundings discussed in Sections 5.3.1 and 5.5.1. Similarly to the projector, we encapsulate Equation (4.14) into a routine called TGenerate, described in Algorithm 3, which solves the problem for a given parameterization $\theta_g, \theta_{\underline{h}}, \theta_{\overline{h}}$, and returns an optimal counterexample $(x^*, \underline{\overline{z}}^*)$, along with the optimal objective value ν^{**}, guaranteeing property feasibility if and only if $\|\nu^{**}\|_1 = 0$.

Algorithm 3 TGenerate

1: **Input:** parameters $\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g$
2: $x^*, \underline{\overline{z}}^*, \nu^{**} \leftarrow$ solve Equation (4.14)
3: **return** $x^*, \underline{\overline{z}}^*, \nu^{**}$

The formulation from Equation (4.14) is partially heuristic in nature, therefore further simplifications can be done for specific properties if they bring computational or quality advantages. In any case, the approach is fairly general, built on a solid rationale, and should result in strong counterexamples.

4.2.3 Enforcer

We now combine the subroutines described above into the main algorithm designed to enforce trace properties into SMiLE models. The procedure is outlined in Algorithm 4, where we treat the variables as iterates, and we omit the superscript $\cdot^*$ to keep the notation clean. The algorithm initializes an empty queue to store counterexamples, and then proceeds through three phases: *pretraining*, *training* and *posttraining*.

Pretraining. This phase provides a property-agnostic warm start to the algorithm through a standard SGD scheme: at each iteration, the algorithm aggregates the provided sample-level loss L across $\mathcal{D}$, to obtain the data-based model-level loss $L_{\mathcal{D},\text{task}}$, and then performs a standard gradient step, by moving the weights θ to the opposite direction of the loss gradient.

Training. This phase represents the main property-aware training, in which the standard weight update, performed to maximize model accuracy, is alternated with the counterexample-based weight projection, executed to enforce property feasibility. Precisely, at each iteration, we first perform a standard gradient step, similarly to the pretraining stage, then we verify the current weights by running TGenerate, which returns $(x, \underline{\overline{z}})$ and ν. We thus check if $\|\nu\|_1 > 0$, to find out whether

the property is violated by the current model parameterization, that is, if $(x, \overline{z})$ is a valid counterexample. In this case, we first update the counterexample queue $\mathcal{C}$ via a first-in-first-out procedure FIFO, which pulls the oldest counterexample and pushes the most recent one, if $\mathcal{C}$ has reached the maximum capacity q. We then correct the head weights θ_g by executing TPROJECT, which simultaneously restores the feasibility of the counterexamples into $\mathcal{C}$.

Posttraining. In the last phase, we run a final accuracy-agnostic enforcement of the property, that is, we deactivate the parameter update driven by the training loss, hence we only repeat the generation-projection loop, in order to eliminate any residual violation. The algorithm terminates by returning the optimized weights θ, together with the dual objective bound $\overline{\nu}$ from the last counterexample generation, which provides an upper bound to property violation, useful when convergence to zero violation is not achieved.

According to the taxonomy proposed in Section 3.4, our trace enforcement falls among the class of projection-based methods.

4.3 Relational Enforcement

In this section, we present our SMiLE enforcement algorithm for relational properties. More specifically, although theoretically applicable to the general setting defined in Equation (3.1), in this work we formulate our method for a real-valued neural network $f\colon \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} = \times_{i=1}^{n}[l_i, u_i] \subseteq \mathbb{R}^n$ and $l \leqslant u$, and 2-arity properties of the form

$$\forall x', x'' \in \mathcal{X}\colon \overbrace{\underline{\delta} \leqslant x' - x'' \leqslant \overline{\delta}}^{Q(x',x'')} \implies \overbrace{\underline{\epsilon} \leqslant f(x';\theta) - f(x'';\theta) \leqslant \overline{\epsilon}}^{R(f(x';\theta),f(x'';\theta))} \tag{4.16}$$

for given input and output bounds $\underline{\delta}, \overline{\delta} \in \mathbb{R}^n$ and $\underline{\epsilon}, \overline{\epsilon} \in \mathbb{R}$. The most common relational properties targeted in the literature, such as the ones reported in Table 3.3, can be derived from Equation (4.16) by configuring its parameters, as shown in Table 4.1, where $\delta, \epsilon \geqslant 0$, $M > 0$ is a sufficiently large value used to relax unnecessary bounds, while $\mathcal{S}, \mathcal{S}^c \subseteq [n]$ 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 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

Algorithm 4 TEnforce

Input: init params θ, train data $\mathcal{D}$, loss L, queue capacity q, SGD params
$\mathcal{C} \leftarrow$ empty queue
Pretraining:
for pretraining step **do**
  $L_{\mathcal{D},\text{task}} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(y, f(x;\theta))$
  $\theta \leftarrow \theta - \eta_{\text{task}} \nabla_\theta L_{\mathcal{D},\text{task}}$
end for
Training:
for training step **do**
  $L_{\mathcal{D},\text{task}} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(y, f(x;\theta))$
  $\theta \leftarrow \theta - \eta_{\text{task}} \nabla_\theta L_{\mathcal{D},\text{task}}$
  $x, \overline{z}, \nu \leftarrow \textsc{Generate}(\theta_g, \theta_{\underline{h}}, \theta_{\overline{h}})$
  if $\|\nu\|_1 > 0$ **then**
    $\mathcal{C} \leftarrow \text{FIFO}(\mathcal{C}, \overline{z}, q)$
    $\theta_g \leftarrow \textsc{Project}(\theta_g, \mathcal{C})$
  end if
end for
Posttraining:
for posttraining step **do**
  $x, \overline{z}, \nu \leftarrow \textsc{Generate}(\theta_g, \theta_{\underline{h}}, \theta_{\overline{h}})$
  $\overline{\nu} \leftarrow$ dual objective bound
  if $\|\nu\|_1 > 0$ **then**
    $\mathcal{C} \leftarrow \text{FIFO}(\mathcal{C}, \overline{z}, q)$
    $\theta_g \leftarrow \textsc{Project}(\theta_g, \mathcal{C})$
  end if
end for
return $\theta, \overline{\nu}$

decrease (↑) or increase (↓). In the remainder of this thesis, we refer to the properties defined in Equation (4.16) simply as relational properties.

Global Relational Properties

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

Table 4.1: Configuration of δ and ϵ for different properties.

Even though our trace enforcement approach readily applies to relational properties, in this setting it proves completely inadequate, due to several structural limitations clarified in the remainder of this section. To enforce these properties, we therefore reengineer the method presented in the previous section, to obtain a more general, scalable, and stable relational variant. In line with Section 4.2, we introduce the two reengineered algorithmic components, i.e., the projector and the generator, in Section 4.3.1 and Section 4.3.2, respectively, then we combine them into the main relational algorithm described in Section 4.3.3.

4.3.1 Projector

The projection operation described in Equation (4.9) can be readily adapted to relational properties as follows:

$$\arg\min_{\hat{\theta}_g} \|\hat{\theta}_g - \theta_g\|_2^2 \tag{4.17a}$$

$$\text{s.t. } \underline{\epsilon} \leqslant g(\underline{z}'^*; \hat{\theta}_g) - g(\underline{z}''^*; \hat{\theta}_g) \leqslant \overline{\epsilon} \quad \forall (x'^*, x''^*, \underline{z}'^*, \underline{z}''^*) \in \mathcal{C} \tag{4.17b}$$

where $(x'^*, x''^*, \underline{z}'^*, \underline{z}''^*) \in \mathcal{X} \times \mathcal{X} \times \overline{\underline{H}}(x'^*, ; \theta_{\underline{h}}, \theta_{\overline{h}}) \times \overline{\underline{H}}(x''^*; \theta_{\underline{h}}, \theta_{\overline{h}})$ is a SMiLE counterexample for relational properties, that is, a pair of inputs satisfying the input bounds, whose clipped embeddings are in the corresponding boxes, and whose outputs violate the output bounds. As observed in preliminary experiments, however, projecting only on g proves insufficient for relational properties. This is because, under the influence of the accuracy-driven loss, without an auxiliary-aware projector, the overapproximation tends to remain large, to allow the more expressive backbone to freely operate in between the bounds, as demonstrated in Section 5.3.3, in particular in Figure 5.5. For relational properties, this consistently leads to large violations, given that, in this setting, strong counterexamples

can always be produced by picking their embeddings at opposite corners of their corresponding boxes. Figure 4.6 illustrates an example of two embeddings maximizing the variation of g, which increases along the yellow arrow, representing the gradient of g with respect to its input. In turn, to resolve such high violations, the auxiliary-agnostic projector forces g to collapse into an almost constant function, this being the only way to ensure that such remote embeddings are propagated within the output bounds.

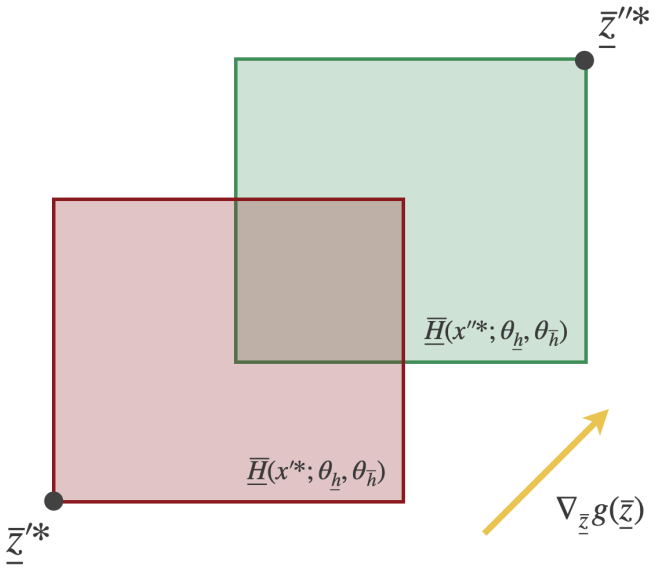


Figure 4.6: Two embeddings maximizing the variation of g.

One way to prevent such degenerate cases is to incorporate $\underline{h}$ and $\overline{h}$ into the projection step, thereby distributing the burden of the enforcement also onto these components, rather than relying solely on g. Precisely:

$$\arg\min_{\hat{\theta}_{\underline{h}}, \hat{\theta}_{\overline{h}}, \hat{\theta}_g} \|\hat{\theta}_{\underline{h}} - \theta_{\underline{h}}\|_2^2 + \|\hat{\theta}_{\overline{h}} - \theta_{\overline{h}}\|_2^2 + \|\hat{\theta}_g - \theta_g\|_2^2 \tag{4.18a}$$

$$\text{s.t. } \underline{\overline{z}}' = \max(\underline{h}_{\hat{\theta}_{\underline{h}}}(x'^*), \min(\underline{\overline{z}}'^*, \overline{h}_{\hat{\theta}_{\overline{h}}}(x'^*)) \qquad \forall (x'^*, x''^*, \underline{\overline{z}}'^*, \underline{\overline{z}}''^*) \in \mathcal{C} \tag{4.18b}$$

$$\underline{\overline{z}}'' = \max(\underline{h}_{\hat{\theta}_{\underline{h}}}(x''^*), \min(\underline{\overline{z}}''^*, \overline{h}_{\hat{\theta}_{\overline{h}}}(x''^*)) \qquad \forall (x'^*, x''^*, \underline{\overline{z}}'^*, \underline{\overline{z}}''^*) \in \mathcal{C} \tag{4.18c}$$

$$\underline{\epsilon} \leqslant g(\underline{\overline{z}}'; \hat{\theta}_g) - g(\underline{\overline{z}}''; \hat{\theta}_g) \leqslant \overline{\epsilon} \qquad \forall (x'^*, x''^*, \underline{\overline{z}}'^*, \underline{\overline{z}}''^*) \in \mathcal{C} \tag{4.18d}$$

Unfortunately, Equation (4.18) presents several computational challenges. Equations (4.18b) and (4.18c) involve the encoding of the auxiliary models $\underline{h}$ and $\overline{h}$, which although assumed simple, may in general be non-linear, as well as min and

max operations, which require the introduction of binary variables to be modeled linearly, hence increasing the non-convexity degree of the problem. Equation (4.18d), moreover, involves products of variables of the form $\overline{\underline{z}}_i \cdot \theta_{g,i}$, which makes the problem quadratically constrained. In particular, preliminary experiments revealed that solving Equation (4.18) to optimality is intractable.

To practically execute the projection, we resort to a heuristic Dual Gradient Ascent approach, where property violation (4.18d) is incorporated as a penalty term into the objective, thus is treated as a soft constraint, rather than a hard one. Switching to a heuristic gradient-based approach significantly speeds up the projection process, but requires the objective to be differentiable. To this aim, we observe that, due to the linearity of g, the embedding components $\overline{\underline{z}}'^*$ and $\overline{\underline{z}}''^*$, for any locally optimal counterexample, always lie at one vertex of their bounding boxes. Based on this observation, we enable backpropagation by applying an *abstraction step* to the counterexample embeddings, that is, we represent them through the associated active box constraints:

$$\forall i \in [n_{d-1}],\ \overline{\underline{z}}_i^* = \underline{h}(x^*; \theta_{\underline{h}})_i \vee \overline{\underline{z}}_i^* = \overline{h}(x^*; \theta_{\overline{h}})_i \tag{4.19}$$

The abstract propagation routine PROPAGATE, outlined in Algorithm 5, manages this step.

Algorithm 5 PROPAGATE

Input: $\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, x^*, \overline{\underline{z}}^*$
$\underline{I} \leftarrow \{i : \overline{\underline{z}}_i^* = \underline{h}(x^*; \theta_{\underline{h}})_i\}$
$\overline{I} \leftarrow \underline{I}^c$
$\overline{\underline{z}} = (\underline{h}(x^*; \theta_{\underline{h}})_{\underline{I}}, \overline{h}(x^*; \theta_{\overline{h}})_{\overline{I}})$
$y = g(\overline{\underline{z}}, \theta_g)$
return y

Algorithm 6 provides the pseudocode of the projection routine, RPROJECT, adopted to resolve a single counterexample, which proves sufficient in this setting, as demonstrated in the experimental section. The algorithm initializes the weights to the provided ones, and a parameter λ_{prop} to 0, before starting the main loop. In each iteration, it propagates the given counterexample through the PROPAGATE routine, to obtain the outputs y', y''. Then, it computes the task loss L_{task} as in Equation (4.18), and the property one L_{prop} as the violation of either the lower bound $\underline{\epsilon}$, or the upper bound $\overline{\epsilon}$, if such violation is positive, before aggregating the two terms into L_{tot}. Finally, the algorithm performs a primal gradient step,

in which it moves $\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g$ in the opposite direction of the gradient to minimize the loss, and a dual ascent step, in which it moves λ_{prop} in the same direction of the gradient to increase the violation penalty. Results from classical Lagrangian methods ensure that, under mild conditions, the process asymptotically converges to a local optimum. Note that, in Algorithm 6, the two loss terms, L_{task} and L_{prop}, do not depend on data, but only on the initial parameters and the provided counterexample.

Algorithm 6 RProject

Input: init param $\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g$, counterexample $(x'^*, x''^*, \underline{z}'^*, \overline{z}''^*)$, SGD params
$\hat{\theta}_{\underline{h}} \leftarrow \theta_{\underline{h}}, \hat{\theta}_{\overline{h}} \leftarrow \theta_{\overline{h}}, \hat{\theta}_g \leftarrow \theta_g, \lambda_{\text{prop}} \leftarrow 0$
for projection step **do**
  $y' \leftarrow \textsc{Propagate}(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, x'^*, \underline{z}'^*)$
  $y'' \leftarrow \textsc{Propagate}(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, x''^*, \underline{z}''^*)$
  $L_{\text{task}} \leftarrow \|\hat{\theta}_{\underline{h}} - \theta_{\underline{h}}\|_2^2 + \|\hat{\theta}_{\overline{h}} - \theta_{\overline{h}}\|_2^2 + \|\hat{\theta}_g - \theta_g\|_2^2$
  $L_{\text{prop}} \leftarrow \max(0, \max(\underline{\varepsilon} - y' + y'', y' - y'' - \overline{\varepsilon}))$
  $L_{\text{tot}} \leftarrow L_{\text{task}} + \lambda_{\text{prop}} L_{\text{prop}}$
  $\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g \leftarrow \theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g - \eta_{\text{task}} \nabla_{\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g} L_{\text{tot}}$
  $\lambda_{\text{prop}} \leftarrow \lambda_{\text{prop}} + \eta_{\text{prop}} \nabla_{\lambda_{\text{prop}}} L_{\text{tot}}$
end for
return $\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g$

4.3.2 Generator

A verification problem for the property defined in Equation (4.16) can be formulated as follows:

$$\exists x', x'' \in \mathcal{X} : \tag{4.20a}$$

$$\underline{\delta} \leqslant x' - x'' \leqslant \overline{\delta} \wedge (f(x';\theta) - f(x'';\theta) < \underline{\epsilon} \vee f(x';\theta) - f(x'';\theta) > \overline{\epsilon}) \tag{4.20b}$$

The problem, in other words, consists in searching for a pair of inputs (x', x'') satisfying the input bounds, whose corresponding outputs violate the output one.

To design an efficient counterexample generator for relational properties, we leverage again the SMiLE architecture, that is, we disregard the embedding h and rely solely on the overapproximation provided by $\underline{h}, \overline{h}$. Precisely, we replace

Equation (4.20) with

$$\exists x', x'' \in \mathcal{X}, \underline{\overline{z}}' \in \underline{\overline{H}}(x'; \theta_{\underline{h}}, \theta_{\overline{h}}), \underline{\overline{z}}'' \in \underline{\overline{H}}(x''; \theta_{\underline{h}}, \theta_{\overline{h}}) : \tag{4.21a}$$

$$\underline{\delta} \leqslant x' - x'' \leqslant \overline{\delta} \wedge (g(\underline{\overline{z}}'; \theta_g) - g(\underline{\overline{z}}''; \theta_g) < \underline{\epsilon} \vee g(\underline{\overline{z}}'; \theta_g) - g(\underline{\overline{z}}''; \theta_g) > \overline{\epsilon}) \tag{4.21b}$$

Removing the link h from the input x to its embedding z requires searching not only for a pair of inputs, but also for a pair of corresponding embeddings, so that a counterexample for the relaxed verification consists in a tuple of the form $(x', x'', \underline{\overline{z}}', \underline{\overline{z}}'')$, as already remarked in the previous section.

In particular, similarly to the trace generator, also in this case we turn the search problem stated in Equation (4.21) into an optimization one, which we formulate as follows:

$$\arg\max_{x', x'', \underline{\overline{z}}', \underline{\overline{z}}''} \nu \tag{4.22a}$$

$$\text{s.t. } l \leqslant x', x'' \leqslant u \tag{4.22b}$$

$$\underline{\delta} \leqslant x' - x'' \leqslant \overline{\delta} \tag{4.22c}$$

$$\underline{\overline{z}}' \geqslant \underline{h}(x'; \theta_{\underline{h}}) \tag{4.22d}$$

$$\underline{\overline{z}}' \leqslant \underline{h}(x'; \theta_{\underline{h}}) + M(1 - p') \tag{4.22e}$$

$$\underline{\overline{z}}' \leqslant \overline{h}(x'; \theta_{\overline{h}}) + Mp' \tag{4.22f}$$

$$\underline{\overline{z}}'' \geqslant \underline{h}(x''; \theta_{\underline{h}}) \tag{4.22g}$$

$$\underline{\overline{z}}'' \leqslant \underline{h}(x''; \theta_{\underline{h}}) + M(1 - p'') \tag{4.22h}$$

$$\underline{\overline{z}}'' \leqslant \overline{h}(x''; \theta_{\overline{h}}) + Mp'' \tag{4.22i}$$

$$y' = g(\underline{\overline{z}}'; \theta_g) \tag{4.22j}$$

$$y'' = g(\underline{\overline{z}}''; \theta_g) \tag{4.22k}$$

$$\nu \leqslant y' - y'' - \overline{\epsilon} + Mb \tag{4.22l}$$

$$\nu \leqslant -y' + y'' + \underline{\epsilon} + M(1 - b) \tag{4.22m}$$

$$x', x'' \in \mathbb{R}^{B^c} \times \{0,1\}^B, \underline{\overline{z}}', \underline{\overline{z}}'' \in \mathbb{R}^n,\ y', y'' \in \mathbb{R} \tag{4.22n}$$

$$b \in \{0,1\}, p \in \{0,1\}^{n_{d-1}},\ \nu \in \mathbb{R}_{>0} \tag{4.22o}$$

where $M > 0$ is a fixed big-M value, while $B, B^c \subseteq [m]$ respectively denote the set of binary variables and its complement, necessary when the learning task involves binary or one-hot encoded features. Equations (4.22d) to (4.22f) and Equations (4.22g) to (4.22i) model the eventually degenerate box constraints as big-M constraints, while Equations (4.22j) and (4.22k) represent property violation, computed as

$$\max(0, \max(\underline{\epsilon} - y' + y'', y' - y'' - \overline{\epsilon})) \tag{4.23}$$

and modeled again via big-M constraints. Note that ν in Equation (4.22) and L_{prop} in Algorithm 6 are defined identically, but we prefer to keep the notation separate: ν is used when property violation is maximized at generation time, while L_{prop} when it is minimized at projection or training time.

Solving Equation (4.22) is dramatically easier than solving Equation (4.20), given that, in the former, the complex backbone h is ignored. However, the problem can still be challenging, especially when the auxiliary models $\underline{h}, \overline{h}$ are moderately complex, or when 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 Equation (4.22) 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 repeat for a maximum number of iterations, after that we stop. In the latter case, since we use Mathematical Programming as our solution technology, we can still provide a bound on the maximum violation level for the network. The procedure is described in Algorithm 7.

Algorithm 7 RGenerate

Input: parameters $\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g$, timeout τ
$(x'^*, x''^*, \underline{z}'^*, \underline{z}''^*) \leftarrow$ null
for each generation step **do**
  $(x'^*, x''^*, \underline{z}'^*, \underline{z}''^*) \leftarrow$ resume solve eq. (4.22) with timout τ
  S $\leftarrow$ solver termination status, , $\overline{\nu}^* \leftarrow$ best dual bound
  if $(x'^*, x''^*, \underline{z}'^*, \underline{z}''^*) \neq$ null **then**
    break
  end if
  if S = infeasible **then**
    break
  end if
  $\tau \leftarrow 2 \cdot \tau$
end for
return $x'^*, x''^*, \underline{z}'^*, \underline{z}''^*, \overline{\nu}^*, S$

4.3.3 Enforcer

Similarly to our trace enforcer, we now aggregate the two routines described above, projector and generator, into the relational enforcer outlined in Algorithm 11, where we simplify the notation by omitting the superscript $\cdot^*$ for the iterates. The algorithm consists of three phases: *pretrainig*, *training* and *posttraining*. Figure 4.7 depicts the algorithm outcome at the end of each phase for a synthetic example, where we train a SMiLE model, with latent dimension $n_{d-1} = 1$ and head $g = \mathrm{id}$, to approximate the function $y = x^3$, while satisfying the robustness property

$$\forall x', x'' \in \mathbb{R}, |x' - x''| \leqslant 0.1 \implies |f(x') - f(x'')| \leqslant 0.5 \cdot \max_{x \in \mathbb{R}, \delta \leqslant 0.1} |x^3 - (x+\delta)^3| \quad (4.24)$$

i.e., we force the model to be 0.5 times more robust than the task itself.

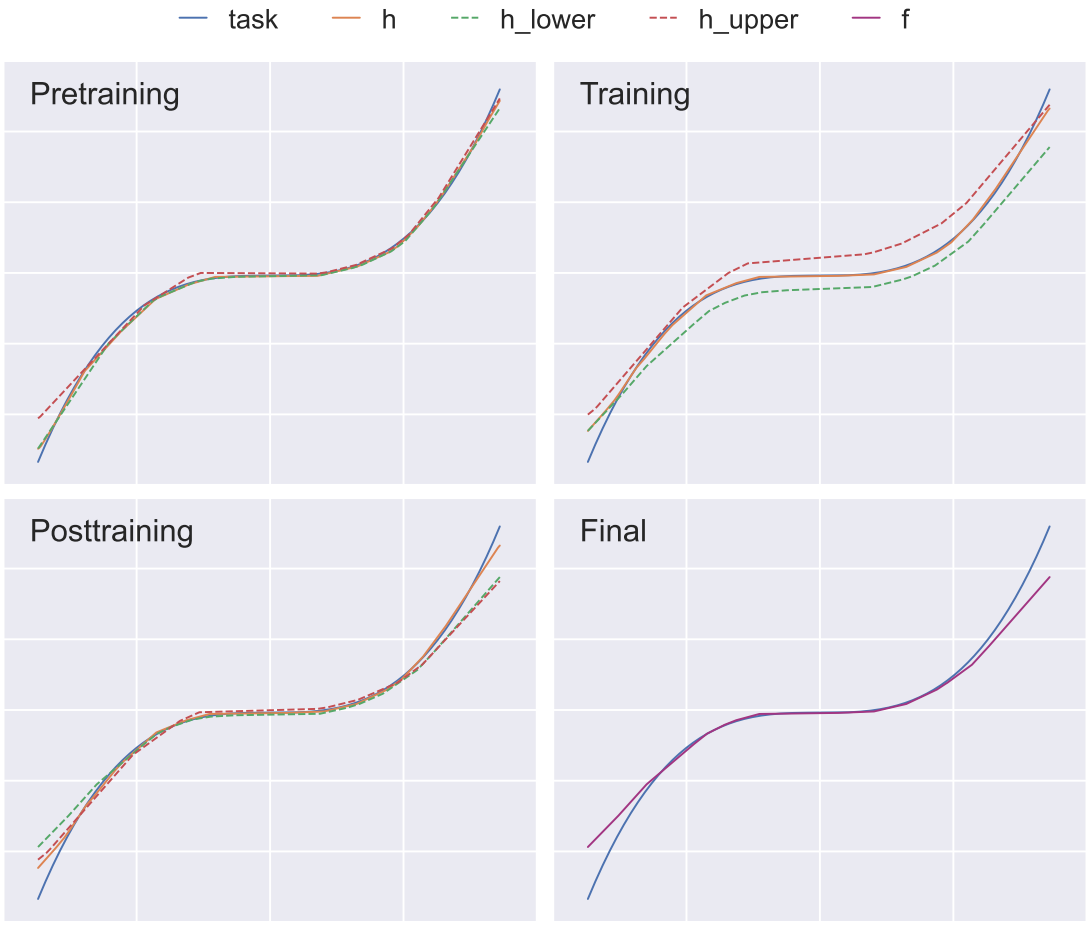


Figure 4.7: The training evolution of a SMiLE model for a synthetic example.

Pretraining. The purely accuracy-driven pretraining used in the trace enforcer tends to produce large boxes, that is, to move the auxiliary models apart, in order to allow the backbone to freely operate in between them, as demonstrated in Section 5.3.3, in particular in Figure 5.5. Such boxes, moreover, may be non-homogeneous across the input space, being wide for some inputs and tight for others. While irrelevant in the trace enforcement, where the projector is auxiliary-agnostic, in the relational one the auxiliary-aware projector, to tighten the box in wide input regions, where high violations arise, often causes degeneracy (i.e., bound flipping) in already narrow ones. When this happens, the semantics of the clipping operator prevents most gradient components from being backpropagated, causing catastrophic training failure. To homogenize the initial overapproximation, we warm-start the algorithm through a standard gradient descent against the custom loss PretrainLoss, outlined in Algorithm 8. In particular, the employed pretraining loss features two terms, $L_{\mathcal{D},\text{task}}$ and $L_{\mathcal{D},\text{box}}$, both obtained by aggregating the provided sample-level loss L on the data $\mathcal{D}$: the former maximizes the accuracy of *all* possible output pathways in the SMiLE computational graph, i.e., $g \circ h$, $g \circ \underline{h}$, and $g \circ \overline{h}$, while the latter penalizes overapproximation degeneracy. As shown in Figure 4.7, at the end of pretraining loop, all output pathways provide, for the considered example, similar results, and no significant flipping occurs.

Algorithm 8 PretrainLoss

Input: init params θ, train data $\mathcal{D}$, loss L, box penalty λ_{box}
$L_{\mathcal{D},f} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(y, f(x;\theta))$
$L_{\mathcal{D},\underline{h}} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(y, g(\underline{z};\theta_g) \circ \underline{h}(x;\theta_{\underline{h}}))$
$L_{\mathcal{D},\overline{h}} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(y, g(\overline{z};\theta_g) \circ \overline{h}(x;\theta_{\overline{h}}))$
$L_{\mathcal{D},h} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(y, g(z;\theta_g) \circ h(x;\theta_h))$
$L_{\mathcal{D},\overline{\text{box}}} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \max(0, h(x;\theta_h) - \overline{h}(x,\theta_{\overline{h}}))$
$L_{\mathcal{D},\underline{\text{box}}} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \max(0, \underline{h}(x;\theta_{\underline{h}}) - h(x;\theta_h))$
$L_{\mathcal{D},\text{task}} \leftarrow L_{\mathcal{D},f} + L_{\mathcal{D},\underline{h}} + L_{\mathcal{D},\overline{h}} + L_{\mathcal{D},h}$
$L_{\mathcal{D},\text{box}} \leftarrow L_{\mathcal{D},\overline{\text{box}}} + L_{\mathcal{D},\underline{\text{box}}}$
return $L_{\mathcal{D},\text{task}} + \lambda_{\text{box}} L_{\mathcal{D},\text{box}}$

Training. For the main training phase, we rely on dual ascent, to reduce both estimation errors and property violations. In each iteration, we first run Resolved, described in Algorithm 9, to check whether the current counterexample has been resolved, in which case we generate a new one by means of TGener-

ATE. Then, we use this counterexample to compute the multi-component loss function TRAINLOSS, described in Algorithm 10 and incorporating: 1) a data-dependent term $L_{\mathcal{D},\text{task}}$ accounting for model accuracy; 2) a data-dependent term $L_{\mathcal{D},\text{box}}$ penalizing box degeneracy, analogous to the one used in PRETRAINLOSS; 3) a data-independent term $L_{\mathcal{D},\text{prop}}$ representing the degree of violation for the incumbent counterexample, analogous to the one used in RPROJECT. Note that the multiplier λ_{box} for $L_{\mathcal{D},\text{box}}$ is fixed, while the one λ_{prop} for L_{prop} is trainable. Moreover, we observe that, in TRAINLOSS, L_{prop} does not depend on data, while $L_{\mathcal{D},\text{task}}$ and $L_{\mathcal{D},\text{box}}$ do. At this point, we perform a primal gradient descent step, followed by a dual gradient ascent one. In the former, we differentiate with respect to the parameters θ and move opposite to the gradient, to reduce the loss value. In the latter, we differentiate with respect to λ_{prop} and move in the same direction of the gradient, to increase the penalty in case of violations. Results from Lagrangian theory ensure asymptotical convergence to a local optimum. Figure 4.7 depicts how, during this phase, the overapproximation defined by $\underline{h}, \overline{h}$ expands, allowing the backbone to more accurately approximate the underlying function in the regions where its shape is compatible with the enforced property.

Algorithm 9 RESOLVED

Input: counterexample $(x'^*, x''^*, \underline{z}'^*, \underline{z}''^*)$
if $(x'^*, x''^*, \underline{z}'^*, \underline{z}''^*) \neq$ null **then**
  $y' \leftarrow$ PROPAGATE$(x'^*, \underline{z}'^*)$
  $y'' \leftarrow$ PROPAGATE$(x''^*, \underline{z}''^*)$
  $L_{\text{prop}} \leftarrow \max(0, \max(\underline{\varepsilon} - y' + y'', y' - y'' - \bar{\varepsilon}))$
  if $L_{\text{prop}} > 0$ **then**
    return False
  end if
end if
return True

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. In this phase, we iteratively execute TGENERATE and TPROJECT until no more counterexample can be found, or the iteration limit is reached. Upon termination, the loop returns

Algorithm 10 TrainLoss

Input: init params θ, train data $\mathcal{D}$, loss L, box penalty λ_{box}, violation penalty λ_{prop}, counterexample $(x'^*, x''^*, \underline{z}'^*, \underline{z}''^*)$
$L_{\mathcal{D},\text{task}} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(y, f(x;\theta))$
$L_{\mathcal{D},\overline{\text{box}}} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \max(0, h(x;\theta_h) - \overline{h}(x,\theta_{\overline{h}}))$
$L_{\mathcal{D},\underline{\text{box}}} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \max(0, \underline{h}(x;\theta_{\underline{h}}) - h(x;\theta_h))$
$L_{\mathcal{D},\text{box}} \leftarrow L_{\mathcal{D},\overline{\text{box}}} + L_{\mathcal{D},\underline{\text{box}}}$
if $(x'^*, x''^*, \underline{z}'^*, \underline{z}''^*) \neq \text{null}$ **then**
    $y' \leftarrow \text{Propagate}(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, x'^*, \underline{z}'^*)$
    $y'' \leftarrow \text{Propagate}(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, x''^*, \underline{z}''^*)$
    $L_{\text{prop}} \leftarrow \max(0, \max(\underline{\varepsilon} - y' + y'', y' - y'' - \bar{\varepsilon}))$
else
    $L_{\text{prop}} \leftarrow 0$
end if
return $L_{\mathcal{D},\text{task}} + \lambda_{\text{box}} L_{\text{box}} + \lambda_{\text{prop}} L_{\text{prop}}$

the model weights θ, together with the certified upper bound on property violation $\overline{\nu}$, which guarantees property satisfaction if it is zero, while provides a guaranteed upper bound on property violation. As shown in Figure 4.7, the posttraining phase tightens again the overapproximation to prevent violations.

The relational enforcer described above differs from its trace counterpart not only on the design of its main algorithmic components, i.e., projector and generator, but also on the way in which these are combined. In particular, while the trace enforcer invokes the projector immediately after each counterexample generation, the relational one performs a standalone projection only in posttraining, to remove residual violations, while at training time, it promotes property feasibility through regularization. This shift from hard to soft enforcement during training stems from the fact that, while the trace projector is MIP-based (Section 4.2.1), the relational one is, like the training procedure itself, gradient-based (Section 4.3.1). This alignment allows the relational projector to be seamlessly merged into the training process, rather than executed as a nested subroutine, which reduces the typical zig-zag behavior of PGD, accelerates convergence, and significantly decreases generator calls. In other words, while our trace enforcer, according to the taxonomy proposed in Section 3.4, falls among the class of projection-based methods, our relational enforcer can be seen as a hybrid approach, combining regularization during training with projection in posttraining.

Algorithm 11 REnforce

Input: init params θ, train data $\mathcal{D}$, loss L, box penalty λ_{box}, timeout τ, SGD params
Pretraining:
for pretraining step **do**
  $L_{\mathcal{D},\text{tot}} \leftarrow \textsc{PretrainLoss}(L, \theta, \lambda_{\text{box}}, \mathcal{D})$
  $\theta \leftarrow \theta - \eta_{\text{task}} \nabla_\theta L_{\mathcal{D},\text{tot}}$
end for
Training:
$x', x'', \overline{z}', \overline{z}'', \overline{\nu}, S \leftarrow \text{RGenerate}(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, \tau)$
$\lambda_{\text{prop}} \leftarrow 0$
for training step **do**
  if $\textsc{Resolved}(x', x'', \overline{z}', \overline{z}'')$ **then**
    $x', x'', \overline{z}', \overline{z}'', \bar{\nu}, S \leftarrow \text{RGenerate}(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, \tau)$
  end if
  $L_{\mathcal{D},\text{tot}} \leftarrow \textsc{TrainLoss}(L, \theta, \lambda_{\text{box}}, \lambda_{\text{prop}}, \mathcal{D}, x', x'', \overline{z}', \overline{z}'')$
  $\theta \leftarrow \theta - \eta_{\text{task}} \nabla_\theta L_{\mathcal{D},\text{tot}}$
  $\lambda_{\text{prop}} \leftarrow \lambda_{\text{prop}} + \eta_{\text{prop}} \nabla_{\lambda_{\text{prop}}} L_{\mathcal{D},\text{tot}}$
end for
Posttraining:
for posttraining step **do**
  $x', x'', \overline{z}', \overline{z}'', \overline{\nu}, S \leftarrow \text{RGenerate}(\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g, \tau)$
  if $S =$ infeasible **then**
    $\overline{\nu} \leftarrow 0$
    break
  end if
  $\theta_{\underline{h}}, \theta_{\overline{h}}, \theta_g \leftarrow \text{RProject}(x', x'', \overline{z}', \overline{z}'')$
end for
return $\theta, \overline{\nu}$

Chapter 5

Trace Applications

In this chapter, we present an extensive computational study, conducted to evaluate the behavior of our SMiLE framework on different trace properties, that is, to assess the quality of the trace enforcement algorithm described in Section 4.2. In particular, we design our computational experiments around the following key research questions:

Q1 *Accuracy*: Is SMiLE able to achieve a competitive predictive performance?

Q2 *Efficiency*: Is SMiLE able to achieve a competitive computational performance?

Q3 *Feasibility*: Is SMiLE able to provide formal guarantees on the satisfaction of the desired property?

Q4 *Generality*: Is SMiLE able to consistently enforce different trace properties into different neural architectures?

Q5 *Overapproximation Analysis*: Is the introduced auxiliary architecture able to effectively speed up the verification and enforcement process, while maintaining acceptable accuracy?

Q6 *Hyperparameter Sensitivity*: How is the SMiLE accuracy affected by its key design choices?

We begin by outlining the computational setup and the approaches adopted as baseline. We then present three different trace applications: *safety*, *stability*, and *exclusiveness*. For each of them, we describe the corresponding adopted benchmark, the conducted experiments and the obtained results. Finally, we summarize the insights drawn from the overall study, with the ultimate goal of addressing the questions formulated above.

5.1 Setup

All experiments, methods and benchmarks presented in this chapter are implemented in Python, by relying on the libraries TensorFlow [AAB+15], Keras [C+15] and Scikit-Learn [PVG+11] for the machine learning components, and on Pyomo [HWW11, BHH+21], OMLT [CJH+22] and Gurobi [Gur] for the optimization ones.

The hardware infrastructure where all the experiments are executed consists of an Apple M3 Pro CPU with 11 cores and an Apple M3 Pro GPU with 14 cores, equipped with 36 GB RAM and running macOS v14.1 as operating system.

5.2 Baseline

For trace applications, we adopt a baseline relying on a *maximum-a-posteriori* output projection $\Pi_{\mathcal{Y}}$, defined as:

$$\Pi_{\mathcal{Y}}(y) = \arg\max_{\hat{y}} \{L(\hat{y} \mid y) \text{ s.t. } R(\hat{y})\} \tag{5.1}$$

where L denotes the likelihood of an adjusted prediction with respect to an original one. Intuitively, we correct an infeasible output y, by projecting it to its closest feasible point $\Pi_{\mathcal{Y}}(y)$, as illustrated in Figure 5.1. Note that, unlike the definition of output projection commonly adopted in the literature, which relies on the standard Euclidean norm, the likelihood adopted in our formulation ensures semantic consistency with the given property, that is, it guarantees the most consistent correction with respect to the constraints.

More specifically, in regression, we assume a Normal distribution and homoskedasticity, i.e., the same variance on all training examples, as commonly done in literature. Under this assumptions, Equation (5.1) is equivalent to minimizing the Mean Squared Error:

$$L(\hat{y} \mid y) = -\sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \tag{5.2}$$

where m denotes the output dimension. In classification, without loss of generality, we instead assume a categorical distribution. Under this assumption, Equation (5.1) is equivalent to minimizing the cross-entropy:

$$\mathcal{L}(\hat{y} \mid y) = -\sum_{i=1}^{m} \hat{y}_i \log y_i + (1 - \hat{y}_i) \log(1 - y_i) \tag{5.3}$$

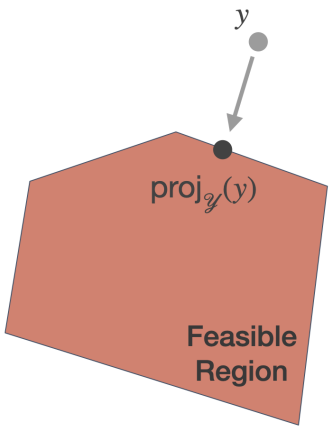


Figure 5.1: An illustration of the projection operator.

Precisely, our baseline consists of two projection-based competitors, PreProcess and PostProcess: both train the model in a property-agnostic fashion, with the former applying $\Pi_{\mathcal{Y}}$ on the data, while the latter on the predictions. Together with them, as an optimal reference, we also consider a theoretical Oracle.

PreProcess. This method applies $\Pi_{\mathcal{Y}}$ in a preprocessing step, to enforce property satisfaction in the data, precisely in the training labels, before the actual training. The main advantage of this approach is its efficiency: given that the highest computational cost, i.e., the one required for property enforcement, is paid offline, the method is fast both at training time, for which any training algorithm can be used, and at inference time. Its drawback, however, is that it is not able to provide formal guarantees on the satisfaction of the property, which makes this approach unsuitable for safety-critical scenarios (e.g., autonomous driving).

PostProcess. This method applies $\Pi_{\mathcal{Y}}$ in a postprocessing step: at inference time, it checks the feasibility of the produced prediction, hence eventually forces it onto the feasible region. The main benefit of this approach is its ability to provide formal satisfaction guarantees, even when trained on potentially infeasible data. Its main disadvantage is that it solves the most computationally demanding operation, i.e., property enforcement, online, which makes this method not an ideal fit for scenarios requiring real-time decisions (e.g., again autonomous driving).

Oracle. As an optimistic reference, we also consider a training-free theoretical approach, Oracle, which applies $\Pi_{\mathcal{Y}}$ to the ground-truth labels both a training and testing time. On one side, Oracle represents the best that we can achieve in terms of model accuracy, if we want to guarantee property satisfaction, that is, it pro-

vides an upper bound on the best feasible predictive performance. On the other side, Oracle represents a metric to quantify the difficulty of satisfying a given property on a dataset, by measuring the accuracy drop caused by the application of $\Pi_{\mathcal{Y}}$.

For all experiments presented in this chapter, unless explicitly indicated, our SMiLE architectures are trained through our trace enforcement framework (Algorithm 4) on property-enforced data: before training SMiLE, as for PreProcesss, we use $\Pi_{\mathcal{Y}}$ on the underlying training data to make sure it satisfies the desired property. Note that, in principle, this pretrainig technique can also be adopted for PostProcess; however, preliminary experiments revealed that, unlike SMiLE, PostProcess does not benefit from this pretraining step.

5.3 Safety

5.3.1 Benchmark

We consider 9 synthetic learning tasks, consisting in the following vector functions:

$$F_{n,\kappa}(x) = \left((\sum_{i=1}^{n} x_i)^{\kappa_1}, \ldots, (\sum_{i=1}^{n} x_i)^{\kappa_m} \right) \tag{5.4}$$

for input dimensions $n \in \{2, 4, 8\}$ and difficulties $\kappa \in \{(1, 2), (3, 4), (1, 2, 3, 4)\}$, with m denoting the κ-dependent output dimension.

For each task $F_{n,\kappa}$, we consider a dataset of 6000 input-output pairs, where each input is uniformly sampled from the interval $[-1, 1]$, that is,

$$\mathcal{D}_{n,\kappa} = \{(x, F_{n,\kappa}(x)),\ x \in [-1, 1]^n\} \tag{5.5}$$

from which we use 5000 instances for training, $\mathcal{D}^{\text{train}}_{n,\kappa}$, and the remaining 1000 for testing, $\mathcal{D}^{\text{test}}_{n,\kappa}$. Finally, we transform this data through standard preprocessing routines (e.g. scaling), as common in any ML workflow.

For a predictor f for task $F_{n,S}$, we consider 6 linear safety properties, selected in order of increasing difficulty (Oracle performance), i.e., of inconsistency with the task/data, from a pool of specifications of the form

$$\forall x \in \mathbb{R}^n : \overbrace{Ax \leqslant b}^{Q(x)} \implies \overbrace{Cf(x) \leqslant d}^{R(f(x))} \tag{5.6}$$

where the two systems $Ax \leqslant b$ and $Cx \leqslant d$ consist of $\log_2(n) + 2$ and $\log_2(m) + 2$ linear constraints, respectively, and each right-hand side and left-hand side parameter is randomly generated from the interval $[-1, 1]$, i.e.,

$$A \in [-1, 1]^{(\log_2(n)+2)\times n} \tag{5.7}$$
$$b \in [-1, 1]^{(\log_2(n)+2)} \tag{5.8}$$
$$C \in [-1, 1]^{(\log_2(m)+2)\times m} \tag{5.9}$$
$$d \in [-1, 1]^{(\log_2(m)+2))} \tag{5.10}$$

These properties can be represented as pairs of polyhedra, respectively defined in the input and output space of the model, as depicted in Figure 5.2. This benchmark is chosen to test, in a synthetic environment, the ability of SMiLE to enforce trace properties into neural networks on property-inconsistent data.

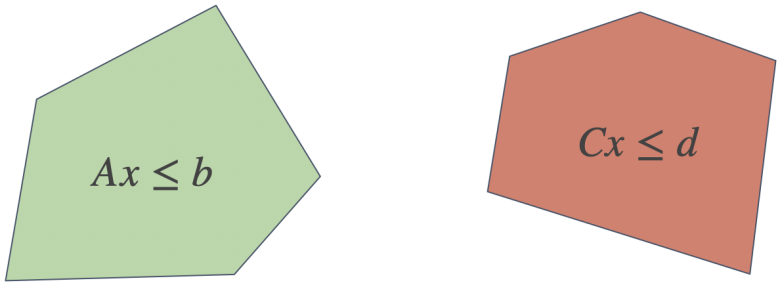


Figure 5.2: An illustration of a safety property.

In order to use our trace enforcer (Algorithm 4) for this benchmark, we need to ground its projector and generator to the properties defined in Equation (5.6), that is, we need to define an implementable formulation of these two components.

Precisely, the projection defined in Equation (4.9), in this setting, can be specified as:

$$\arg\min_{\hat{\theta}_g} \|\hat{\theta}_g - \theta_g\|_2^2 \tag{5.11a}$$
$$\text{s.t.} \quad y = \hat{\theta}_{g,0} + \hat{\theta}_{g,1:n}\,\overline{z}^* \qquad \forall (x^*, \overline{z}^*) \in \mathcal{C} \tag{5.11b}$$
$$Cy \leqslant d \qquad \forall (x^*, \overline{z}^*) \in \mathcal{C} \tag{5.11c}$$

representing a poly-time solvable Quadratic Program.

The process for obtaining a grounded formulation for the counterexample generation problem in Equation (4.14g) is, instead, slightly more involved. The key

observation is that resolving a counterexample $(x^*, \overline{z}^*)$, violating a given linear inequality, requires a translation that is proportional to the classical Linear Programming notion of constraint violation. In particular, let us consider a single linear inequality $C_k y \leqslant d_k$, identified by the k-th row in the matrix C and element in the vector d. Restoring feasibility for $(x^*, \overline{z}^*)$ via translation is, in this case, a Quadratic Program with a positive definite matrix:

$$\nu^* = \arg\min_{\nu} \{\|\nu\|_2^2 \text{ s.t. } C_k(y + \nu) \leqslant d_k\} \tag{5.12}$$

where $y = \theta_{g,0} + \theta_{g,1:n}\,\overline{z}^*$. As such, its solution can be characterized by means of the Karush-Kuhn-Tucker (KKT) optimality conditions [Kar39, KT51], which allow to re-write Equation (5.12) in closed form.

Precisely, we start by defining the problem Lagrangian:

$$L(\nu, \lambda) = \frac{1}{2}\|\nu\|_2^2 + \lambda(C_k(y + \nu) - d_k) \tag{5.13}$$

where the scaling factor $1/2$ is introduced for convenience, while $\lambda \in \mathbb{R}_{\geqslant 0}$ is the multiplier associated to the constraint. The corresponding KKT conditions are:

$$\text{Stationarity:} \qquad \nu^\top + \lambda C_k^\top = 0 \tag{5.14a}$$

$$\text{Primal feasibility:} \qquad C_k(y + \nu) \leqslant d_k \tag{5.14b}$$

$$\text{Dual feasibility:} \qquad \lambda \geqslant 0 \tag{5.14c}$$

$$\text{Complementary slackness:} \qquad \lambda\big(C_k(y + \nu) - d_k\big) = 0 \tag{5.14d}$$

Now, if $C_k y - d_k \leqslant 0$, meaning that $(x^*, \overline{z}^*)$ does not represent a valid counterexample, a solution to Equation (5.14) is obtained by setting $\lambda = 0$ and $\nu = 0$. Conversely, if $C_k y - d_k > 0$, then we have $\nu = -\lambda C_k^T$ and

$$C_k(y - \lambda C_k^T) = d_k \tag{5.15}$$

since complementary slackness implies that the constraint $C_k(y + \nu) \leqslant d_k$ is tight. We can then derive the value of λ as

$$\lambda = \frac{1}{C_k^T C_k}(C_k y - d_k) \tag{5.16}$$

from which we obtain:

$$\nu = -\frac{1}{C_k^T C_k}(C_k y - d_k) C_k^T \tag{5.17}$$

which shows that the amount of translation needed to resolve the counterexample is proportional (in terms of absolute value) to the degree of constraint violation. For simplicity, we drop the constant vector $(C_k^T C_k)^{-1} C_k^T$, hence we account for both considered cases (valid/non-valid) by using the expression:

$$\max(0, C_k y - d_k) \tag{5.18}$$

By considering every constraint in this fashion and summing up, we obtain:

$$\begin{aligned}
\arg\max_{x,\overline{\underline{z}}} \ & \nu && \text{(5.19a)}\\
\text{s.t. } & Ax \leqslant b && \text{(5.19b)}\\
& \overline{\underline{z}} \geqslant \underline{h}(x; \theta_{\underline{h}}) && \text{(5.19c)}\\
& \overline{\underline{z}} \leqslant \underline{h}(x; \theta_{\underline{h}}) + M(1-p) && \text{(5.19d)}\\
& \overline{\underline{z}} \leqslant \overline{h}(x; \theta_{\overline{h}}) + Mp && \text{(5.19e)}\\
& y = \theta_{g,0} + \theta_{g,1:n}\, \overline{\underline{z}} && \text{(5.19f)}\\
& \nu = u^T (Cy - d) && \text{(5.19g)}\\
& p \in \{0,1\}^l, u \in \{0,1\}^m && \text{(5.19h)}
\end{aligned}$$

where l denotes the dimension of the latent space, Equations (5.19c) to (5.19e) model eventually degenerate bounding box constraints, while the binaries u in Equation (5.19g) are used to select only positive components of the vector $Cy - d$, ensuring actual violation.

5.3.2 Accuracy & Feasibility

For each of the 9 tasks and each of the 6 corresponding properties, we produce a property-aware predictor f via each of the competing approaches. In particular, we train on $\mathcal{D}_{n,\kappa}^{\text{train}}$ by adopting the hyperparameters reported in Table 5.1, where MSE stands for Mean Squared Error, and the architectures listed in Table 5.2, where σ_d denotes a network layer of width d and activation function σ. Note that none of these configurations affect Oracle, this being training-free, while some of them affect only SMiLE, such as the capacity of the counterexample queue, as well as the auxiliary architectures. Note moreover that, only for this benchmark, unlike in the methodology and the remaining experimental sections, we use selector-equipped SMiLE architectures, as described in Equation (4.6) and illustrated in Figure 4.3, to handle the adopted non-trivial input predicates.

We evaluate the *accuracy* of each trained model f on the corresponding testing

Parameter	Value
Validation Split	0.2
Loss	MSE
Batch Size	128
Maximum Epochs	1000
Stop Patience	5
Queue Capacity	1

Table 5.1: Training hyperparameters for the safety benchmark.

Component	Architecture
h	$\text{relu}_{nm} \circ \text{relu}_{2nm} \circ \text{relu}_{nm}$
$\underline{h}, \overline{h}$	linear_{nm}
g	linear_{m}

Table 5.2: Architectures for the safety benchmark.

dataset $\mathcal{D}^{\text{test}}_{n,\kappa}$, in terms of the *coefficient of determination* R^2, defined as

$$R^2(f) = 1 - \frac{\sum_{(x,y)\in\mathcal{D}^{\text{test}}_{n,\kappa}}(y - f(x))^2}{\sum_{(x,y)\in\mathcal{D}^{\text{test}}_{n,\kappa}}(y - \overline{y})^2} \tag{5.20}$$

with $\overline{y} = \frac{1}{|\mathcal{D}^{\text{test}}_{n,\kappa}|}\sum_{(x,y)\in\mathcal{D}^{\text{test}}_{n,\kappa}} y$ denoting the mean of the ground-truth labels. Note that $R^2(f)$ ranges in the interval $[-\infty, 1]$, with 1 being the best possible accuracy.

In Figure 5.3, we report a comparison among the considered approaches, by aggregating the R^2 results across the considered properties, sorted by difficulty. The figure shows that, as expected, the performance of all models deteriorates as the difficulty of the enforced property increases. PostProcess, other than providing satisfaction guarantees, seems also able to achieve solid accuracy results, especially on highly demanding properties, where it performs as good as Oracle. As already remarked, however, this method is systematically slow at inference time, although such inefficiency, rather than for the simple linear constraints over continuous variables considered in this benchmark, mostly arises for more complex properties, such as the combinatorial ones investigated in Section 5.5, where inference runtime represents a dramatic limitation of this approach. PreProcess attains higher accuracy than PostProcess and, for difficult properties, can even outperform Oracle. This last observation, although apparently counterintuitive, actually reveals the main limitation of this method, that is, its potential infeasibility: the observed performance advantage of PreProcess, indeed, clearly indicates that this approach violates the enforced property, this being the only way to outperform

Oracle, as demonstrated in Figure 5.4. SMiLE, finally, achieves highly competitive predictive performance on all properties. Moreover, although the convergence of the SMiLE training to zero violation is not theoretically guaranteed, all SMiLE models trained in this experiment successfully achieved it, thereby ensuring full property satisfaction. In other words, our framework demonstrates the ability to match its competitors in terms of accuracy, while avoiding their main limitations, namely, property infeasibility and inference inefficiency, thanks to its feasibility-by-design nature.

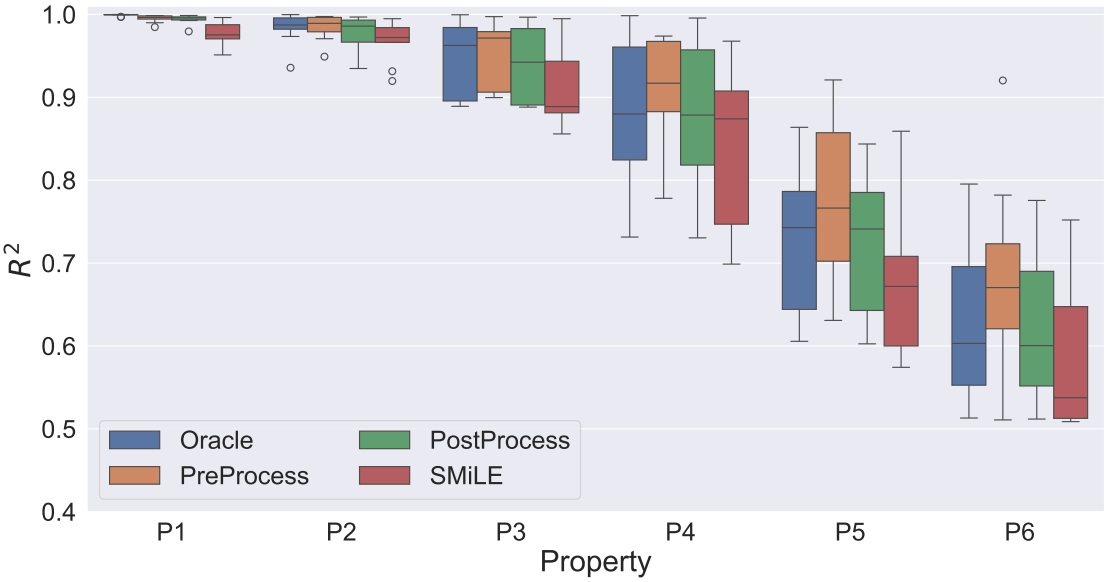


Figure 5.3: Accuracy evaluation on the safety benchmark.

To provide evidence of the unreliability of PreProcess in terms of property satisfaction, we consider an additional metric, i.e., *empirical violation*, defined as:

$$\nu_{\text{emp}}(f) = \frac{100}{|\mathcal{D}^{\text{test}}_{n,\kappa}|} \sum_{(x,y)\in\mathcal{D}^{\text{test}}_{n,\kappa}} \mathbb{1}\left((Ax \leqslant b) \wedge \neg(Cf(x) \leqslant d)\right) \tag{5.21}$$

with $\mathbb{1}$ denoting the property indicator function. This metric quantifies the percentage of test samples (x, y), such that x satisfies the input constraints $Ax \leqslant b$, while $f(x)$ violates at lease one output constraint, i.e., $C_k f(x) > d_k$ for some k.

We display the empirical violation ν_{emp} on the considered properties, sorted again in terms of difficulty, in Figure 5.4, which only reports the evaluation of PreProcess, given that both PostProcess and SMiLE formally guarantee zero violation. As shown by the table, PreProcess exhibits a very high violation rate,

which reaches an average value of almost 20% on the most demanding properties, while exceeding 50% in the worst cases.

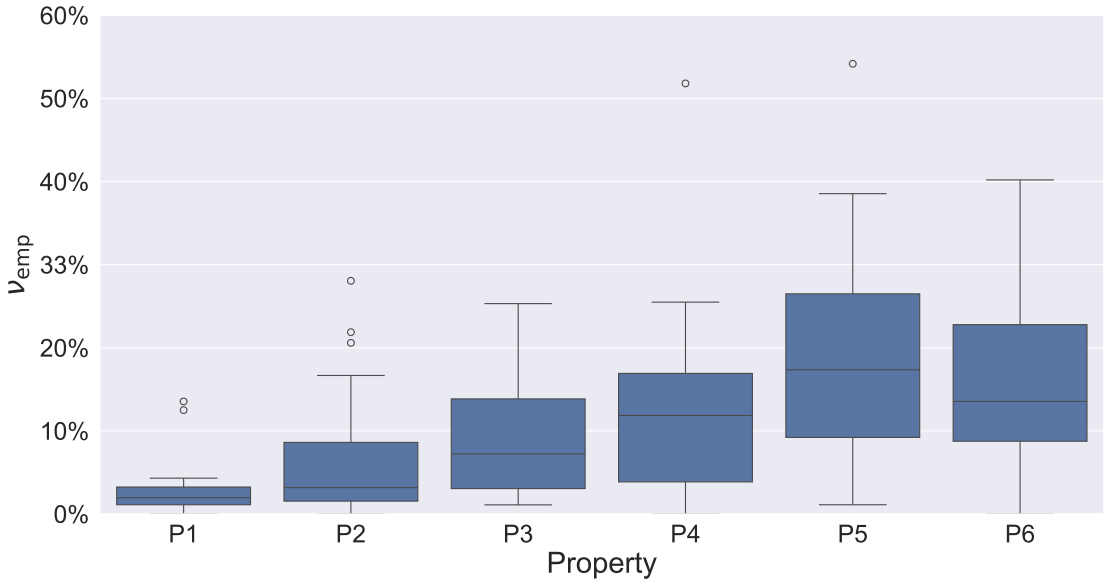


Figure 5.4: PreProcess violation on the safety benchmark.

Finally, we present an ablation study to investigate, in terms of accuracy, both the impact of the key design choices of the SMiLE architecture, namely, the auxiliary models $\underline{h}, \overline{h}$, and the ability of our framework to handle backbones h of different sizes. More precisely, we compare two overapproximators with different complexity, constant versus linear, as well as two embedding models with different depth and width, small versus large, as specified in Table 5.3, while adopting the configurations specified in Table 5.1 and Table 5.2 for all other hyperparamenteres.

Ablation	**Type**	**Architecture**
$\underline{h}, \overline{h}$	constant	constant_{nm}
	linear	linear_{nm}
h	small	$\text{relu}_{nm} \circ \text{relu}_{2nm} \circ \text{relu}_{nm}$
	large	$\text{relu}_{nm} \circ \text{relu}_{2nm} \circ \text{relu}_{3nm} \circ \text{relu}_{2nm} \circ \text{relu}_{nm}$

Table 5.3: Ablation study configurations for the safety benchmark.

Table 5.4, where we report the accuracy results aggregated across tasks and properties, shows that our framework is robust with respect to different archi-

tectural configurations, with only small differences reported among them. This suggests, on one side, that defining the hyperparameters for our framework should not be significantly more difficult than for regular scenarios, and on the other, that SMiLE can handle backbones of different complexities.

Ablation	**Type**	**Accuracy ($\mathbf{R^2}$)**
$\underline{h}, \overline{h}$	constant	0.814
	linear	0.819
h	small	0.819
	large	0.823

Table 5.4: Ablation study results on the safety benchmark.

5.3.3 Overapproximation Analysis

In this section, we analyze the overapproximation produced by the auxiliaries, hence we investigate the precision and efficiency of the resulting verification test.

First of all, we consider different strategies to tighten, for each input x, the predicted box $H(x; \theta_{\underline{h}}, \theta_{\overline{h}})$, hence to increase the quality of the overapproximation. In particular, we design three training losses, to penalize the size of such box at training time. Precisely, given a SMiLE architecture f with latent dimension l, and a standard loss function L, we define:

$$L_1(y, f(x;\theta)) = L(y, f(x;\theta)) + \lambda \frac{m}{l} \frac{\|\overline{h}(x;\theta_{\overline{h}}) - \underline{h}(x;\theta_{\underline{h}})\|_1}{\|\theta_g\|_1} \tag{5.22a}$$

$$L_2(y, f(x;\theta)) = L(y, f(x;\theta)) + \lambda \frac{m}{l} \frac{\|\overline{h}(x;\theta_{\overline{h}}) - \underline{h}(x;\theta_{\underline{h}})\|_2^2}{\|\theta_g\|_2^2} \tag{5.22b}$$

$$L_3(y, f(x;\theta)) = L(y, f(x;\theta)) + \lambda' \frac{1}{l} \|\overline{h}(x;\theta_{\overline{h}}) - \underline{h}(x;\theta_{\underline{h}})\|_2^2 + \lambda'' \frac{1}{m} \|\theta_g\|_2^2 \tag{5.22c}$$

L_1 penalizes the box size by introducing a penalty term, multiplied by a penalty weight λ, consisting in the ratio between the ℓ^1-norm of the component-wise distances between the two bounds, normalized by the hidden dimension l, and the ℓ^1-norm of the head weights, normalized by the output dimension m. L_2 replaces the ℓ^1-norm in L_1 with the squared ℓ^2-norm, to mitigate issues with non-zero gradients and training instability. Finally, L_3 separates the regularizer numerator and denominator into two terms with distinct weights λ' and λ''.

For the task of minimum dimension and medium difficulty $F_{2,(3,4)}$, we train a SMiLE architecture for each of the 3 regularized losses L_1, L_2 and L_3, with $\lambda, \lambda', \lambda'' = 10^{-4}$, as well as one for the standard non-regularized loss $L_0 = L$, by varying the architectures as reported in Table 5.5, while adopting the configurations specified in Table 5.1 and Table 5.2 for all other hyperparamenteres. In particular, the training is performed in a property-agnostic fashion, rather than via the trace enforcement framework described in Algorithm 4, given that the goal of this computational study is to analyze the impact of each regularizer on the size of the overapproximation box, regardless of any desired property to enforce.

Ablation	**Type**	**Architecture**
$\underline{h}, \overline{h}$	linear	linear_8
	non-linear	$\text{linear}_8 \circ \text{relu}_3 \circ \text{linear}_8$
h	small	linear_8
	medium	$\text{relu}_8 \circ \text{relu}_3 \circ \text{linear}_8$
	large	$\text{relu}_8 \circ \text{relu}_{16} \circ \text{relu}_{32} \circ \text{relu}_{16} \circ \text{linear}_8$

Table 5.5: Ablation study configurations for the safety benchmark.

Figures 5.5 and 5.6 report, for each task and regularization approach, the training evolution of the box size, computed as the total ℓ^1-norm of the component-wise distances between the bounds, and the one of the validation loss, respectively, both aggregated over the batch data at each epoch.

Figure 5.5 shows that the considered regularizers, across all architectures, are able to tighten the overapproximation box during training, with L_1 and L_3 being the least and most effective, respectively. In contrast, the unregularized training L_0 exhibits the opposite behavior: without regularization, rather than moving $\underline{h}$ and $\overline{h}$ close to each other, the training algorithm pushes them apart, evidently to avoid clipping and, consequently, to maximize accuracy, by allowing the more expressive h to freely operate in between the bounds. This represents the motivation underlying the formulation adopted, in the relational enforcement framework, for the projector and pretraining, which indeed are both designed to prevent unreasonably large and non-homogeneous boxes, as explained in Section 4.3.1 and Section 4.3.3. Figure 5.6 shows instead that, apart from a slight convergence slowdown observed for L_1 for the small backbone case, the validation loss evolves, for all introduced regularizations, similarly to the standard L_0 during training, indicating that the introduction of the penalty term does not compromise accuracy.

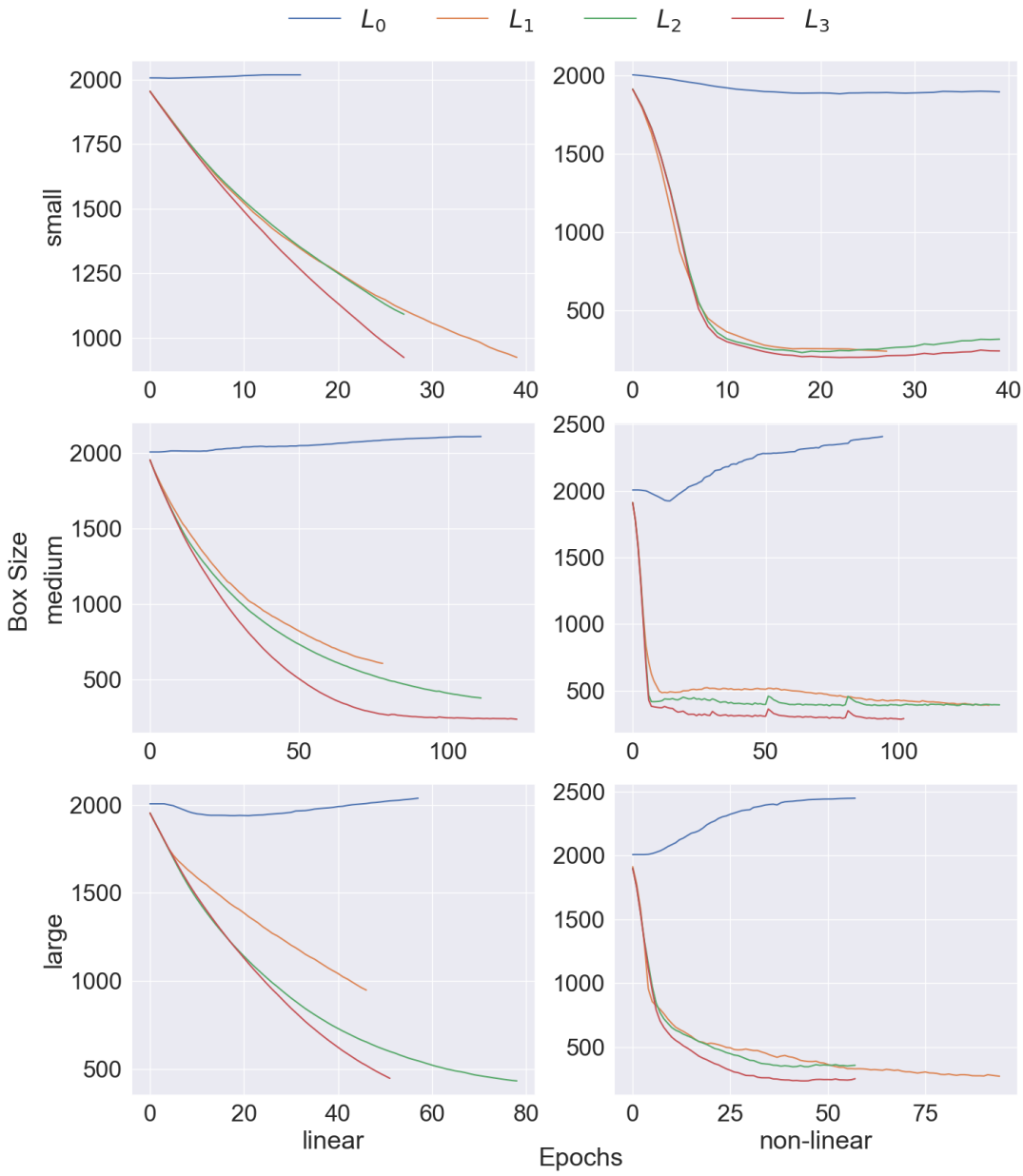


Figure 5.5: Box size evolution during training on the safety benchmark.

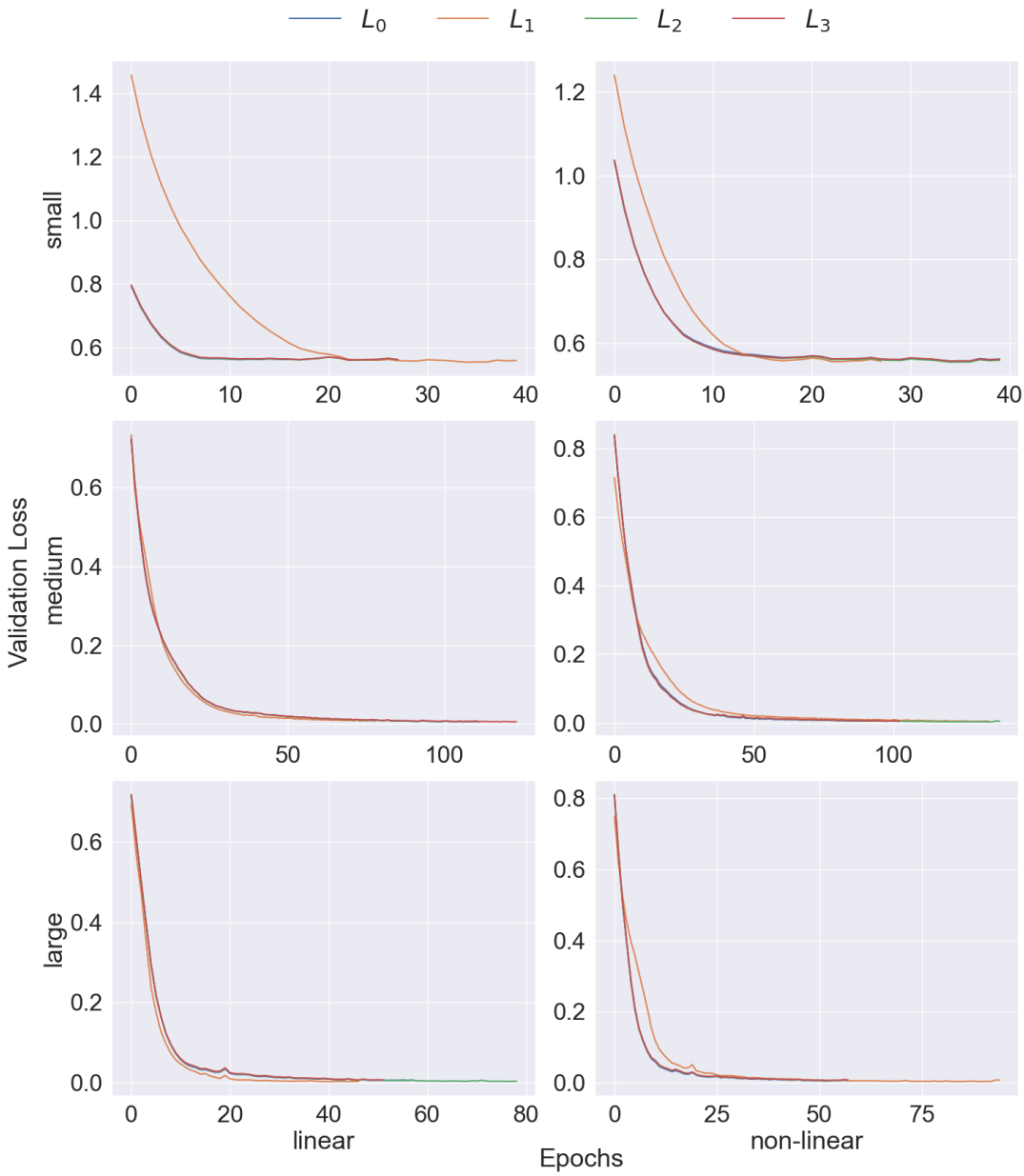


Figure 5.6: Validation loss evolution during training on the safety benchmark.

Besides the effect of the regularizers on the size of the box and on the accuracy of the model at training time, we are primarily interested in their impact on the precision of the SMiLE verification test, whose improvement represents the ultimate objective of this experiment. To this aim, in a post-hoc phase, we consider 10 properties of different difficulties, again for the task $F_{2,(3,4)}$, and we verify, for each of them, each trained SMiLE architecture f through our trace generator TGenerate (Equation (5.19)), with a time limit of 300s. Then, for each model, we count the number of false (non-violating) counterexamples produced across the 10 properties, that is, the pairs $(x^*, \overline{z}^*) \in \mathbb{R}^n \times H(x; \theta_{\underline{h}}, \theta_{\overline{h}})$, flagged with positive violation, on which however the model does not violate, that is:

$$(Ax^* \leqslant b) \wedge \neg(Cg(\overline{z}^*) \leqslant d) \wedge (Cf(x^*) \leqslant d) \tag{5.23}$$

Figure 5.7 depicts the average percentage of false counterexamples for the considered penalty approaches. We observe that, even by training without a box penalty term (L_0), the resulting SMiLE generator guarantees an acceptable precision, being able to produce a true counterexample, on average, in 85% of the cases, while failing only in the remaining 15%. Adopting a regularization, however, significantly decreases the average failure rate, which indeed drops below 2% for the most effective regularizers (L_2 and L_3).

The eventual generation of false counterexamples represents the price to pay for enabling the otherwise intractable verification and subsequent enforcement, embodying the core idea of SMiLE: trading precision for tractability. To empirically validate this intuition, we finally investigate the computational efficiency of our approximate SMiLE generator, in comparison with two variants: an exact verifier for the full SMiLE architecture, comprising $h, \underline{h}, \overline{h}$, clip and g, and an exact verifier designed solely for the backbone h and head g. The former provides insights into the computational resources necessary for applications that require not only sound, but also complete SMiLE verification, whereas the latter reveals the computational burden of performing even a single property-verification step, prior to any enforcement, on a standard neural network.

To obtain an exact SMiLE generator, we need to re-introduce the backbone and the clipping operator into the backbone-agnostic SMilE generator described

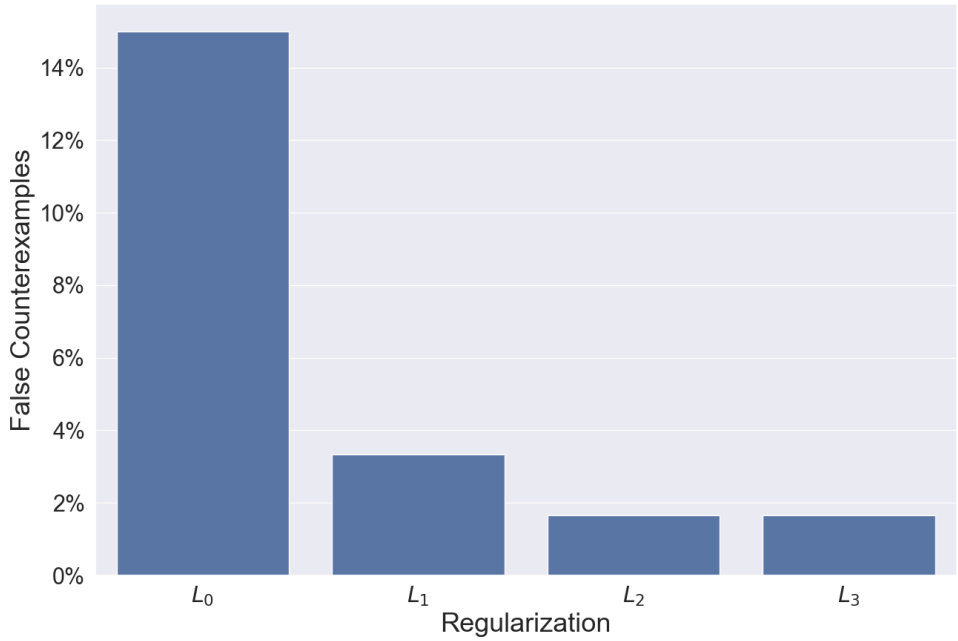


Figure 5.7: False counterexamples produced on the safety benchmark.

in Section 4.2.2. Precisely, for safety properties:

$$\arg\max_{x} \nu \tag{5.24a}$$

$$\text{s.t. } Ax \leqslant b \tag{5.24b}$$

$$\overline{z} \geqslant \underline{h}(x;\theta_{\underline{h}}) \tag{5.24c}$$

$$\overline{z} \leqslant \underline{h}(x;\theta_{\underline{h}}) + M(1-p) \tag{5.24d}$$

$$\overline{z} \leqslant \overline{h}(x;\theta_{\overline{h}}) + Mp \tag{5.24e}$$

$$\overline{z} \leqslant h(x;\theta_h) + M\underline{p} \tag{5.24f}$$

$$\overline{z} \geqslant h(x;\theta_h) - M\overline{p} \tag{5.24g}$$

$$\overline{z} \leqslant \underline{h}(x;\theta_{\underline{h}}) + M(1-\underline{p}) \tag{5.24h}$$

$$\overline{z} \geqslant \overline{h}(x;\theta_{\overline{h}}) - M(1-\overline{p}) \tag{5.24i}$$

$$\underline{p} + \overline{p} \leqslant 1 \tag{5.24j}$$

$$y = \theta_{g,0} + \theta_{g,1:n}\,\underline{z} \tag{5.24k}$$

$$\nu = u^T(Cy - d) \tag{5.24l}$$

$$p, \underline{p}, \overline{p} \in \{0,1\}^l, u \in \{0,1\}^m \tag{5.24m}$$

where Equations (5.24f) to (5.24i), together with the bounding box constraints formulated in Equations (5.24c) to (5.24e), model the clipping operator in Equation (4.3), i.e.,

$$\overline{\underline{z}}_i = \begin{cases} \underline{h}(x;\theta_{\underline{h}})_i & \text{if } h(x;\theta_h)_i < \underline{h}(x;\theta_{\underline{h}})_i \\ h(x;\theta_h)_i & \text{if } \underline{h}(x;\theta_{\underline{h}})_i \leqslant h(x;\theta_h)_i \leqslant \overline{h}(x;\theta_{\overline{h}})_i \\ \overline{h}(x;\theta_{\overline{h}})_i & \text{if } h(x;\theta_h)_i > \overline{h}(x;\theta_{\overline{h}})_i \end{cases} \tag{5.25a}$$

with the binaries $\underline{p}_i$ and $\overline{p}_i$ indicating, for each component $i \in [l]$, the interval where $h(x;\theta_h)_i$ falls:

$$\underline{p}_i, \overline{p}_i = \begin{cases} 1,0 & \text{if } h(x;\theta_h)_i < \underline{h}(x;\theta_{\underline{h}})_i \\ 0,0 & \text{if } \underline{h}(x;\theta_{\underline{h}})_i \leqslant h(x;\theta_h)_i \leqslant \overline{h}(x;\theta_{\overline{h}})_i \\ 0,1 & \text{if } h(x;\theta_h)_i > \overline{h}(x;\theta_{\overline{h}})_i \end{cases} \tag{5.26a}$$

Note that, while the clipped $\overline{\underline{z}}$, in the approximate formulation provided in Equation (5.19), represents a free variable, in the exact formulation in Equation (5.24), it is determined by the input x via the introduced backbone h and clipping clip. We instead formulate the exact backbone generator as follows:

$$\arg\max_x \ \nu \tag{5.27a}$$

$$\text{s.t. } Ax \leqslant b \tag{5.27b}$$

$$z = h(x;\theta_h) \tag{5.27c}$$

$$y = \theta_{g,0} + \theta_{g,1:n}\, z \tag{5.27d}$$

$$\nu = u^T(Cy - d) \tag{5.27e}$$

$$u \in \{0,1\}^m \tag{5.27f}$$

In Figures 5.8 and 5.9, we report a comparison of the three verification approaches, for each architectural configuration, in terms of percentage of termination status and average runtime, respectively, over the considered regularizations and properties. In particular, in Figure 5.8, *optimal* means that the process converged to optimality, while *timelimit* and *nosol* indicate that it has reached the time limit with and without a primal solution, respectively. Figure 5.8 shows that, while the approximate SMiLE verifier consistently returns an optimal counterexample, both the exact SMiLE and the exact backbone verifiers, for the most complex configurations, fail to converge to optimality. Interestingly, despite operating on a larger architecture, the exact SMiLE verifier attains optimality more frequently than the exact backbone one in most cases. However, for large backbones and non-linear auxiliaries, its performance drops significantly, by failing to

produce any solution at all in many cases. Overall, this suggests that, when sufficiently simple, the introduced auxiliary architecture can reduce the difficulty of the verification process, not only in the relaxed setting, but also in the exact one. Figure 5.9 shows instead that, while the approximate verifier fully scales with the size of the backbone, being only affected by the complexity of the auxiliary components, the exact verifiers experience a significant slowdown as the backbone grows. Unlike the exact backbone verifier, however, the exact SMiLE verifier remains capable of handling large backbones, as long as the overapproximation remains linear, confirming the trend observed in Figure 5.8. Conversely, exact backbone verification becomes intractable even for medium-sized architectures, thereby reinforcing the core motivation underlying the methodology introduced in this thesis: without the structural advantages of a SMiLE architecture, even the enforcement of simple properties, such as linear constraints, becomes infeasible for standard neural networks.

Figure 5.8: Termination status evaluation on the safety benchmark.

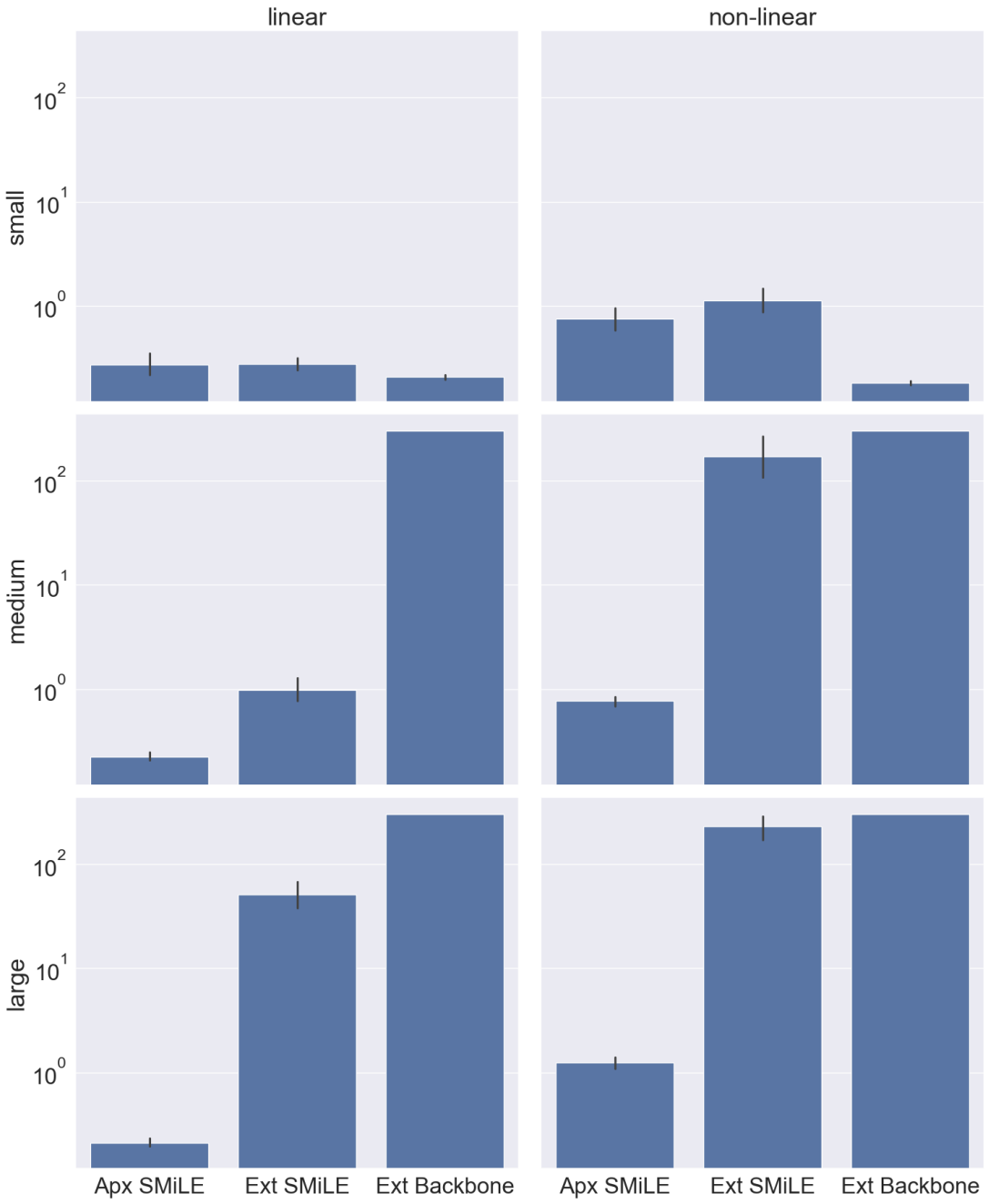


Figure 5.9: Runtime (s) evaluation on the safety benchmark.

5.4 Stability

5.4.1 Benchmark

In this set of experiments, we consider *Multi-Step Time Series Forecasting*, a problem arising in a wide range of domains, such as energy [LDD18], manufacturing [ZYW19] or epidemiology [SSK+20], just to name a few. The problem consists in estimating a window of m consecutive values $s_{t+1}, \ldots, s_{t+m}$ of a time series, by observing a window of n previous values $s_{t-n+1}, \ldots, s_t$ of the same series $S = (s_t)_{t=0}^{t=N}$, as depicted in Figure 5.10.

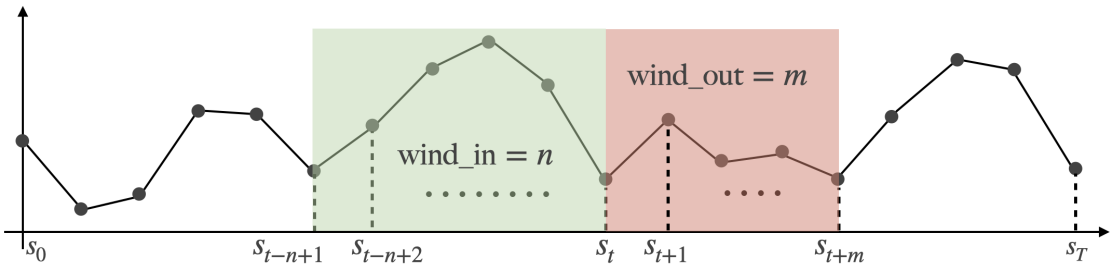


Figure 5.10: An illustration of a multi-step time series forecasting task.

We train and test on public data released by the forecasting M4 Competition [MSA20], which includes real-world time series collected from a variety of sectors, such as demographic, finance and industry, among others. Specifically, from this repository, we select 25 series for which the PreProcess approach guarantees both a good predictive performance and some degree of violation in a least one of the adopted properties, described in the following paragraph. Each series $S = (s_t)_{t=0}^{t=N}$ represents a single learning task, and is preprocessed via common preprocessing techniques in time series forecasting [BJRL15]. Precisely, we first apply a *differencing* transformation to each series, to remove trends and seasonality, that is, we replace it with the stationary $S = (s_{t+1} - s_t)_{t=0}^{t=N-1}$. We then use an *overlapping windowing operation* to obtain a learning-suitable dataset $\mathcal{D}_S$ of input-output pairs

$$(x, y) = ((s_{t-n+1}, \ldots, s_t), (s_{t+1}, \ldots, s_{t+m})) \tag{5.28}$$

for $t = n, \ldots, N - m$, where the input and output dimensions n and m, as a result of a pilot experiment, are fixed to 8 and 4, respectively. Finally, we split this dataset according to a chronological 80%-20% criterion, as illustrated in Figure 5.11, to produce a train and test set $\mathcal{D}_S^{\text{train}}$ and $\mathcal{D}_S^{\text{test}}$, respectively, that we finally standardize as common in deep learning pipelines.

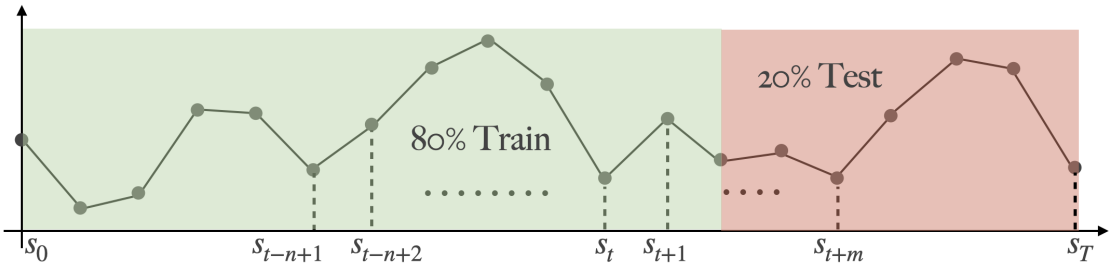


Figure 5.11: The adopted train-test split for each forecasting task.

Finally, for a time series forecaster f and a series S, we consider 3 properties defined as:

$$\forall x \in \mathbb{R}^n : \overbrace{\top}^{Q(x)} \implies \overbrace{|f(x)_i - f(x)_{i+1}| = |s_i - s_{i+1}| \leqslant \Delta_q, \forall i \in [m]}^{R(f(x))} \tag{5.29}$$

for $q = 0.90, 0.95, 1.00$, where $\Delta_q \in \mathbb{R}_{\geqslant 0}$ denotes the q-quantile of the set of deviations $\{|s_t - s_{t+1}|\}_{t=0}^{t=N}$. For a given series, in particular, q is inversely correlated with the difficulty of the property: the smaller its value, the more demanding the corresponding property. As illustrated in Figure 5.12, these properties can be thought of as a form of stability: they prevent unreasonably high deviations between two consecutive predictions.

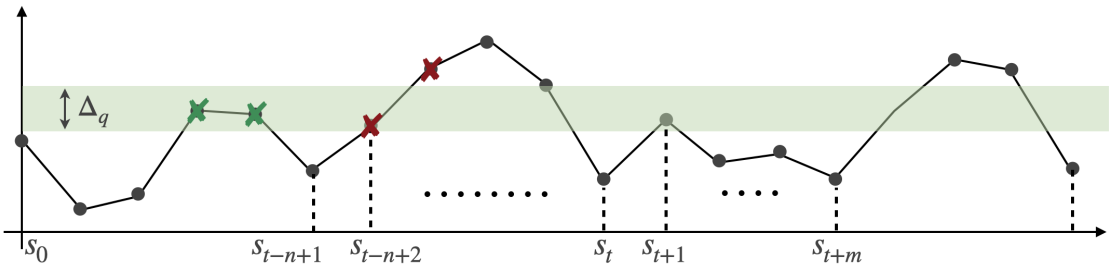


Figure 5.12: An illustration of a stability property.

Note that Equation (5.29) can be equivalently defined in polythopic form as:

$$\forall x \in \mathbb{R}^n : \overbrace{\top}^{Q(x)} \implies \overbrace{Cf(x) \leqslant d}^{R(f(x))} \tag{5.30}$$

with

$$C = \begin{pmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & -1 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -1 & 1 \end{pmatrix} \quad d = \begin{pmatrix} \Delta_q \\ \Delta_q \\ \vdots \\ \Delta_q \\ -\Delta_q \\ -\Delta_q \\ \vdots \\ -\Delta_q \end{pmatrix} \tag{5.31}$$

These properties, in other words, represent a special case of the linear safety specifications used in the previous section, defined in Equation (5.6). For this reason, we can enforce these properties into a SMiLE architecture by adopting, in the trace enforcement algorithm, the same generator and projector groundings presented in Section 5.3.1.

5.4.2 Accuracy

For each of the 25 tasks and each of the 3 corresponding properties, we produce a property-aware predictor f via each of the competing approaches. In particular, we train the training-based approaches by adopting the hyperparameters and architectures listed in Tables 5.2 and 5.6, respectively.

Parameter	Value
Validation Split	0.2
Optimizer	Adam
Loss	MSE
Batch Size	32
Maximum Epochs	1000
Stop Patience	15
Queue Capacity	1

Table 5.6: Training hyperparameters for the stablity benchmark.

Component	Architecture
h	$\text{relu}_{nm} \circ \text{relu}_{2nm} \circ \text{relu}_{3nm} \circ \text{relu}_{2nm} \circ \text{relu}_{nm}$
$\underline{h}, \overline{h}$	constant_{nm}
g	linear_{m}

Table 5.7: Architectures for the stability benchmark.

Finally, similarly to the safety benchmark, we evaluate the *accuracy* of each trained model, on the corresponding test set $\mathcal{D}_S^{\text{test}}$, in terms of the *coefficient of determination* R^2, defined in Equation (5.20).

In Figure 5.13, we compare the accuracy of the four competitors across the three properties, sorted by difficulty. The figure shows that, for this benchmark, the considered models are able to achieve a prediction quality very close to the one of Oracle, with an average R^2 value that, for all of them, remains above 0.95. Also in this benchmark, in particular, our method SMilE converged to zero violation during its training, which demonstrates its ability to match its competitors in terms of accuracy, while providing full satisfaction guarantees by design. Finally, we observe that, in contrast to the safety benchmark (Figure 5.3), in the stability one, the theoretical Oracle always achieves maximum accuracy, except for some outliers. This is because, in this benchmark, the considered properties are more realistic with respect to the underlying tasks, unlike in the safety one, where properties and tasks, being synthetic, can be strongly misaligned.

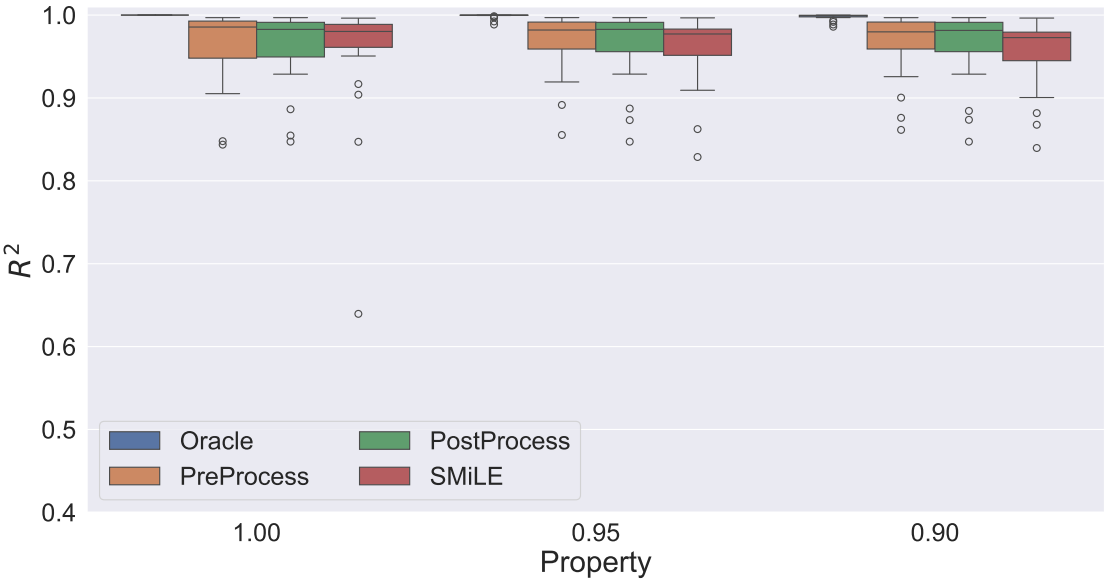


Figure 5.13: Accuracy evaluation on the stability benchmark.

Finally, we conduct an ablation study to analyze, again, both the robustenss of SMiLE with respect to the chosen auxiliary models $\underline{h}$ and $\overline{h}$, as well as its generality in terms of the adopted backbone h. In particular, following the setup of the safety benchmark, we compare two auxiliary complexities, constant versus linear, and two backbone sizes, small versus large. For this benchmark, moreover, we also explore two different backbone architectures: a Feedforward Neural Network (FNN) and

a Recurrent Neural Network (RNN), where the latter employs LSTM cells in the first layer to better capture the sequential nature of the considered time series forecasting tasks. The detailed configurations are provided in Table 5.8, whereas the remaining hyperparameters follow the settings reported in Table 5.6.

Ablation	**Type**	**Architecture**
$\underline{h}, \overline{h}$	constant	constant_{nm}
	linear	linear_{nm}
h	small	$\text{relu}_{nm} \circ \text{relu}_{2nm} \circ \text{relu}_{nm}$
	large	$\text{relu}_{nm} \circ \text{relu}_{2nm} \circ \text{relu}_{3nm} \circ \text{relu}_{2nm} \circ \text{relu}_{nm}$
	FNN	$\text{relu}_{nm} \circ \text{relu}_{2nm} \circ \text{relu}_{3nm} \circ \text{relu}_{2nm} \circ \text{relu}_{nm}$
	RNN	$\text{lstm}_{nm} \circ \text{relu}_{2nm} \circ \text{relu}_{3nm} \circ \text{relu}_{2nm} \circ \text{relu}_{nm}$

Table 5.8: Ablation study configurations for the stability benchmark.

Table 5.9, which reports the accuracy results aggregated across tasks and properties, shows a very small variability in the SMiLE accuracy across the different setups, demonstrating its robustness and generality with respect to its hyperparameters and the backbone, respectively. In particular, constant auxiliary models $\underline{h}$ and $\overline{h}$ yield higher accuracy than linear ones, suggesting that, for the sake of efficiency, the complexity of the overapproximation can eventually be reduced to a minimum, without compromising performance. Finally, and remarkably, the RNN backbone also achieves promising accuracy, even if lower than its FNN variant, demonstrating the abilty of SMiLE to seamlessly handle recurrent architectures as well, hence confirming its flexibility and applicability to a broader class of models.

Ablation	**Type**	**Accuracy ($\mathbf{R^2}$)**
$\underline{h}, \overline{h}$	constant	0.958
	linear	0.953
h	small	0.962
	large	0.958
	FNN	0.958
	RNN	0.950

Table 5.9: Ablation study results on the stability benchmark.

5.5 Exclusiveness

5.5.1 Benchmark

In this benchmark, we consider *Multi-Label Classification*, a problem occurring in a diverse range of sectors, from biomedicine [DIP+24] to autonomous driving [LZD+25]. The problem consists in tagging a given input x with multiple labels $y \in \{0,1\}^m$, where $y_i = 1$ indicates that the corresponding tag i is assigned, as illustrated in Figure 5.14.

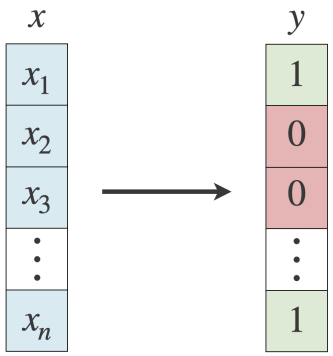


Figure 5.14: An illustration of multi-label classification.

We train and test on data from a UCI public repository [Moy24], which offers a variety of multi-label datasets across different domains. From this repository, in particular, we select 5 sets of different size, input and output dimension, and cardinality, that is, the average number of labels associated with each instance, as listed in Table 5.10. Each set $\mathcal{D}$ represents a single learning task, that we split according to a random 80%-20% criterion into a train and test set, $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$, respectively. Also in this case, moreover, we apply standard preprocessing transformations to the data.

Dataset	Size	Input	Output	Cardinality	Domain
Water-quality	1060	16	14	5.07	Chemistry
Flags	194	19	7	3.39	Image
Yeast	2417	103	14	4.23	Biology
foodtruck	407	21	12	2.29	Recommendation
CHD-49	555	49	6	2.58	Medicine

Table 5.10: Selected multi-label classification datasets.

For a multi-label classifier f and a dataset $\mathcal{D}$, we consider 3 exclusiveness properties of the form

$$\forall x \in \mathbb{R}^n : \overbrace{\top}^{Q(x)} \implies \overbrace{\mathbb{1}(f(x)_u \geqslant 0) + \mathbb{1}(f(x)_v \geqslant 0) \leqslant 1,\ \forall u, v \in \mathcal{F}}^{R(f(x))} \tag{5.32}$$

for $q = 0.0, 0.3, 0.6$, where $f(x)$ denotes the logit output of the model, while $\mathcal{F}$ consists in the following set of mutually exclusive classes:

$$\mathcal{F} = \{(u, v) \in [m]^2 \mid \text{freq}(u, v) \leqslant \Delta_q\} \tag{5.33}$$

with O representing the set of possible classes, $\text{freq}(u, v)$ the frequency with which the pair (u, v) occurs among the true labels in $\mathcal{D}$, and Δ_q the q-quantile of the set of pair frequencies $\{\text{freq}(u, v)\}_{u,v \in [m]}$. Note that, in this case, the difficulty of the property directly correlates with q: the larger its value, the more demanding the corresponding property. These specifications can be thought of as a form of safety or stability: we prevent the prediction of dangerous or unusual combinations, as depicted in Figure 5.15.

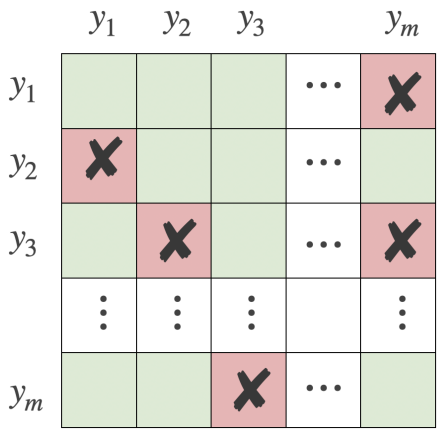


Figure 5.15: An illustration of an exclusiveness property.

Equation (5.32) represents a *combinatorial property*, due to the presence of the binary function $\mathbb{1}$ in its definition, whose action is equivalent to the sigmoid activation, typically used in the last network layer to transform the continuous logit into the final binary label. The combinatorial nature of these properties, in particular, prevents the adoption of the projector and generator adopted in the previous benchmarks (Section 5.3.1), which need to be reformulated.

Precisely, the projection problem from eq. (4.9), for given head weights θ_g and counterexample queue $\mathcal{C}$, can be grounded to an exclusiveness property as follows:

$$\begin{aligned}
&\arg\min_{\hat{\theta}_g} \|\hat{\theta}_g - \theta_g\|_2^2 && && \text{(5.34a)}\\
&\quad \text{s.t. } y = \hat{\theta}_{g,0} + \hat{\theta}_{g,1:n}\,\underline{\overline{z}}^* && \forall (x^*, \underline{\overline{z}}^*) \in \mathcal{C} && \text{(5.34b)}\\
&\qquad M I_i \geqslant y_i && \forall i \in [m], \forall (x^*, \underline{\overline{z}}^*) \in \mathcal{C} && \text{(5.34c)}\\
&\qquad I_u + I_v \leqslant 1 && \forall (u,v) \in F, \forall (x^*, \underline{\overline{z}}^*) \in \mathcal{C} && \text{(5.34d)}\\
&\qquad I_i \in \{0,1\} && \forall i \in [m], \forall (x^*, \underline{\overline{z}}^*) \in \mathcal{C} && \text{(5.34e)}
\end{aligned}$$

where I_i is an auxiliary variable modeling the $\mathbb{1}(y_i \geqslant 0)$ predicates, while $M > 0$ denotes a sufficiently large constant used to allow $I_i = 0$ only if y_i is negative. Equation (5.34) is a Mixed Integer Quadratic Programming problem, which modern mathematical programming solvers are capable to handle.

To design an implementable formulation of the trace generator in Algorithm 3 for exclusiveness properties, instead, we rely on the observation that, if mutually exclusive classes u and v are predicted at the same time, then forcing either $f(x)_u$ or $f(x)_v$ to be less than 0 resolves the violation. Precisely:

$$\begin{aligned}
&\arg\max_{x, \underline{\overline{z}}} \ \nu && && \text{(5.35a)}\\
&\quad \text{s.t. } Q(x) && && \text{(5.35b)}\\
&\qquad \underline{\overline{z}} \geqslant \underline{h}(x; \theta_{\underline{h}}) && && \text{(5.35c)}\\
&\qquad \underline{\overline{z}} \leqslant \underline{h}(x; \theta_{\underline{h}}) + M(1-p) && && \text{(5.35d)}\\
&\qquad \underline{\overline{z}} \leqslant \overline{h}(x; \theta_{\overline{h}}) + Mp && && \text{(5.35e)}\\
&\qquad y = \theta_{g,0} + \theta_{g,1:n}\,\underline{\overline{z}} && && \text{(5.35f)}\\
&\qquad M I_i - M \leqslant y_i && \forall i \in [m] && \text{(5.35g)}\\
&\qquad I_{u,v} \leqslant \frac{1}{2}(I_u + I_v) && \forall u, v \in \mathcal{F} && \text{(5.35h)}\\
&\qquad t_{u,v} \leqslant y_u && \forall u, v \in \mathcal{F} && \text{(5.35i)}\\
&\qquad t_{u,v} \leqslant y_v && \forall u, v \in \mathcal{F} && \text{(5.35j)}\\
&\qquad \nu = I_{u,v}\, t_{u,v} && \forall u, v \in \mathcal{F} && \text{(5.35k)}\\
&\qquad I_i \in \{0,1\} && \forall i \in [m] && \text{(5.35l)}\\
&\qquad I_{u,v}, t_{u,v} \in \{0,1\} && \forall u, v \in \mathcal{F} && \text{(5.35m)}\\
&\qquad p \in \{0,1\}^l && && \text{(5.35n)}
\end{aligned}$$

where Equations (5.35c) to (5.35e) model the eventually degenerate box constraints, Equation (5.35g) ensures that the indicator variables I_i can be set to

1 only if the corresponding logit output is at least 0, Equation (5.35h) allows the auxiliary binary variable $I_{u,v}$ to 1 only if the corresponding forbidden pair u and v is predicted, and finally Equations (5.35i) to (5.35k) linearize the operator $\min(y_u, y_v)$, which is the maximized in the objective. Equation (5.35) is a Mixed Integer Quadratic Program, solvable by modern combinatorial solvers.

5.5.2 Accuracy & Runtime

For each of the 5 datasets and 3 corresponding properties, we produce a property-aware predictor f via each of the competing approaches. In particular, we train the training-based methods on $\mathcal{D}^{\text{train}}$, by adopting the hyperparameters listed in Table 5.11, where BCE stands for Binary Cross-Entropy, and architectures listed in Table 5.12.

Parameter	Value
Validation Split	0.2
Optimizer	Adam
Loss	BCE
Batch Size	32
Maximum Epochs	1000
Stop Patience	30
Queue Capacity (SMiLE)	10

Table 5.11: Training hyperparameters for the exclusiveness benchmark.

Component	Architecture
h	$\text{relu}_{\lfloor 4\log_2(nm)\rfloor} \circ \text{relu}_{\lfloor 8\log_2(nm)\rfloor} \circ \text{relu}_{\lfloor 4\log_2(nm)\rfloor}$
$\underline{h}, \overline{h}$	$\text{constant}_{\lfloor 4\log_2(nm)\rfloor}$
g	$\text{linear}_{\lfloor 4\log_2(nm)\rfloor}$

Table 5.12: Architectures for the exclusiveness benchmark.

We evaluate the *accuracy* of the multi-label classifiers f, on $\mathcal{D}^{\text{test}}$, in terms of the *average class accuracy*, commonly adopted in multi-label classification:

$$\text{AvgAcc}(f) = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{|\mathcal{D}^{\text{test}}|}\sum_{(x,y)\in\mathcal{D}^{\text{test}}}\mathbf{1}(y_j = f(x)_j)\right) \tag{5.36}$$

Note that $\text{AvgAcc}(f)$ ranges within $[0, 1]$, with 1 representing the best possible accuracy.

Figure 5.16 presents a comparison of the four competing methods, in terms of accuracy, across the three considered properties, arranged by increasing difficulty. For this benchmark, all trained models, when compared with their safety and stability counterparts, exhibit lower predictive performance, evidently due to higher difficulty of the multi-label classification problem. In any case, our framework demonstrates strong competitiveness, being able, in some cases, to even outperform its competitors. Remarkably, also for this benchmark, all SMiLE models reach zero property violation by the end their training, which reinforces the trend consistently observed through the computational study presented in this chapter: SMiLE is able to predict as good as its competitors, while providing full satisfaction guarantees by design, that is, without requiring any expensive post-hoc correction of the prediction to ensure its feasibility.

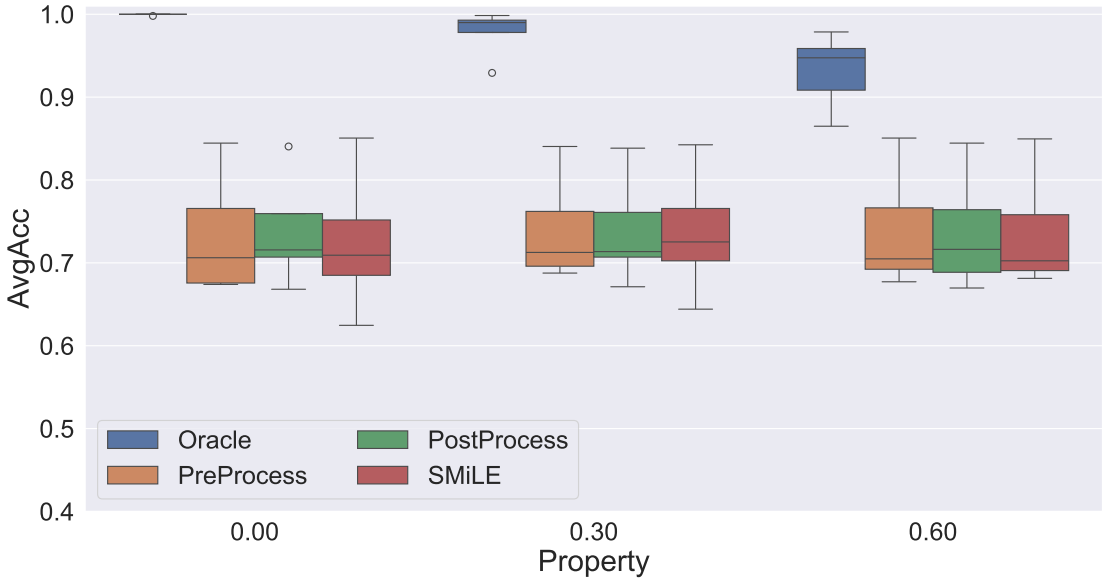


Figure 5.16: Accuracy evaluation on the exclusiveness benchmark.

In particular, the ability to produce feasible-by-design models, in this setting, becomes notably desirable, given the combinatorial nature of the considered specifications. For these properties, indeed, a feasibility-enforcing output projection, such as the one adopted by PostProcess, can become computationally very demanding, hence can substantially limit the applicability of the approach. Clear evidence of the intractability of PostProcess, for this benchmark, is provided in Figure 5.17, where we report the inference runtime, across the considered properties, of PostProcess and SMiLE, relative to the one of PreProcess, whose performance is comparable to the one of any standard network.

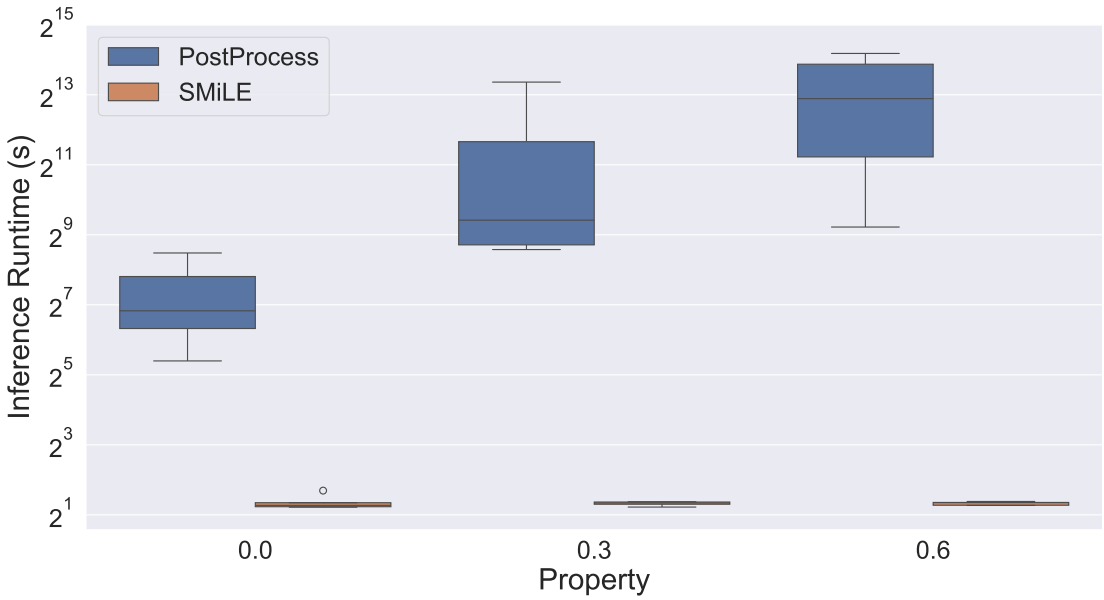


Figure 5.17: Inference runtime (s) evaluation on the exclusiveness benchmark.

As shown by the figure, to provide feasibility guarantees, PostProcess requires a significant computational effort at inference time, with a slowdown rising up to 2^{13} on the most demanding properties. In contrast, SMiLE is, regardless of property difficulty, only around twice as slow as PreProcess at inference time, evidently due to the effect of the introduced overapproximation architecture.

5.6 Discussion

We conclude this chapter by answering the research questions formalized in its introduction.

Q1 *Accuracy*: SMiLE demonstrated the ability to match, across all evaluated benchmarks, the predictive performance of its competitors, and in few cases to even surpass them.

Q2 *Efficiency*: On the exclusiveness benchmark, SMiLE offers, in terms of inference runtime, a substantial advantage over postprocess, which pays a prohibitive computational cost to guarantee the validity of the property.

Q3 *Feasibility*: In all considered applications, SMiLE consistently converged to zero property violation, hence providing full property satisfaction guarantees by design.

Q4 *Generality*: Throughout the presented computational analysis, SMiLE demonstrated the ability to enforce a broad spectrum of properties, from simple linear specifications to more demanding combinatorial constraints, into a variety of neural architectures, from small feed-forward networks to larger recurrent structures.

Q5 *Overapproximation Analysis*: On the safety benchmark, the auxiliary architecture adopted by SMiLE proved able to significantly speed up the property verification problem, while guaranteeing an acceptable verification precision, especially when trained though dedicated Lagrangian techniques, hence to realistically enable the otherwise intractable production of feasible-by-design neural networks.

Q6 *Hyperparameter Sensitivity*: The ablation study, conducted on the safety and stability benchmarks, highlighted the robustness of SMiLE with respect to its key design choices, suggesting that the effort required by the hyperparamenter configuration of our framework should not be considerably larger than the one necessary for standard neural architectures.

Chapter 6

Relational Applications

Building upon the previous chapter, presenting an evaluation of our framework SMiLE on trace properties, in this chapter we conduct an extensive computational study to assess its performance on relational ones, that is, to investigate the quality of the relational enforcement framework described in Section 4.3. In particular, similarly to Chapter 5, we design this empirical analysis around a set of key research questions, namely:

Q1 *Accuracy*: Is SMiLE able to achieve a competitive predictive performance?

Q2 *Efficiency*: Is SMiLE able to achieve a competitive computational performance?

Q3 *Feasibility*: Is SMiLE able to provide formal guarantees on the satisfaction of the desired property?

Q4 *Generality*: Is SMiLE able to consistently enforce different trace properties into different neural architectures?

Q5 *Applicability* Is SMiLE able to provide practically valuable outcomes?

We begin by outlining the computational setup, after that we present three different relational applications: *monotonicity*, *robustness* and *fairness*. For each of them, we describe the corresponding adopted benchmark and baseline, the conducted experiments and the obtained results. Finally, we summarize the insights drawn from the overall study, aiming at addressing the questions formulated above. Note that, while in the trace assessment we consider property-generic baselines, that is, approaches capable of handling different trace specifications, for relational properties we evaluate SMiLE against property-specific competitors, due to the lack of methodologies, in the state of the art, capable of handling all relational applications considered in this study.

6.1 Setup

Our relational SMiLE framework, as well as all benchmarks and experiments presented in this chapter, are implemented in Python, by relying on the libraries TensorFlow [AAB+15], Keras [C+15] and Scikit-Learn [PVG+11] for the machine learning components, and on Pyomo [HWW11, BHH+21], OMLT [CJH+22] and Gurobi [Gur] for the optimization ones. For the considered competitors, instead, we used the official code released by their authors.

The hardware infrastructure where all the experiments are executed consists of an Apple M3 Pro CPU with 11 cores and an Apple M3 Pro GPU with 14 cores, equipped with 36 GB RAM and running macOS v15.5 as operating system.

6.2 Monotonicity

6.2.1 Benchmark

We consider 9 synthetic tasks, given by the 1-to-1 *sinusoidally perturbed* function

$$F_{\alpha,\omega}(x) = x + \alpha \sin(\omega x) \tag{6.1}$$

for *amplitude* $\alpha \in \{2, 3, 4\}$ and *frequency* $\omega \in \{0.6, 0.8, 1.0\}$.

For each task $F_{\alpha,\omega}(x)$, we consider a dataset of 1000 input-output pairs, where each input is uniformly sampled from the interval $[-10, 10]$, that is,

$$\mathcal{D}_{\alpha,\omega} = \{(x, F_{\alpha,\omega}(x)),\ x \in [-10, 10]^n\} \tag{6.2}$$

from which we use 800 instances for training, $\mathcal{D}^{\text{train}}_{\alpha,\omega}$, and the remaining 200 for testing, $\mathcal{D}^{\text{test}}_{\alpha,\omega}$. Finally, we transform this data through standard preprocessing routines (e.g. scaling), as common in any ML workflow.

For a model f predicting the task $F_{\alpha,\omega}$, we consider the following *non-decreasing monotonicity* property:

$$\forall x \in \mathbb{R}\colon \overbrace{x' \leqslant x''}^{Q(x',x'')} \implies \overbrace{f(x') \leqslant f(x'') + \epsilon}^{R(f(x'),f(x''))} \tag{6.3}$$

where $\epsilon > 0$ is a tolerance factor, necessary to facilitate the convergence to zero violation. Without any tolerance ($\epsilon = 0$), indeed, Equation (6.3) can be easily violated by selecting an input x and two distinct embeddings within the corresponding box, i.e., $\underline{z}', \underline{z}'' \in \underline{H}(x; \theta_{\underline{h}}, \theta_{\overline{h}})$ with $\underline{z}' \neq \underline{z}''$, such that $g(\underline{z}') \neq g(\underline{z}'')$. The

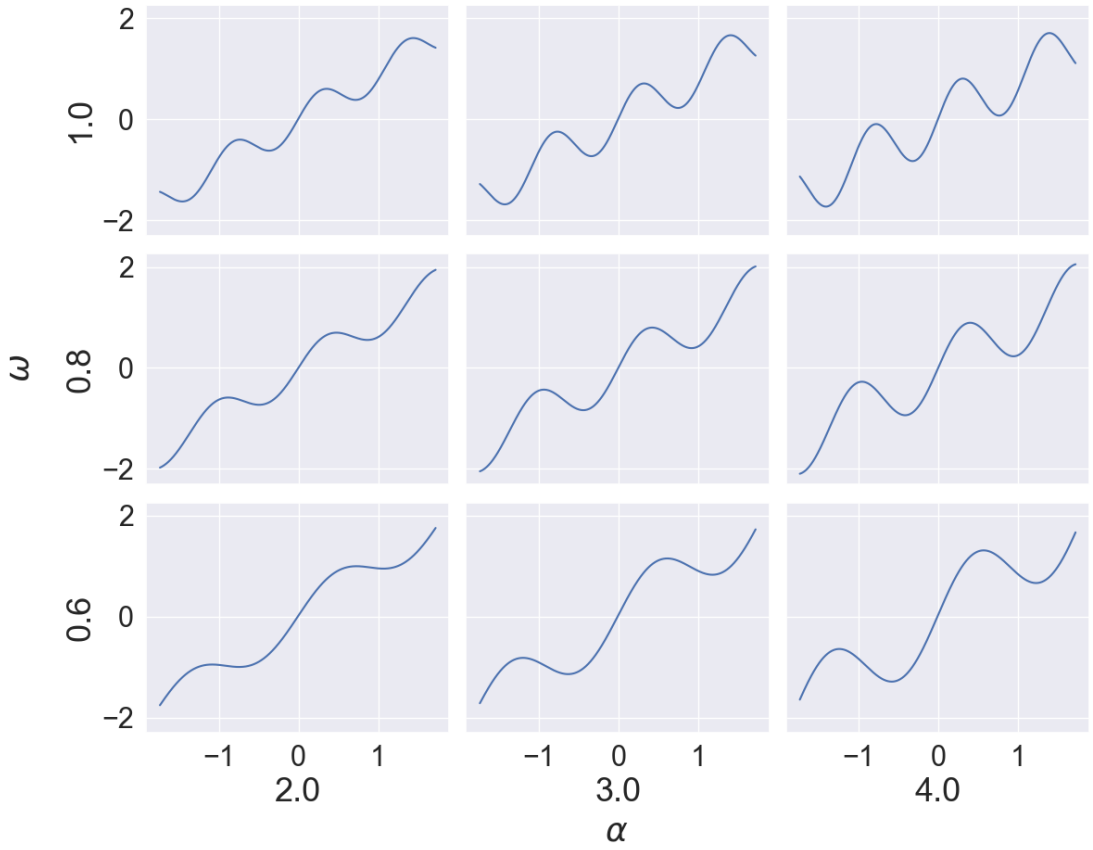


Figure 6.1: The considered tasks for the monotonicity benchmark.

only way to avoid such trivial violations would be to make g constant over the entire box $\overline{\underline{H}}(x; \theta_{\underline{h}}, \theta_{\overline{h}})$ for each input x, which however represents an overly restrictive condition.

Note that, according to the general definition of relational properties provided in Equation (4.16), Equation (6.3) can be equivalently specified as follows:

$$\forall x \in \mathbb{R}: \overbrace{-M}^{\underline{\delta}} \leqslant x' - x'' \leqslant \overbrace{0}^{\overline{\delta}} \implies \overbrace{-M}^{\underline{\epsilon}} \leqslant f(x') - f(x'') \leqslant \overbrace{\epsilon}^{\overline{\epsilon}} \tag{6.4}$$

where $M > 0$ is a sufficiently large real value. In other words, Equation (6.3) can be enforced into a SMilE architecture by using the relational enforcer (Algorithm 11) in a straightforward fashion, that is, without any grounding of its projector and generator, differently from the trace applications presented in the previous chapter.

This benchmark is chosen to test, in a controlled environment, the ability of SMiLE to enforce relational properties into neural networks on property-inconsistent data, similarly to the synthetic safety benchmark presented in Section 5.3. Indeed, the degree of violation of the task in Equation (6.1), with respect to property in Equation (6.3), progressively increases together with α and ω, which provides a challenging testbed to assess the behavior of our framework under a property-data misalignment scenario.

6.2.2 Accuracy

For each of the 9 tasks, we produce a property-aware predictor f using the SMiLE relational framework. In particular, we train the models on the train set $\mathcal{D}^{\text{train}}_{n,\kappa}$, by adopting the hyperparameters reported in Table 6.1 and the architectures listed in Table 6.2, where some configurations affect only SMiLE, namely, time and iteration limit for the generator, and the auxiliary models.

Parameter	**Value**
Optimizer	Adam
Loss	MSE
Batch Size	64
Pretrain Epochs	1000
Pretrain Stop Patience	10
Train Epochs	100
Generator Time Limit (s)	2
Generator Iteration Limit	8

Table 6.1: Training hyperparameters for the monotonicity benchmark.

Component	**Architecture**
h	$\text{relu}_{16} \circ \text{relu}_{32} \circ \text{relu}_{64} \circ \text{relu}_{32} \circ \text{relu}_{16} \circ \text{linear}_8$
$\underline{h}, \overline{h}$	$\text{relu}_{32} \circ \text{linear}_8$
g	linear_1

Table 6.2: Architectures for the monotonicity benchmark.

We evaluate the *accuracy* of each trained model f on the corresponding test dataset $\mathcal{D}^{\text{test}}_{n,\kappa}$, in terms of the *coefficient of determination* R^2, defined in Equation (5.20). In the line plot in Figure 6.2, we depict the curves of the produced SMiLE models, while in the heatmap in Figure 6.3, we report their accuracy. We highlight that the training of each model converged to zero property violation.

Figure 6.2 depicts how the SMiLE curves, in order to maximize accuracy while guaranteeing feasibility, follow the underlying tasks as long as these remain monotonic, while they tend to flatten in the non-monotonic regions. Figure 6.3 shows instead how our method can, overall, achieve acceptable results even when the property is substantially violated by the data.

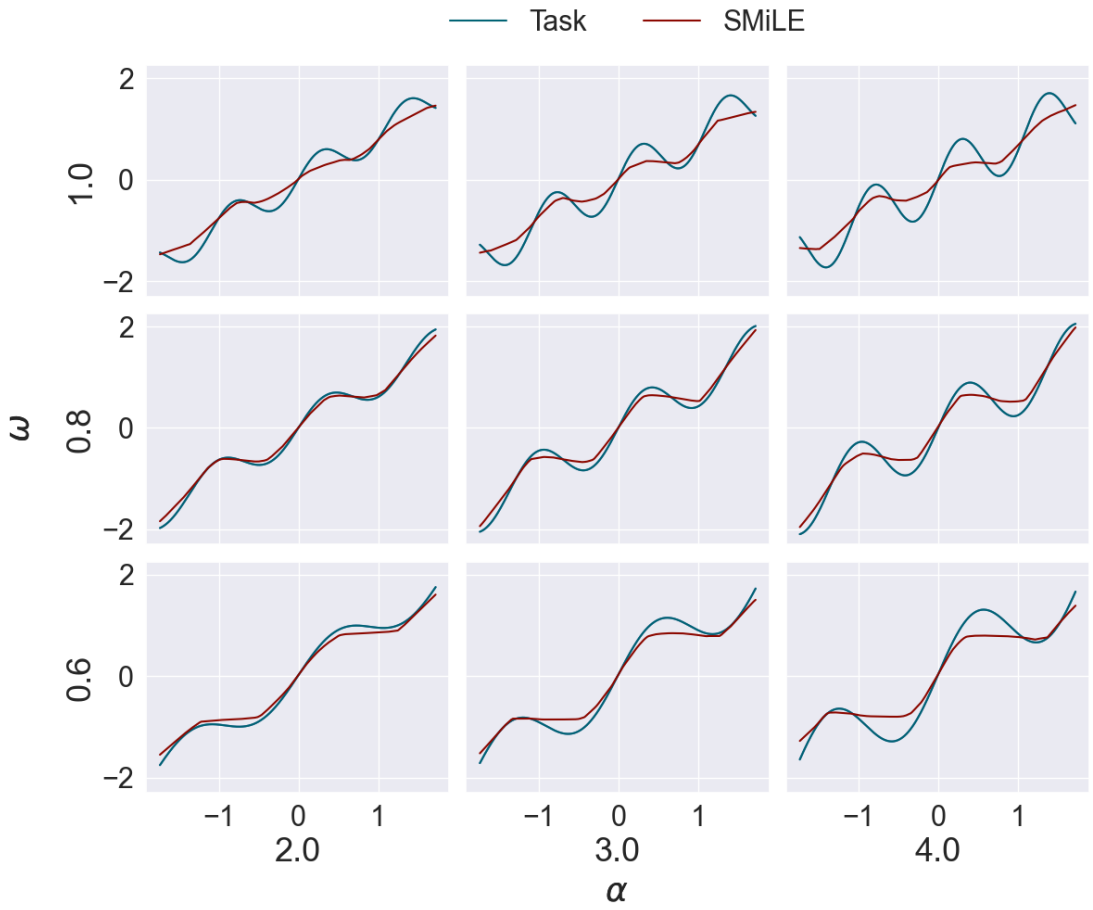


Figure 6.2: Monotonic models trained on the monotonicity benchmark.

6.3 Robustness

6.3.1 Benchmark

We consider the binary classification task of detecting the digit "0" from the MNIST dataset [LBBH98b], a publicly available and widely used benchmark to train and evaluate computer vision models. Precisely, the dataset consists of 70,000 grayscale

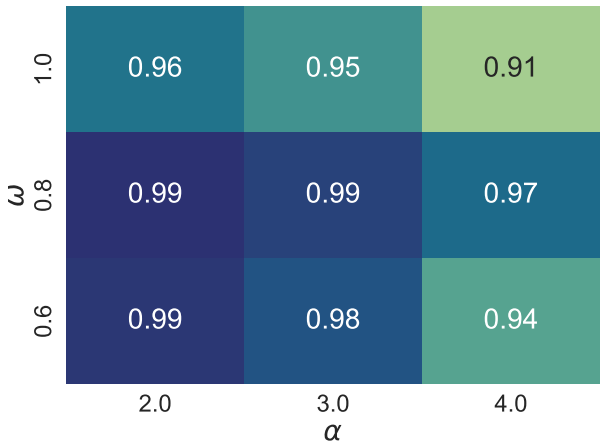


Figure 6.3: Accuracy evaluation on the monotonicity benchmark.

images of handwritten digits (0-9), as exemplified in Figure 6.4, each of size $n = 28 \times 28$ and with each pixel ranging from 0 (white) to 255 (black), which we rescale within 0 and 1, so that each input x falls in the hypercube $[-1, 1]^n$. The dataset $\mathcal{D}$ is divided into 60,000 training samples, $\mathcal{D}^{\text{train}}$, and 10,000 testing ones, $\mathcal{D}^{\text{test}}$.

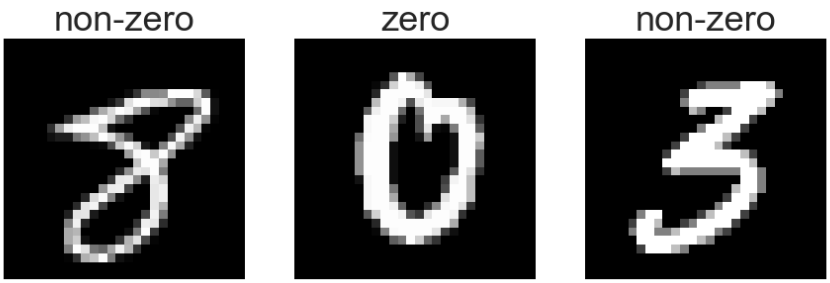


Figure 6.4: Examples of MNIST instances labeled according to our classification.

For a binary classifier f, we consider 15 δ, ϵ-*robustness* properties of the form

$$\forall x', x'' \in \mathbb{R}^n : \overbrace{\|x' - x''\|_\infty \leqslant \delta}^{Q(x', x'')} \implies \overbrace{|f(x') - f(x'')| \leqslant \epsilon}^{R(f(x'), f(x''))} \tag{6.5}$$

where $f(x) \in \mathbb{R}$ is meant as the model logit, for $\delta \in \{0.010, 0.025, 0.050, 0.075, 0.100\}$ and $\epsilon \in \{0.75, 1.00, 1.25\}$. In other words, we force a bound on the logit variation, provided a bound on the input variation, measured in terms of the l^p-norm. Note that the final class predicted by the model can be obtained by applying a sigmoid to the logit $f(x)$, and then rounding the result to the closest integer, i.e., $\text{round}(\text{sigmoid}(f(x))) \in \{0, 1\}$, or equivalently by thresholding the logit against

0 via the indicator function $\mathbb{1}(f(x) > 0) \in \{0, 1\}$, where 0 and 1 represent class "non-zero" and "zero", respectively.

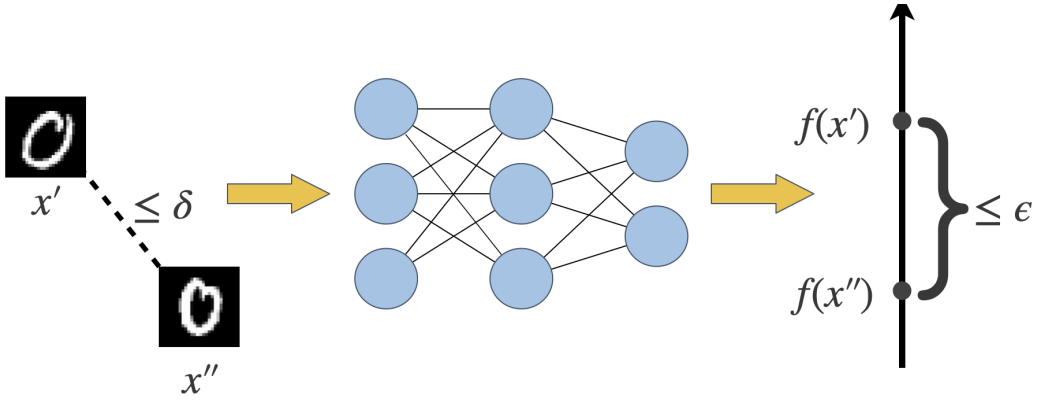


Figure 6.5: An illustration of a δ, ϵ-robustness property.

Similarly to monotonicity, the properties defined in Equation (6.5) can be equivalently specified according to the general definition of relational properties provided in Equation (4.16), precisely as

$$\forall x', x'' \in \mathbb{R}^n : \overbrace{-\delta}^{\underline{\delta}} \leqslant x' - x'' \leqslant \overbrace{\delta}^{\overline{\delta}} \implies \overbrace{-\epsilon}^{\underline{\epsilon}} \leqslant f(x') - f(x'') \leqslant \overbrace{\epsilon}^{\overline{\epsilon}} \tag{6.6}$$

for $\delta, \epsilon \geqslant 0$. In other words, also for this benchmark we can use the SMiLE relational enforcer (Algorithm 11) without any further grounding of its algorithmic components, namely, projector and generator.

The δ, ϵ-robustness defined in Equation (6.5) ensures that, for each input x, $f(\mathcal{B}^n_\delta(x)) \subseteq \mathcal{B}^1_\epsilon(f(x))$, where $\mathcal{B}^m_\gamma(x)$ denotes the m-dimensional ball of radius γ centered in x:

$$\mathcal{B}^m_\gamma(x) = \{\hat{x} \in \mathbb{R}^n : \|x - \hat{x}\|_\infty \leqslant \gamma\} \tag{6.7}$$

This means that, whenever the logit $f(x)$ falls into $[-\infty, -\epsilon) \cup (\epsilon, \infty]$, the class predicted by the model remains consistent over each l^p perturbation of x of "strength" at most δ, as none of such perturbations would be able to change the logit by more than ϵ, hence to flip the prediction, as illustrated in Figure 6.6.

This allows to equip the model with an efficient, specifically *constant-time*, *rejection-based mechanism*, to use in real time against adversarial attacks. Precisely, the defense checks whether the logit $f(x)$, for a test input x, lies outside the rejection region $[-\epsilon, \epsilon]$, hence, according to the result, it either issues a safety certificate, or raises a warning to the user, as illustrated in Figure 6.7. In other

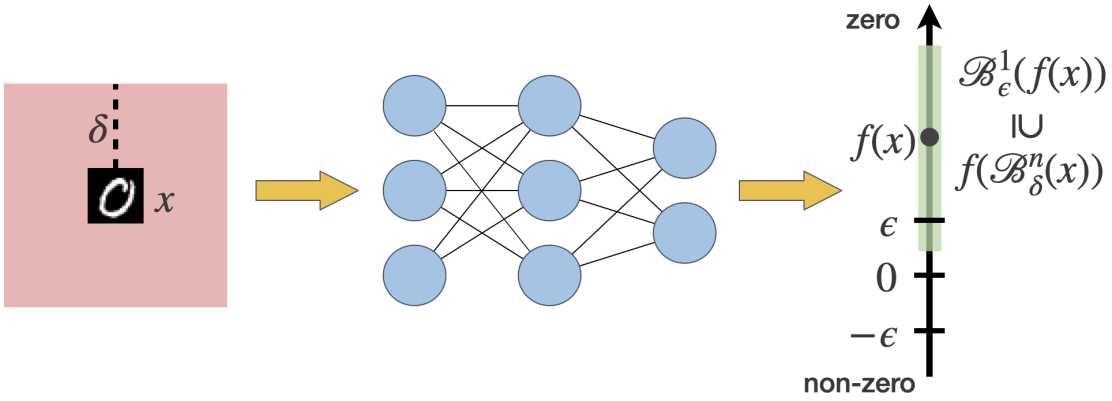


Figure 6.6: Consistency of a δ, ϵ-robust model f under perturbations of radius δ.

words, providing formal guarantees on δ, ϵ-robustness, that is, on the global relational property defined in Equation (6.5), triggers certifications on adversarial robustness defined around each point in the domain, that is, on the set of the following local trace properties

$$\forall x \in \mathbb{R}^n : \overbrace{\|\dot{x} - x\|_\infty \leqslant \delta}^{Q(x)} \implies \overbrace{\mathbb{1}(f(\dot{x}) > 0) = \mathbb{1}(f(x) > 0)}^{R(f(x))} \tag{6.8}$$

for each $\dot{x} \in \mathbb{R}^n$. According to the taxonomy of enforcement methods provided in Section 3.4, SMiLE can then be seen as a guaranteed enforcer with respect to global robustness, and a certified enforcer with respect to adversarial robustness.

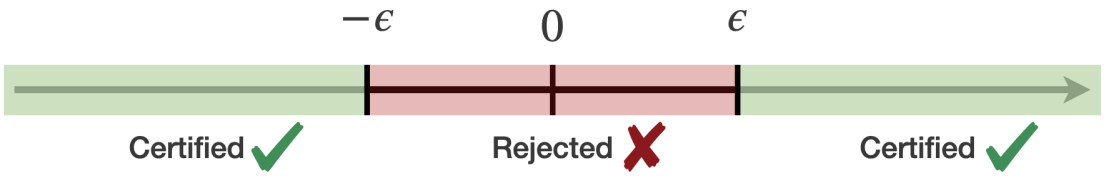


Figure 6.7: An illustration of the SMiLE adversarial defense.

6.3.2 Baseline

As baseline for this benchmark we consider CROWN-IBP [ZCX+20], a state-of-the-art enforcer for adversarial robustness. This method penalizes property violation, by training the model against a convex combination of a standard and a robustness loss, where the latter is computed locally around the training data via a sound but incomplete robustness verifier, integrating the already-existing CROWN (COnvex

Relaxation of Neural Networks) [ZWC+18] with IBP (Interval Bound Propagation) [GDS+19]. The method depends on the maximum allowed perturbation δ, as well as on a convex combination hyperparameter λ, dynamically adjusted according to a scheduling strategy. However, it does not depend on ϵ, that is, it does not allow to decide an a priori bound on the output variation.

At inference time, similarly to SMiLE, CROWN-IBP offers a defense mechanism to detect, and eventually reject, malicious inputs. At training time, however, the two approaches differ fundamentally: while SMiLE enforces robustness globally over the entire input domain, CROWN-IBP only promotes robustness locally around the training samples, thus enforcing a weaker notion of robustness. In other words, SMiLE acts as a guaranteed enforcer for global robustness, whereas CROWN-IBP serves as a certified enforcer for local one. This asymmetry is expected to penalize SMiLE in terms of predictive accuracy, yet to benefit it in terms of defense efficiency: indeed, while the global relational property enforced in SMiLE enables a constant-time rejection-based defense, the rejection region in CROWN-IBP is input-dependent, hence must be computed online, for each test instance, using the adopted verifier.

Beyond the type of enforcement, the two methods also differ in generality, both in terms of supported properties and learning paradigms: in contrast to SMiLE, CROWN-IBP can only enforce adversarial robustness in classification models.

Finally, together with CROWN-IBP, we also consider a base model unaware of the property to satisfy, which we refer to as Agnostic.

6.3.3 Accuracy & Runtime

On the considered task, we train an Agnostic model in a property-agnostic fashion, 5 CROWN-IBP models, one for each of the considered δ, and 15 SMiLE models, one for each of the considered δ and ϵ. All three approaches are trained on the training set $\mathcal{D}^{\text{train}}$, using the hyperparameters and architectures reported in Table 6.3 and Table 6.4, respectively. For CROWN-IBP–specific hyperparameters, we instead rely on the default configuration provided by the authors in their original paper or implementation.

As common in the adversarial robustness literature, we evaluate each trained model f in terms of three types of accuracy: *clean*, *PGD*, and *verified*. Clean

Parameter	Value
Optimizer	Adam
Loss	MSE
Batch Size	256
Pretrain Epochs	20
Pretrain Stop Patience	-
Train Epochs	20
Generator Time Limit (s)	2
Generator Iteration Limit	8

Table 6.3: Training hyperparameters for the robustness benchmark.

Component	Architecture
h	$\text{conv}_8 \circ \text{conv}_8 \circ \text{conv}_{16} \circ \text{conv}_{16} \circ \text{conv}_{32} \circ \text{conv}_{32} \circ$ $\text{flatten} \circ \text{relu}_{32} \circ \text{linear}_8$
$\underline{h}, \overline{h}$	linear_8
g	linear_1

Table 6.4: Architectures for the robustness benchmark.

accuracy corresponds to the standard accuracy on the unperturbed test set:

$$\text{Clean}(f) = \frac{1}{|\mathcal{D}^{\text{test}}_{1000}|} \sum_{(x,y)\in\mathcal{D}^{\text{test}}_{1000}} \mathbb{1}\big(y = \mathbb{1}\big(f(x) > 0\big)\big)\,, \tag{6.9}$$

i.e., the percentage of correctly classified test instances. PGD accuracy measures the empirical robustness under adversarial perturbations:

$$\text{PGD}_\delta(f) = \frac{1}{|\mathcal{D}^{\text{test}}_{1000}|} \sum_{(x,y)\in\mathcal{D}^{\text{test}}_{1000}} \mathbb{1}\big(y = \mathbb{1}\big(f(\text{PGD-Attack}_\delta(x)) > 0\big)\big)\,, \tag{6.10}$$

where $\delta \geqslant 0$, while $\text{PGD-Attack}_\delta(x)$ denotes the adversarial example generated from x via a 100-step Projected Gradient Descent (PGD) attack [MMS+19] of strength (maximum allowed perturbation) at most δ. Verified accuracy, finally, represents the percentage of predictions that are both correct and formally certified:

$$\text{Verified}_\delta(f) = \frac{1}{|\mathcal{D}^{\text{test}}_{1000}|} \sum_{(x,y)\in\mathcal{D}^{\text{test}}_{1000}} \mathbb{1}\big(y = \mathbb{1}\big(f(x) > 0\big) \wedge \text{Accept}_\delta(x)\big)\,, \tag{6.11}$$

where $\text{Accept}_\delta(x)$ indicates whether, according to the SMiLE or CROWN-IBP certification mechanism, the prediction for x, with respect to the perturbation δ, is

accepted (i.e., provably robust). Note that, if the model lacks a certification mechanism (e.g., Agnostic), then Accept(x) always evaluates to $\perp$ (false). Moreover, all three metrics range from a minimum of 0 to a maximum of 1, i.e., the higher the better.

To ensure the computational tractability of the evaluation, we restrict it to the first 1,000 instances of the test set, i.e., $\mathcal{D}^{\text{test}}_{1000} \subset \mathcal{D}^{\text{test}}$. This limitation is motivated by the computation of the PGD accuracy, which involves running a heuristic yet computationally intensive attack.

We compare the predictive performance of the three competitors in Table 6.5, where all reported SMiLE models are guaranteed robust: their training successfully terminated with zero property violation, enabling the corresponding real-time defense. CROWN-IBP clearly 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, which highlights the adversarial vulnerability of a property-agnostic model. SMiLE, on the other hand, while less accurate than the property-specific CROWN-IBP, exhibits only a moderate decline in clean, verified, and PGD accuracy as δ intensifies, demonstrating its ability to produce, with respect to a property-agnostic approach, significantly more robust, other than verifiable, networks. With respect to ϵ, instead, while the clean and PGD accuracy of SMiLE, for each fixed δ, tend to deteriorate, its verified performance does not show a clear trend, which suggests to keep this hyperparameter to the highest value among the ones considered.

Finally, while SMiLE may not match CROWN-IBP in terms of predictive accuracy, it clearly outperforms it in terms of runtime, as reported in Table 6.6, which presents the average training and inference time for the three models. As shown in the table, SMiLE can be trained in approximately half the time required by CROWN-IBP and, more remarkably, it can both predict and certify in roughly the same time as the standard Agnostic, whereas CROWN-IBP requires, for the same procedure, about two orders of magnitude (10^2) more.

6.4 Fairness

6.4.1 Benchmark

We consider 3 benchmarks, widely used in literature to assess AI fairness in social contexts: Compas, Law and Crime.

δ	**Model**	ϵ	**Clean**	**PGD**	**Verified**
0.01	Agnostic	–	99.60	99.50	0.00
0.01	CROWN-IBP	–	100.00	99.90	99.90
0.01	SMiLE	1.25	98.80	98.40	97.40
0.01	SMiLE	1.00	98.80	98.50	97.60
0.01	SMiLE	0.75	98.80	98.50	97.70
0.03	Agnostic	–	99.60	98.40	0.00
0.03	CROWN-IBP	–	99.80	99.70	99.60
0.03	SMiLE	1.25	98.80	98.10	96.00
0.03	SMiLE	1.00	98.60	97.90	96.70
0.03	SMiLE	0.75	98.40	97.70	96.40
0.05	Agnostic	–	99.60	90.40	0.00
0.05	CROWN-IBP	–	99.70	99.40	99.20
0.05	SMiLE	1.25	98.50	96.60	94.60
0.05	SMiLE	1.00	98.20	96.30	94.90
0.05	SMiLE	0.75	96.90	95.30	93.50
0.07	Agnostic	–	99.60	75.70	0.00
0.07	CROWN-IBP	–	99.60	99.20	99.20
0.07	SMiLE	1.25	97.70	95.20	92.40
0.07	SMiLE	1.00	97.30	94.40	92.50
0.07	SMiLE	0.75	96.20	93.00	92.20
0.10	Agnostic	–	99.60	66.80	0.00
0.10	CROWN-IBP	–	99.60	99.30	99.00
0.10	SMiLE	1.25	97.20	93.10	90.60
0.10	SMiLE	1.00	96.00	91.90	90.30
0.10	SMiLE	0.75	94.40	91.50	90.80

Table 6.5: Accuracy evaluation on the robustness benchmark

The Compas dataset [ALMK16] contains demographic and criminal history information about defendants from Broward County, Florida, and we use it for a binary classification task, consisting in predicting whether a defendant will reoffend within two years. The Law dataset [Wig98] includes demographic and academic information on law students, such as race, gender, LSAT scores and undergraduate GPA, and we consider it again for a binary classification task, consisting in deciding the admission of a candidate to a law school. Finally, the Crime dataset [RB02] combines socioeconomic, demographic and law enforcement data from U.S. communities, and we employ it for a regression task, consisting in estimating violent crime rate in a community.

Model	Training (s)	Inference (s)
Agnostic	136	0.02
CROWNIBP	14138	11.20
SMiLE	7717	0.03

Table 6.6: Runtime evaluation on the robustness benchmark.

Each dataset $\mathcal{D}$ is preprocessed by dropping any missing rows or columns, normalizing numerical features in between $[0, 1]$, and one-hot encoding categorical ones, before splitting it according to random 70%-30% criterion, to obtain a train and test set, $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$, respectively. Table 6.7 reports some basic statistics on the considered datasets.

Dataset	Size	Input	Task	Domain
Compas	3168	12	Binary Classification	Justice
Law	67578	33	Binary Classification	Education
Crime	1198	100	Regression	Crime

Table 6.7: Adopted datasets for the fairness benchmark.

For a SMiLE predictor f, we consider 5 ϵ-fairness properties, defined as

$$\forall x', x'' : \overbrace{|x'_i - x''_i| \leqslant \delta \ \forall i \neq s}^{Q(x', x'')} \implies \overbrace{|f(x') - f(x'')| \leqslant \epsilon}^{R(f(x'), f(x''))} \tag{6.12}$$

where $f(x)$, according to the benchmark, denotes either the logit or the final output, δ is fixed to 0, while $\epsilon \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ 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, with such values for ϵ chosen on the basis of a pilot experiment. In other words, we force a bound on the logit/output variation for any pair of *mutually counterfactual inputs*, that is, identical inputs with respect to all features, except for the *sensitive* (*protected*) one s. In particular, for all benchmarks, we set s to *race*, a binary feature indicating, in Compas and Law, whether the individual is Caucasian, while in Crime, whether the percentage of non-Caucasian individuals within the community exceed 6%. Note that this is a common setting in the fairness literature, as suggested by [QRI+22]. For the binary classification tasks, moreover, the final prediction of the model can be obtained by thresholding the produced logit against 0 via the indicator function $\mathbb{1}(f(x) > 0) \in \{0, 1\}$.

Similarly to the relational properties considered in the previous sections, the properties defined in Equation (6.12) can be equivalently specified according to the general definition of relational property provided in Equation (4.16), precisely:

$$\forall x', x'' \in \mathbb{R}^n : \begin{cases} \overbrace{0}^{\underline{\delta}_i} \leqslant x'_i - x''_i \leqslant \overbrace{0}^{\overline{\delta}_i} & \forall i \neq s \\ \underbrace{-M}_{\underline{\delta}_s} \leqslant x'_s - x''_s \leqslant \underbrace{M}_{\overline{\delta}_s} \end{cases} \implies \overbrace{-\epsilon}^{\underline{\epsilon}} \leqslant f(x') - f(x'') \leqslant \overbrace{\epsilon}^{\overline{\epsilon}} \quad (6.13)$$

where $\delta_i, \epsilon \geqslant 0$, and $M > 0$ is a sufficiently large value. In other words, also for this benchmark we can use the SMiLE relational enforcer (Algorithm 11) without any further grounding of its algorithmic components, namely, projector and generator.

6.4.2 Baseline

For this relational application, we evaluate our framework against CertiFair [KS23], a state-of-the-art fairness enforcer. This method penalizes property violation, by training the model against a convex combination of a standard and a fairness loss, computed via a sound but incomplete verifier. The method depends on a convex combination hyperparameter λ, while it is independent of both the counterfactual perturbation δ and the output bound ϵ.

At training time, similarly to SMiLE, CertiFair enforces fairness globally, i.e., on the entire input space. At inference time, however, the two methods differ significantly: while SMiLE, upon convergence, provides a bound on the model variation over any pair of counterfactual inputs, CertiFair lacks such formal satisfaction guarantees for the produced model. In other words, as an enforcement method, SMiLE is guaranteed, while CertiFair is heuristic.

Moreover, beyond the type of enforcement, the two methods also differ in generality, in particular in terms of supported properties: in contrast to SMiLE, indeed, CertiFair is only designed for fairness.

Finally, together with CertiFair, we also consider a base model unaware of the property to satisfy, which we refer to as Agnostic.

6.4.3 Accuracy & Counterfactual Variation

For each task, we train an Agnostic model in a property-agnostic fashion, 5 SMiLE models, one for each of the considered fairness properties, and 5 CertiFair models,

for different values of its main hyperparameter λ, controlling the degree of property enforcement, and calibrated on the basis of a pilot experiment to span the reasonable hyperparameter space: $\lambda \in \{0.025, 0.050, 0.075, 0.100, 0.125\}$ for Compas and Law, and $\lambda \in \{0.08, 0.09, 0.10, 0.11, 0.12\}$ for Crime.

All three approaches are trained on the training set $\mathcal{D}^{\text{train}}$, using the hyperparameters and architectures reported in Table 6.8 and Table 6.9. For CertiFair–specific hyperparameters, we instead rely on the default configuration provided by the authors in their original paper or implementation.

Parameter	**Value**		
	Compas	*Law*	*Crime*
Optimizer	Adam	Adam	Adam
Loss	BCE	BCE	MSE
Batch size	256	1024	128
Pretrain Epochs	50	50	50
Pretrain Stop Patience	–	–	–
Train Epochs	50	50	50
Generator Time Limit (s)	2	2	2
Generator Iteration Limit	8	8	8

Table 6.8: Training hyperparameters for the fairness benchmark.

Component	**Architecture**
h	$\text{relu}_{32} \circ \text{relu}_{32} \circ \text{linear}_8$
$\underline{h}, \overline{h}$	$\text{relu}_4 \circ \text{linear}_8$
g	linear_1

Table 6.9: Architectures for the fairness benchmark.

We evaluate each trained model f, on the test set $\mathcal{D}^{\text{test}}$, along two dimensions: *predictive quality*, denoted as $\text{PredQuality}(f)$ and corresponding to the standard R^2 (Equation (5.20)) or accuracy (i.e., percentage of correct predictions), according to the nature of the task, and *counterfactual variation*, defined instead as

$$\text{CounterVar}(f) = \max_{(x,y)\in\mathcal{D}^{\text{test}}} |f(x) - f(x^c)| \tag{6.14}$$

where x^c denotes the counterfactual of the input x, that is,

$$x_i^c = \begin{cases} x_i & \forall i \neq s \\ 1 - x_i & \forall i = s \end{cases} \tag{6.15}$$

In other words, CounterVar quantifies unfairness as the maximum absolute difference in model output/logit, on any pair of counterfactual samples from the test set, where the counterfactual of a sample is obtained by flipping its protected attribute. Note that the best reachable value is 1 for PredQuality(f), i.e., the higher the better, 0 for CounterVar(f), i.e., the lower the better.

We compare the three approaches in Figure 6.8, where each dot represents a model evaluated along the two dimensions, CounterVar(f) and PredQuality(f) on the x- and y-axis, respectively, so that the performance of a model decreases from the upper-left corner of the plot, corresponding to high accuracy and fairness, to the lower-right one, corresponding instead to the opposite situation. 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, with respect to its competitor. We highlight, in particular, that the reported SMiLE models, during their training, achieved again zero property violation, meaning that, besides decreasing the counterfactual variation in-distribution (on the test set), they are also able to guarantee a variation upper bound (i.e., the enforced ϵ) out of distribution.

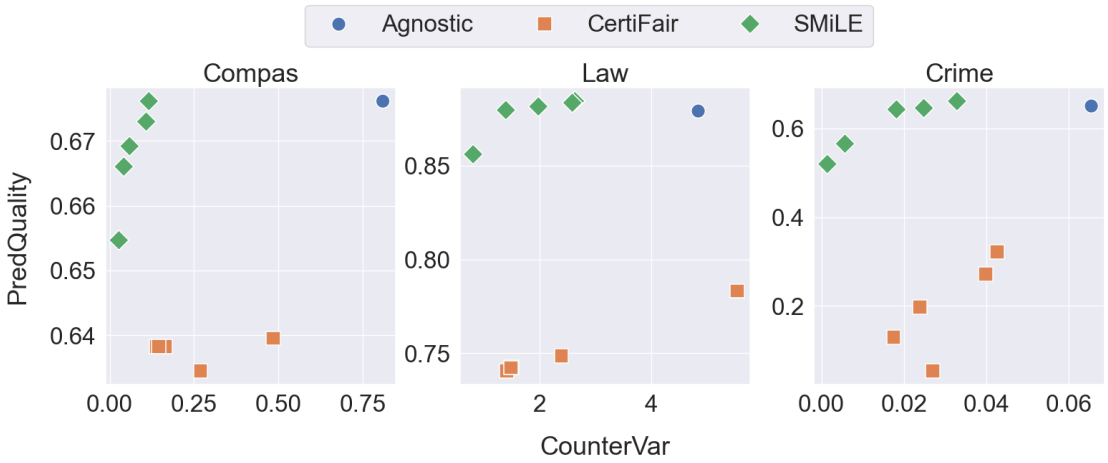
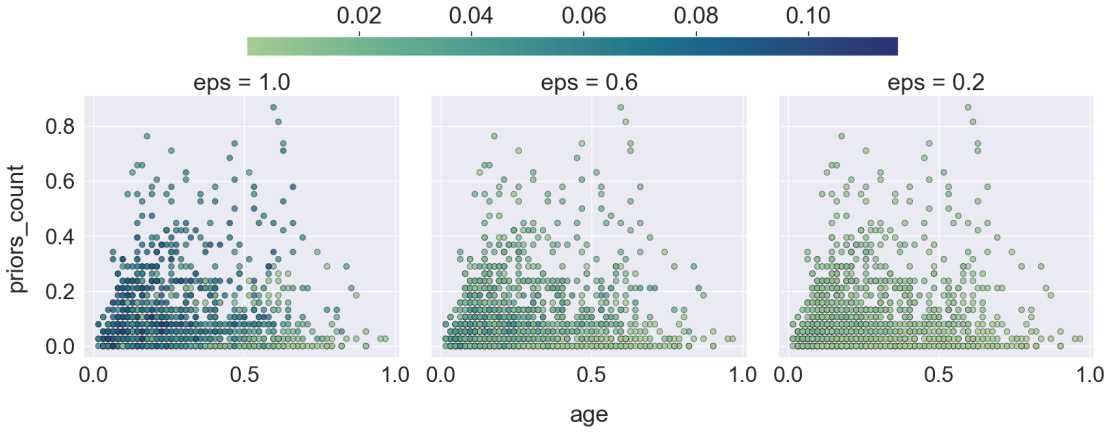
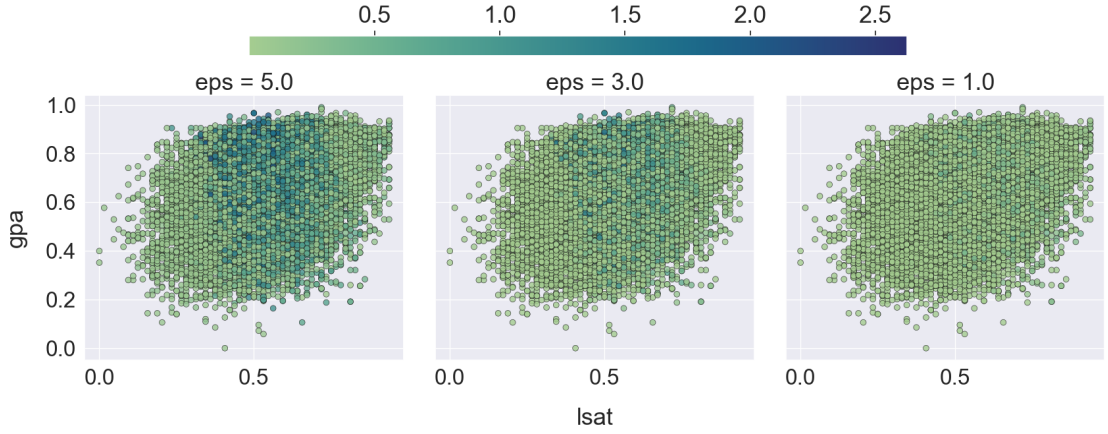


Figure 6.8: Variation and accuracy evaluation on the fairness benchmarks.

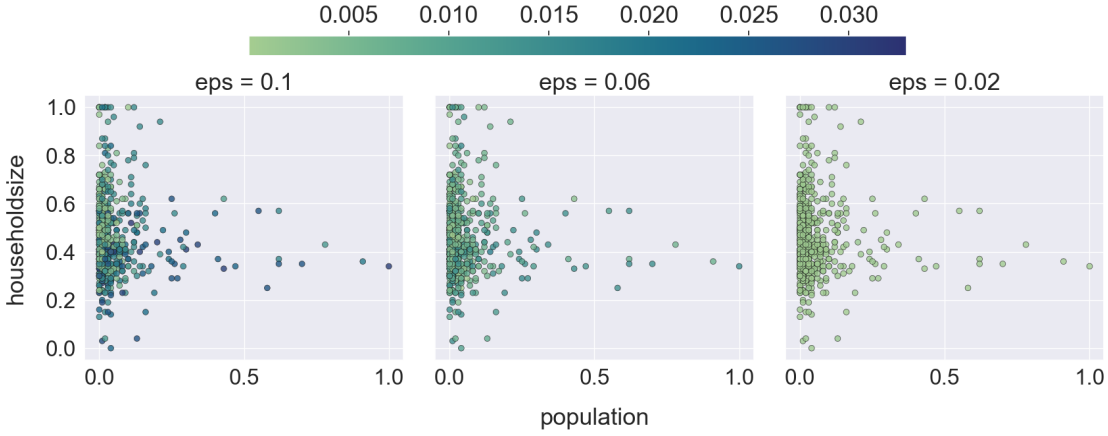
Finally, Figure 6.9 shows, for each benchmark, how the counterfactual variation of SMiLE (color gradient) progressively decreases with ϵ, over the pair of counterfactual samples (dots) projected on two dataset-specific non-protected features.



(a) Compas



(b) Law



(c) Crime

Figure 6.9: Counterfactual variation for different ϵ on the fairness benchmarks.

6.5 Discussion

We conclude this chapter by answering the research questions formalized in its introduction.

Q1 *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.

Q2 *Efficiency*: On the robustness benchmark, SMiLE offers a significant computational advantage over its property-aware competitor, both at training and inference time.

Q3 *Feasibility*: Each SMiLE model in our computational study successfully converged to zero property violation, demonstrating that, in practice, our framework can consistently provide satisfaction guarantees.

Q4 *Generality*: The strong results achieved across three relational properties, monotonicity, robustness, and fairness, and for different neural architectures, from shallow feedforward networks to deep convolutional ones, demonstrate the broad generality of SMiLE.

Q5 *Applicability*: On robustness and fairness benchmarks, 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.

Chapter 7

Conclusion

7.1 Final Remarks

In this thesis, we addressed the problem of enforcing user-defined properties into data-driven systems, to align them with the mechanics of the domain where they operate, or with the values of the society they interact with.

We first formalized the problem, discussed its complexity, and reviewed the relevant literature, presenting a taxonomy of scenarios, properties, verifiers, and enforcers. In particular, we identified key gaps in the current landscape of approaches, namely, their limited generality with respect to both properties and models, and their inability to provide formal satisfaction guarantees. To bridge these gaps, we introduced SMiLE (Safe Machine Learning via Embedded Overapproximation), a novel framework built upon the integration of machine learning and combinatorial optimization, and consisting of two main components: a verification-friendly neural architecture and a dedicated property-aware training algorithm. Finally, we conducted an extensive computational study across a diverse range of properties, models and tasks.

The obtained results demonstrated the ability of SMiLE to provably enforce the satisfaction of generic properties, from simple linear constraints to more demanding combinatorial or relational requirements, into arbitrary neural architectures, from shallow feedforward networks to more complex convolutional and recurrent structures, while matching the predictive performance of property-specific baselines. Moreover, SMiLE exhibited high robustness with respect to its key design choices, and high efficiency both at training and inference time. Overall, these findings confirm the ability of our framework to provide formal guarantees, as well as its generality and practical applicability.

7.2 Roadmap

Despite the encouraging results achieved by our method, we recognize its current limitations and identify several avenues for improvement, which we summarize in the following roadmap for future research.

Latent Dimension. The ablation study presented in the experimental section leaves several hyperparameters underexplored, notably the dimension of the latent space. Investigating different configurations for this design choice could lead to improved accuracy and efficiency.

Auxiliary Architecture. The auxiliary architecture currently adopted in SMiLE admits several variants, such as a factorized representation of the overapproximation, expressed as a combination of angle and magnitude. Implementing this variant could prevent the box degeneracy phenomenon.

Functional Properties. Our framework does not yet support functional properties that relate inputs and outputs, such as differential equations. Generalizing SMiLE to these properties would expand its compatibility to a variety of knowledge sources, such as physics.

Variable Input Size. The current implementation of SMiLE assumes fixed-size inputs. Generalizing SMiLE to inputs of variable size, such as text sequences of different lengths or graphs with different numbers of nodes, would extend its compatibility to various state-of-the-art architectures, such as Transformers and Graph Neural Networks.

Reinforcement Learning. Our framework currently supports only supervised learning tasks. Generalizing SMiLE to reinforcement learning settings would extend its compatibility to sequential decision-making.

Non-Neural Models. The core idea of SMiLE, i.e., bypassing the complexity of the target model via a simpler auxiliary mechanism, has been explored only for neural networks. Adapting it to other model families, such as tree- or kernel-based methods, could broaden the applicability and impact of our framework.

In conclusion, we believe that our work highlights the potential of integrating data-driven and knowledge-based paradigms to improve the compliance, safety, and trustworthiness of Artificial Intelligence, an essential step toward fostering its beneficial and responsible impact on our societies.

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: `https://www.tensorflow.org/`.

[ADAA25] Khaleel Ibrahim Al-Daoud and Ibrahim A. Abu-AlSondos. Robust ai for financial fraud detection in the gcc: A hybrid framework for imbalance, drift, and adversarial threats. *Journal of Theoretical and Applied Electronic Commerce Research*, 20(2), 2025. URL: `https://www.mdpi.com/0718-1876/20/2/121`, `doi:10.3390/jtaer20020121`.

[AHM+20] Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183:3–39, 2020. `doi:10.1007/s10107-020-01474-5`.

[ALMK16] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica*, 2016. URL: `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`.

[ARI+25] James Amarel, Christopher Rudolf, Athanasios Iliopoulos, John G. Michopoulos, and Leslie N. Smith. Symmetry constrained neural

networks for detection and localization of damage in metal plates. *APL Machine Learning*, 3(2), 2025. URL: http://dx.doi.org/10.1063/5.0242345, doi:10.1063/5.0242345.

[AVS+22] Diego Nieves Avendano, Nathan Vandermoortele, Colin Soete, Pieter Moens, Agusmian Partogi Ompusunggu, Dirk Deschrijver, and Sofie Van Hoecke. A semi-supervised approach with monotonic constraints for improved remaining useful life estimation. *Sensors*, 22(4):1590, 2022. URL: https://www.mdpi.com/1424-8220/22/4/1590, doi:10.3390/s22041590.

[B+16] Mariusz Bojarski et al. End to end learning for self-driving cars. In *arXiv preprint arXiv:1604.07316*, 2016. URL: https://arxiv.org/abs/1604.07316.

[BA25] Ajmal Rashid Bhat and Sumeer Ahmed. Artificial intelligence (ai) in drug design and discovery: A comprehensive review. *In Silico Research in Biomedicine*, 1:100049, 2025. URL: https://www.sciencedirect.com/science/article/pii/S3050787125000484, doi:10.1016/j.insi.2025.100049.

[BBC+19] Chris Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Ludwig Debiak, Carles Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[BBJW24] Christopher Jan-Steffen Brix, Stanley Bak, Taylor T. Johnson, and Haoze Wu. The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results. Technical Report RWTH-2025-02729, Universitätsbibliothek der RWTH Aachen, Aachen, Germany, 2024. URL: https://arxiv.org/abs/2412.19985, arXiv:2412.19985, doi:10.48550/arXiv.2412.19985.

[Ber06] Timo Berthold. Primal heuristics for mixed integer programs. Master's thesis, Technische Universität Berlin, 2006. URL: https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/1029.

[Ber14] Timo Berthold. *Heuristic algorithms in global MINLP solvers*. PhD thesis, Technische Universität Berlin, 2014. URL: https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/5448.

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings*

of the Fifth Annual Workshop on Computational Learning Theory, COLT '92, pages 144–152. Association for Computing Machinery, 1992. doi:10.1145/130385.130401.

[BHH+21] Michael L. Bynum, Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, third edition, 2021.

[BHL+10] Timo Berthold, Stefan Heinz, Marco Lübbecke, Rolf H. Möhring, and Jens Schulz. A constraint integer programming approach for resource-constrained project scheduling. In Andrea Lodi, Michela Milano, and Paolo Toth, editors, *Proc. of CPAIOR 2010*, volume 6140 of *LNCS*, pages 313–317. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-13520-0_34.

[BIL+16] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16. Curran Associates Inc., 2016.

[BJRL15] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[BLM15] Alessio Bonfietti, Michele Lombardi, and Michela Milano. Embedding decision trees and random forests in constraint programming. In Laurent Michel, editor, *Integration of AI and OR Techniques in Constraint Programming*, pages 74–90. Springer International Publishing, 2015.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. URL: https://arxiv.org/abs/2005.14165.

[BPW+22] Elias Benussi, Andrea Patanè, Matthew Wicker, Luca Laurenti, and Marta Kwiatkowska. Individual fairness guarantees for neural networks. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, pages 651–658. International Joint Conferences on Artificial Intelligence Organization, 2022. doi:10.24963/ijcai.2022/92.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. doi:10.1023/A:1010933404324.

[BS07] Ralf Borndörfer and Thomas Schlechte. Models for railway track allocation. In Christian Liebchen, Ravindra K. Ahuja, and Juan A. Mesa, editors, *7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'07)*, volume 7 of *OpenAccess Series in Informatics (OASIcs)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2007. doi:10.4230/OASIcs.ATMOS.2007.1170.

[BS24] Debangshu Banerjee and Gagandeep Singh. Relational dnn verification with cross executional bound refinement. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[bWAH+21] Umair bin Waheed, Tariq Alkhalifah, Ehsan Haghighat, Chao Song, and Jean Virieux. Pinntomo: Seismic tomography using physics-informed neural networks, 2021. URL: https://arxiv.org/abs/2104.01588, arXiv:2104.01588.

[BXS24] Debangshu Banerjee, Changming Xu, and Gagandeep Singh. Input-relational verification of deep neural networks. *Proc. ACM Program. Lang.*, 8(PLDI), 2024. doi:10.1145/3656377.

[C+15] François Chollet et al. Keras. https://keras.io, 2015.

[Cal24] California State Legislature. California ai transparency act (sb 942). https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=202320240SB942, 2024. Requires disclosure of AI-generated content.

[CJH+22] F. Ceccon, J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C. D Laird, and R. Misener. Omlt: Optimization & machine learning toolkit. *Journal of Machine Learning Research*, 23(349):1–8, 2022.

[Col24] Colorado General Assembly. Colorado artificial intelligence act (senate bill 24-205). https://leg.colorado.gov/sites/default/files/2024a_205_signed.pdf, 2024. Official text of the bill.

[CRK19] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of*

Machine Learning Research, pages 1310–1320. PMLR, 2019. URL: https://proceedings.mlr.press/v97/cohen19c.html.

[CSKea24] A. K. Chew, M. Sender, Z. Kaplan, and et al. Advancing material property prediction: using physics-informed machine learning models for viscosity. *Journal of Cheminformatics*, 16:31, 2024. doi:10.1186/s13321-024-00820-5.

[CvMBB14] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[CW16] Taco S. Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2990–2999. JMLR.org, 2016.

[CW17] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE Computer Society, 2017. URL: https://doi.ieeecomputersociety.org/10.1109/SP.2017.49, doi:10.1109/SP.2017.49.

[CZ24] Shuyi Chen and Shixiang Zhu. Counterfactual fairness through transforming data orthogonal to bias. *arXiv preprint arXiv:2403.17852*, 2024. version 2. URL: https://arxiv.org/abs/2403.17852.

[Dat20] Leonid Datta. A survey on activation functions and their relation with xavier and he normal initialization, 2020. arXiv:2004.06632.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[DFK16] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*,

volume 48 of *Proceedings of Machine Learning Research*, pages 1889–1898. PMLR, 2016. URL: https://proceedings.mlr.press/v48/dieleman16.html.

[DGP+24] Stefano Demarchi, Daniele Guidotti, Luca Pulina, et al. Never2: learning and verification of neural networks. *Soft Computing*, 28:11647–11665, 2024. doi:10.1007/s00500-024-09907-5.

[DIP+24] I. Diakou, E. Iliopoulos, E. Papakonstantinou, K. Dragoumani, C. Yapijakis, C. Iliopoulos, D. A. Spandidos, G. P. Chrousos, E. Eliopoulos, and D. Vlachakis. Multi-label classification of biomedical data. *Medicine International (London)*, 4(6):68, 2024. doi:10.3892/mi.2024.192.

[dLMVW22] Petter Eilif de Lange, Borger Melsom, Christian Bakke Vennerød, and Sjur Westgaard. Explainable ai for credit assessment in banks. *Journal of Risk and Financial Management*, 15(12), 2022. URL: https://www.mdpi.com/1911-8074/15/12/556, doi:10.3390/jrfm15120556.

[Elm90] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. URL: https://www.sciencedirect.com/science/article/pii/036402139090002E, doi:10.1016/0364-0213(90)90002-E.

[Eur17] European Union. Regulation (eu) 2017/745 of the european parliament and of the council on medical devices, 2017. URL: https://eur-lex.europa.eu/eli/reg/2017/745/oj/eng.

[Eur24] European Union. Eu artificial intelligence act, 2024. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52021PC0206.

[FJ18] M. Fischetti and J. Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23:296–309, 2018. doi:10.1007/s10601-018-9285-6.

[FMK+21] Ivan Fursov, Matvey Morozov, Nina Kaploukhaya, Elizaveta Kovtun, Rodrigo Rivera-Castro, Gleb Gusev, Dmitry Babaev, Ivan Kireev, Alexey Zaytsev, and Evgeny Burnaev. Adversarial attacks on deep models for financial transaction records. *CoRR*, abs/2106.08361, 2021. URL: https://arxiv.org/abs/2106.08361, arXiv:2106.08361.

[GAK19] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2221–2231, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3292500.3330691.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GDB24] Kshitij Goyal, Sebastijan Dumancic, and Hendrik Blockeel. Deepsade: learning neural networks that guarantee domain constraint satisfaction. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. doi:10.1609/aaai.v38i11.29109.

[GDS+19] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models, 2019. URL: https://arxiv.org/abs/1810.12715, arXiv:1810.12715.

[GKM+24] Robert X. Gao, Jörg Krüger, Marion Merklein, Hans-Christian Möhring, and József Váncza. Artificial intelligence in manufacturing: State of the art, perspectives, and future directions. *CIRP Annals*, 73(2):723–749, 2024. URL: https://www.sciencedirect.com/science/article/pii/S000785062400115X, doi:10.1016/j.cirp.2024.04.101.

[GKR24] Kévin Garanger, Julie Kraus, and Julian J. Rimoli. Symmetry-enforcing neural networks with applications to constitutive modeling. *Extreme Mechanics Letters*, 71:102188, 2024. URL: https://www.sciencedirect.com/science/article/pii/S2352431624000683, doi:10.1016/j.eml.2024.102188.

[GL21] Eleonora Giunchiglia and Thomas Lukasiewicz. Multi-label classification neural networks with hard logical constraints. *Journal of Artificial Intelligence Research*, 72:1–34, 2021. doi:10.1613/jair.1.12850.

[GMDC+18] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2018. doi:10.1109/SP.2018.00058.

[Gom58] Ralph E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958. doi:978-3-540-68279-0_4.

[Goo] Google or-tools. URL: https://developers.google.com/optimization.

[GPSB19] Kathrin Grosse, David Pfaff, Michael Thomas Smith, and Michael Backes. The limitations of model uncertainty in adversarial settings, 2019. URL: https://arxiv.org/abs/1812.02606, arXiv:1812.02606.

[GQC+20] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. In *Proceedings of Interspeech 2020*, 2020. URL: https://arxiv.org/abs/2005.08100.

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[Gur] Gurobi Optimization. URL: https://www.gurobi.com.

[HCR+22] Lars Hofeditz, Sven Clausen, Andreas Rieß, Susanne Müller, Daniel Klöß, Marco Mevius, and Frank Pallas. Applying XAI to an AI-based system for candidate management to mitigate bias and discrimination in hiring. *Electronic Markets*, 32(5):2207–2233, 2022. doi:10.1007/s12525-022-00600-9.

[HH25] Manzoor Hussain and Jang-Eui Hong. Evaluating and improving adversarial robustness of deep learning models for intelligent vehicle safety. *IEEE Transactions on Reliability*, 74(4):4559–4573, 2025. doi:10.1109/TR.2024.3458805.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HWW11] William E Hart, Jean-Paul Watson, and David L Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

[HYL+19] Xiang He, Sibei Yang, Guanbin Li, Haofeng Li, Huiyou Chang, and Yizhou Yu. Non-local context encoder: robust biomedical image segmentation against adversarial attacks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. doi:10.1609/aaai.v33i01.33018417.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. URL: https://arxiv.org/abs/1512.03385.

[Int06] International Electrotechnical Commission. Iec 62304: Medical device software — software life cycle processes. Technical Report IEC 62304:2006, International Electrotechnical Commission, Geneva, Switzerland, 2006. URL: https://www.iso.org/standard/38421.html.

[Int25] International Bar Association. Japan's emerging framework for responsible ai: legislation, guidelines and guidance, 2025. URL: https://www.ibanet.org/japan-emerging-framework-ai-legislation-guidelines.

[IRD+25] F.G. Iampi, A. Rega, T.M.L. Diallo, et al. Analysing the role of physics-informed neural networks in modelling industrial systems through case studies in automotive manufacturing. *International Journal of Interactive Design and Manufacturing (IJIDeM)*, 2025. doi:10.1007/s12008-025-02364-w.

[ISO18] ISO/TC 22/SC 32. Road vehicles — functional safety — part 1: Vocabulary. Standard ISO 26262-1:2018, International Organization for Standardization, 2018. URL: https://www.iso.org/standard/68383.html.

[ISO19] ISO/TC 22/SC 32. Road vehicles — safety of the intended functionality. Standard ISO/PAS 21448:2019, International Organization

for Standardization, 2019. URL: https://www.iso.org/standard/77490.html.

[JAK25] R. Jayaweera, H. Agrawal, and N.M. Karie. Federated security for privacy preservation of healthcare data in edge-cloud environments. *Sensors (Basel)*, 25(16), 2025. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC12390488/, doi:10.3390/s25165108.

[JQC+25] Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Lukas Vierling, Donghai Hong, Jiayi Zhou, Zhaowei Zhang, Fanzhi Zeng, Juntao Dai, Xuehai Pan, Kwan Yee Ng, Aidan O'Gara, Hua Xu, Brian Tse, Jie Fu, Stephen McAleer, Yaodong Yang, Yizhou Wang, Song-Chun Zhu, Yike Guo, and Wen Gao. Ai alignment: A comprehensive survey, 2025. URL: https://arxiv.org/abs/2310.19852, arXiv:2310.19852.

[JR20] Kai Jia and Martin Rinard. Efficient exact verification of binarized neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc., 2020.

[JVW+25] Yan Jia, Clare Verrill, Kieron White, Monica Dolton, Margaret Horton, Mufaddal Jafferji, and Ibrahim Habli. A deployment safety case for ai-assisted prostate cancer diagnosis. *Computers in Biology and Medicine*, 192:110237, 2025. URL: https://www.sciencedirect.com/science/article/pii/S0010482525005888, doi:10.1016/j.compbiomed.2025.110237.

[JWK+20] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan S. Read, Jacob A. Zwart, Michael Steinbach, and Vipin Kumar. Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles. *ACM/IMS Transactions on Data Science*, 1(1):1–25, 2020. doi:10.1145/3447814.

[Kal24] Katikapalli Subramanyam Kalyan. A survey of gpt-3 family large language models including chatgpt and gpt-4. *Natural Language Processing Journal*, 6:100048, 2024. URL: https://www.sciencedirect.com/science/article/pii/S2949719123000456, doi:10.1016/j.nlp.2023.100048.

[Kar39] William Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, University of Chicago, 1939. Master's Thesis.

[Kar72] Richard Karp. Reducibility among combinatorial problems. In Raymond Miller and James Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972. doi:10.1007/978-1-4684-2001-2_9.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KBD+17] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification (CAV)*, pages 97–117. Springer, 2017. doi:10.1007/978-3-319-63387-9_5.

[KC12] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, 2012. doi:10.1007/s10115-011-0463-8.

[KKKS25] Kadir Kesgin, Salih Kiraz, Selahattin Kosunalp, and Bozhana Stoycheva. Beyond performance: Explaining and ensuring fairness in student academic performance prediction with machine learning. *Applied Sciences*, 15(15):8409, 2025. doi:10.3390/app15158409.

[KLB+25] Konstantin Kaulen, Tobias Ladner, Stanley Bak, Christopher Brix, Hai Duong, Thomas Flinkow, Taylor T. Johnson, Lukas Koller, Edoardo Manino, ThanhVu H Nguyen, and Haoze Wu. The 6th international verification of neural networks competition (vnn-comp 2025): Summary and results, 2025. URL: https://arxiv.org/abs/2512.19007, arXiv:2512.19007.

[KRBA16] Jim Y. J. Kuo, David A. Romero, J. Christopher Beck, and Cristina H. Amon. Wind farm layout optimization on complex terrains–integrating a cfd wake model with mixed-integer programming. *Applied Energy*, 178:404–414, 2016. doi:10.1016/j.apenergy.2016.06.085.

[Kri23] Moez Krichen. Convolutional neural networks: A survey. *Computers*, 12(8), 2023. URL: https://www.mdpi.com/2073-431X/12/8/151, doi:10.3390/computers12080151.

[KS23] Haitham Khedr and Yasser Shoukry. Certifair: A framework for certified global fairness of neural networks. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)*, pages 1–9. AAAI Press, 2023. URL: https://arxiv.org/abs/2205.09927.

[KT51] Harold W. Kuhn and Albert W. Tucker. Nonlinear programming. In Jerzy Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, CA, 1951. University of California Press.

[LAL+21] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, and Mykel J. Kochenderfer. Algorithms for verifying deep neural networks. *Found. Trends Optim.*, 4(3–4):244–404, 2021. doi:10.1561/2400000035.

[LB08] Fabien Lauer and Gérard Bloch. Incorporating prior knowledge in support vector machines for classification: A review. *Neurocomput.*, 71(7–9):1578–1594, 2008. doi:10.1016/j.neucom.2007.04.010.

[LBBH98a] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:10.1109/5.726791.

[LBBH98b] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LD60] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. doi:10.2307/1910129.

[LDD18] Jesus Lago, Fjo De Ridder, and Bart De Schutter. Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms. *Applied Energy*, 221:386–405, 2018. URL: https://www.sciencedirect.com/science/article/pii/S030626191830196X, doi:10.1016/j.apenergy.2018.02.069.

[LG16] Michele Lombardi and Stefano Gualandi. A lagrangian propagator for artificial neural networks in constraint programming. *Constraints*, 21:435–462, 2016. doi:10.1007/s10601-015-9234-6.

[LGW19] Preethi Lahoti, Krishna P. Gummadi, and Gerhard Weikum. Operationalizing individual fairness with pairwise fair representations. *Proc. VLDB Endow.*, 13(4):506–518, 2019. doi:10.14778/3372716.3372723.

[LHZL20] Xingchao Liu, Xing Han, Na Zhang, and Qiang Liu. Certified monotonic neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[LKB+17] Geert Litjens, Thijs Kooi, Babak E. Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017. doi:10.1016/j.media.2017.07.005.

[LLL+19] Zihao Liu, Qi Liu, Tao Liu, Nuo Xu, Xue Lin, Yanzhi Wang, and Wujie Wen. Feature distillation: Dnn-oriented jpeg compression against adversarial examples, 2019. URL: https://arxiv.org/abs/1803.05787, arXiv:1803.05787.

[LM18] Michele Lombardi and Michela Milano. Boosting combinatorial problem modeling with machine learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 5472–5478. AAAI Press, 2018. doi:https://dl.acm.org/doi/10.5555/3304652.3304786.

[LMB17] Michele Lombardi, Michela Milano, and Andrea Bartolini. Empirical decision model learning. *Artificial Intelligence*, 244:343–367, 2017. doi:10.1016/j.artint.2016.01.005.

[LPZ21] Xin Li, Deng Pan, and Dongxiao Zhu. Defending against adversarial attacks on medical imaging ai system, classification or detection? In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 1677–1681, 2021. doi:10.1109/ISBI48211.2021.9433761.

[LRP19] Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019. URL: https://openreview.net/pdf?id=BklHpjCqKm.

[LS25] Giacomo Lastrucci and Artur M. Schweidtmann. Enforce: Nonlinear constrained learning with adaptive-depth neural projection, 2025. URL: https://arxiv.org/abs/2502.06774, arXiv:2502.06774.

[LWF21] Klas Leino, Zifan Wang, and Matt Fredrikson. Globally-robust neural networks. In *International Conference on Machine Learning (ICML)*, 2021.

[LWF+22] Haochen Liu, Yiqi Wang, Wenqi Fan, Xiaorui Liu, Yaxin Li, Shaili Jain, Yunhao Liu, Anil Jain, and Jiliang Tang. Trustworthy ai:

A computational perspective. *ACM Trans. Intell. Syst. Technol.*, 14(1), 2022. doi:10.1145/3546872.

[LYLC18] Junying Li, Zichen Yang, Haifeng Liu, and Deng Cai. Deep rotation equivariant network. *Neurocomputing*, 290:26–33, 2018. URL: https://www.sciencedirect.com/science/article/pii/S0925231218301644, doi:10.1016/j.neucom.2018.02.029.

[LZD+25] Ke Li, Chenyu Zhang, Yuxin Ding, Xianbiao Hu, and Ruwen Qin. Multi-label scene classification for autonomous vehicles: Acquiring and accumulating knowledge from diverse datasets, 2025. URL: https://arxiv.org/abs/2506.17101, arXiv:2506.17101.

[LZMW09] Fuhai Li, Xiaobo Zhou, Jinwen Ma, and Stephen T. C. Wong. Multiple nuclei tracking using integer programming for quantitative cancer cell cycle analysis. *IEEE transactions on medical imaging*, 29(1):96–105, 2009. doi:10.1109/TMI.2009.2027813.

[LZSJ19] Xuankang Lin, He Zhu, Roopsha Samanta, and Suresh Jagannathan. ART: abstraction refinement-guided training for provably correct neural networks. *CoRR*, abs/1907.10662, 2019. URL: http://arxiv.org/abs/1907.10662, arXiv:1907.10662.

[MA25] Youngjae Min and Navid Azizan. Hardnet: Hard-constrained neural networks with universal approximation guarantees, 2025. URL: https://arxiv.org/abs/2410.10807, arXiv:2410.10807.

[MDFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016. doi:10.1109/CVPR.2016.282.

[MGC+25] C. Meng, S. Griesemer, D. Cao, S. Seo, and Y. Liu. When physics meets machine learning: A survey of physics-informed machine learning. *Machine Learning for Computational Science and Engineering*, 1:20, 2025. doi:10.1007/s44379-025-00016-0.

[MIM+18] Nikhil Muralidhar, Mohammad Raihanul Islam, Manish Marwah, Anuj Karpatne, and Naren Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 36–45, 2018. doi:10.1109/BigData.2018.8621955.

[ML23] Samuele Marro and Michele Lombardi. Computational asymmetries in robust classification. In *International Conference on Machine Learning*, pages 24082–24138. PMLR, 2023.

[MMS+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018. URL: https://arxiv.org/abs/1706.06083.

[MMS+19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019. URL: https://arxiv.org/abs/1706.06083, arXiv:1706.06083.

[MMS+22] Mark Niklas Müller, Gleb Makarchuk, Gagandeep Singh, Markus Püschel, and Martin Vechev. Prima: general and precise neural network certification via scalable convex hull approximations. *Proc. ACM Program. Lang.*, 6(POPL), 2022. doi:10.1145/3498704.

[MMWW02] Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1):397 – 446, 2002. doi:10.1016/S0166-218X(01)00348-1.

[Moy24] Jose M. Moyano. Multi-label classification dataset repository, 2024. URL: https://www.uco.es/kdis/mllresources/#Read2010.

[MP43] W. S. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[MSA20] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. doi:10.1016/j.ijforecast.2019.04.014.

[MSF23] Kiarash Mohammadi, Aishwarya Sivaraman, and Golnoosh Farnadi. 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, 2023. doi:10.1145/3617694.3623243.

[MSO24] Ibomoiye Domor Mienye, Theo G. Swart, and George Obaido. Recurrent neural networks: A comprehensive review of architectures, variants, and applications. *Information*, 15(9), 2024. URL: https://www.mdpi.com/2078-2489/15/9/517, doi:10.3390/info15090517.

[MSR+24] Giulia Millevoi, Gabriele Santin, Martina Rama, Michele Tizzoni, and Bruno Lepri. Bridging physics and machine learning: Exploring seir epidemiological modeling using physics-informed neural networks. *arXiv preprint arXiv:2506.03897*, 2024. URL: https://arxiv.org/abs/2506.03897.

[MT25] Ashutosh K. Mishra and Emma Tolley. Spinn: Advancing cosmological simulations of fuzzy dark matter with physics informed neural networks. *The Astrophysical Journal*, 988(1), jul 2025. URL: https://dx.doi.org/10.3847/1538-4357/ade43e, doi:10.3847/1538-4357/ade43e.

[MWK+23] Aurelien Meray, Lijing Wang, Takuya Kurihana, Ilijana Mastilovic, Satyarth Praveen, Zexuan Xu, Alexander Lavin, Milad Memarzadeh, and Haruko Wainwright. Physics-informed surrogate modeling for supporting climate resilience at groundwater contamination sites. *Computers & Geosciences*, 183:105508, 2023. doi:10.1016/j.cageo.2023.105508.

[MZB+23] Sajid Mannan, Mohd Zaki, Suresh Bishnoi, Daniel R. Cassar, Jeanini Jiusti, Julio Cesar Ferreira Faria, Johan F.S. Christensen, Nitya Nand Gosvami, Morten M. Smedskjaer, Edgar Dutra Zanotto, and N.M. Anoop Krishnan. Glass hardness: Predicting composition and load effects via symbolic reasoning-informed machine learning. *Acta Materialia*, 255:119046, 2023. doi:10.1016/j.actamat.2023.119046.

[Nat23] National Institute of Standards and Technology. Ai risk management framework (ai rmf), 2023. URL: https://www.nist.gov/itl/ai-risk-management-framework.

[NTF22] Géraldin Nanfack, Paul Temple, and Benoît Frénay. Constraint enforcement on decision trees: A survey. *ACM Comput. Surv.*, 54(10), 2022. doi:10.1145/3506734.

[OEC19] OECD. Oecd ai principles, 2019. URL: https://www.oecd.org/going-digital/ai/principles/.

[PAU24] Mandar Pitale, Alireza Abbaspour, and Devesh Upadhyay. Inherent diverse redundant safety mechanisms for ai-based software elements in automotive applications, 2024. URL: https://arxiv.org/abs/2402.08208, arXiv:2402.08208.

[PB+14] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.

[PDD+25] Nga Pham, Minh Kha Do, Tran Vu Dai, Pham Ngoc Hung, and Anh Nguyen-Duc. Fairedu: A multiple regression-based method for enhancing fairness in machine learning models for educational applications. *Expert Systems with Applications*, 269:126219, 2025. URL: https://www.sciencedirect.com/science/article/pii/S0957417424030860, doi:10.1016/j.eswa.2024.126219.

[PHC+25] Balamurugan Palanisamy, Vikas Hassija, Arpita Chatterjee, Arpita Mandal, Debanshi Chakraborty, Amit Pandey, G. S. S. Chalapathi, and Dhruv Kumar. Transformers for vision: A survey on innovative methods for computer vision. *IEEE Access*, 13:95496–95523, 2025. doi:10.1109/ACCESS.2025.3571735.

[PRK23] I. Perez-Raya and S. G. Kandlikar. Thermal modeling of patient-specific breast cancer with physics-based artificial intelligence. *ASME Journal of Heat Transfer*, 145(3), 2023. doi:10.1115/1.4055347.

[PSMF20] David Pfau, James S. Spencer, Alexander G. D. G. Matthews, and W. M. C. Foulkes. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Phys. Rev. Res.*, 2:033429, Sep 2020. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.2.033429, doi:10.1103/PhysRevResearch.2.033429.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[QMB+25] Ying Qian, Éric Marty, Avranil Basu, Eamon B. O'Dea, Xianqiao Wang, Spencer Fox, Pejman Rohani, John M. Drake, and He Li. Physics-informed deep learning for infectious disease forecasting. *arXiv preprint arXiv:2501.09298*, 2025. URL: https://arxiv.org/abs/2501.09298.

[QRI+22] Tai Le Quy, Arjun Roy, Vasileios Iosifidis, Wenbin Zhang, and Eirini Ntoutsi. A survey on datasets for fairness-aware machine learning. *WIREs Data Mining and Knowledge Discovery*, 12(3), 2022. doi:10.1002/widm.1452.

[QRLB20] Facundo Quiroga, Franco Ronchetti, Laura Lanzarini, and Aurelio F. Bariviera. Revisiting data augmentation for rotational invariance in convolutional neural networks. In Joan Carles Ferrer-Comalat, Salvador Linares-Mustarós, José M. Merigó, and Janusz Kacprzyk, editors, *Modelling and Simulation in Management Sciences*, pages 127–141, Cham, 2020. Springer International Publishing.

[QSC+17] Yao Qin, Dongjin Song, Haifeng Chen, Weinan Cheng, Guokun Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2627–2633, 2017. URL: https://www.ijcai.org/Proceedings/2017/0373.pdf.

[RB02] Michael Redmond and Amit Baveja. Communities and crime data set. https://archive.ics.uci.edu/ml/datasets/communities+and+crime, 2002. UCI Machine Learning Repository.

[RCR+23] Alexander Rodriguez, Jiaming Cui, Naren Ramakrishnan, Bijaya Adhikari, and B. Aditya Prakash. Einns: Epidemiologically-informed neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 2023. URL: https://arxiv.org/abs/2202.10446.

[RFLM23] Jan G. Rittig, Kobi C. Felton, Alexei A. Lapkin, and Alexander Mitsos. Gibbs–duhem-informed neural networks for binary activity coefficient prediction. *Digital Discovery*, 2:1752–1767, 2023. doi:10.1039/D3DD00103B.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi:10.1038/323533a0.

[RK22] Idan Refaeli and Guy Katz. Minimal multi-layer modifications of deep neural networks. In Omri Isac, Radoslav Ivanov, Guy Katz, Nina Narodytska, and Laura Nenzi, editors, *Software Verification*

and Formal Methods for ML-Enabled Autonomous Systems, pages 46–66. Springer International Publishing, 2022.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. doi:10.1037/h0042519.

[RPK19] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi:10.1016/j.jcp.2018.10.045.

[RS23] Davor Runje and Sharath M Shankaranarayana. Constrained monotonic neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[Rud17] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017. URL: https://arxiv.org/abs/1609.04747, arXiv:1609.04747.

[SBM24] Claudia Sirocchi, Alessandro Bogliolo, and Silvia Montagna. Medical-informed machine learning: integrating prior knowledge into medical decision systems. *BMC Medical Informatics and Decision Making*, 24(Suppl 4):186, 2024. doi:10.1186/s12911-024-02582-4.

[SCI] SCIP. URL: https://www.scipopt.org.

[SCYP+20] Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E. Hurtado, and Ellen Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42, 2020. doi:10.3389/fphy.2020.00042.

[SDAM24] Ashkan Safari, Mohammadreza Daneshvar, and Amjad Anvari-Moghaddam. Energy intelligence: A systematic review of artificial intelligence for energy management. *Applied Sciences*, 14(23), 2024. URL: https://www.mdpi.com/2076-3417/14/23/11112, doi:10.3390/app142311112.

[SFMVdB20] Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein, and Guy Van den Broeck. Counterexample-guided learning of monotonic neural networks. In *Proceedings of the 34th International Conference*

on Neural Information Processing Systems, NIPS '20. Curran Associates Inc., 2020.

[SGM+18] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 10825–10836. Curran Associates Inc., 2018.

[SGPW20] Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020. doi:10.1016/j.cma.2019.112732.

[SGT+09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Guido Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi:10.1109/TNN.2008.2005605.

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[SJ18] Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2219–2228, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3219819.3220088.

[SKN+18] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples, 2018. URL: https://arxiv.org/abs/1710.10766, arXiv:1710.10766.

[SKR20] Thiago Serra, Abhinav Kumar, and Srikumar Ramalingam. Lossless compression of deep neural networks. In Emmanuel Hebrard and Nysret Musliu, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 417–430. Springer International Publishing, 2020.

[SLB+20] Khemraj Shukla, Patricio Clark Di Leoni, James Blackshire, Daniel Sparkman, and George Em Karniadakis. Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. *Journal of Nondestructive Evaluation*, 39(3):61, 2020. doi:10.1007/s10921-020-00705-1.

[SLX+24] Jasraj Singh, Fang Liu, Hong Xu, Bee Chin Ng, and Wei Zhang. Lingml: Linguistic-informed machine learning for enhanced fake news detection, 2024. URL: https://arxiv.org/abs/2405.04165, arXiv:2405.04165.

[SMLD18] Yasser Shoukry, Shaunak Mishra, Zutian Luo, and Suhas Diggavi. Sybil attack resilient traffic networks: A physics-based trust propagation approach. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 43–54, 2018. doi:10.1109/ICCPS.2018.00013.

[SQL+24] Lu Shi, Bin Qi, Jiarui Luo, Yang Zhang, Zhanzhao Liang, Zhaowei Gao, Wenke Deng, and Lin Sun. Aegis:an advanced llm-based multi-agent for intelligent functional safety engineering, 2024. URL: https://arxiv.org/abs/2410.12475, arXiv:2410.12475.

[SRD+25] Vinitra Swamy, Davide Romano, Bhargav Srinivasa Desikan, Oana-Maria Camburu, and Tanja Käser. illuminate: an llm-xai framework leveraging social science explanation theories towards actionable student performance feedback. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. doi:10.1609/aaai.v39i27.35065.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[SSK+20] Sourabh Shastri, Kuljeet Singh, Sachin Kumar, Paramjit Kour, and Vibhakar Mansotra. Time series forecasting of covid-19 using deep learning models: India-usa comparative case study. *Chaos, Solitons & Fractals*, 140:110227, 2020. URL: https://www.sciencedirect.com/science/article/pii/S0960077920306238, doi:10.1016/j.chaos.2020.110227.

[SSO+19] Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2019. doi:10.1109/ICRA.2019.8794351.

[ST21] Matthew Sotoudeh and Aditya V. Thakur. Provable repair of deep neural networks. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021. Association for Computing Machinery, 2021. doi:10.1145/3453483.3454064.

[STR18] Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4558–4566. PMLR, 2018.

[SYY+23] Yi Sheng, Junhuan Yang, Lei Yang, Yiyu Shi, Jingtongf Hu, and Weiwen Jiang. Muffin: A framework toward multi-dimension ai fairness by uniting off-the-shelf models, 2023. URL: https://arxiv.org/abs/2308.13730.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2015. URL: https://arxiv.org/abs/1409.1556.

[TKM+21] Alexander Thebelt, Jan Kronqvist, Miten Mistry, Robert M. Lee, Nathan Sudermann-Merx, and Ruth Misener. Entmoot: A framework for optimization over ensemble tree models. *Computers & Chemical Engineering*, 151:107343, 2021. doi:10.1016/j.compchemeng.2021.107343.

[TKTM21] Calvin Tsay, Jan Kronqvist, Alexander Thebelt, and Ruth Misener. Partition-based formulations for mixed-integer optimization of trained reLU neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[TMLR+24] Anne M Tumlin, Diego Manzanas Lopez, Preston Robinette, Yuying Zhao, Tyler Derr, and Taylor T Johnson. Fairnnv: The neu-

ral network verification tool for certifying fairness. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF '24, pages 36–44. Association for Computing Machinery, 2024. doi:10.1145/3677052.3698677.

[Toc76] L. J. Tockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[TSK+18] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018. URL: https://arxiv.org/abs/1802.08219.

[Tur49] Alan Turing. On checking a large routine. In *Report of a Conference on High Speed Automatic Calculating Machines*, pages 67–69, 1949.

[TXT17] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

[Uni20] United States Congress. Artificial intelligence initiative act, 2020. URL: https://www.congress.gov/bill/116th-congress/house-bill/6216.

[vRMB+23] Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, Michal Walczak, Jochen Garcke, Christian Bauckhage, and Jannis Schuecker. Informed machine learning – a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):614–633, 2023. doi:10.1109/TKDE.2021.3079836.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL: https://arxiv.org/abs/1706.03762.

[WGTB17] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *2017 IEEE Conference on Computer*

Vision and Pattern Recognition (CVPR), pages 7168–7177, 2017. doi:10.1109/CVPR.2017.758.

[Wig98] Linda F. Wightman. Lsac national longitudinal bar passage study. Technical report, Law School Admission Council, Newtown, PA, 1998.

[WIZ+24] Haoze Wu, Omri Isac, Aleksandar Zeljić, Teruhiro Tagomori, Matthew Daggitt, Wen Kokke, Idan Refaeli, Guy Amir, Kyle Julian, Shahaf Bassan, Pei Huang, Ori Lahav, Min Wu, Min Zhang, Ekaterina Komendantskaya, Guy Katz, and Clark Barrett. Marabou 2.0: A versatile formal analyzer of neural networks. In Arie Gurfinkel and Vijay Ganesh, editors, *Computer Aided Verification*, pages 249–264. Springer Nature Switzerland, 2024.

[WVP18] Christina Wadsworth, Francesca Vera, and Chris Piech. Achieving fairness through adversarial learning: an application to recidivism prediction, 2018. URL: https://arxiv.org/abs/1807.00199, arXiv:1807.00199.

[WWZ+23] Magdalena Wysocka, Olaf Wysocki, Michaël Zufferey, et al. A systematic review of biologically-informed deep learning models for cancer: fundamental trends for encoding and interpreting oncology data. *BMC Bioinformatics*, 24:198, 2023. doi:10.1186/s12859-023-05262-8.

[WY25] Xiaoyang Wang and Christopher C. Yang. Balancing fairness and performance in healthcare ai: A gradient reconciliation approach. In Riccardo Bellazzi, José Manuel Juarez Herrero, Lucia Sacchi, and Blaž Zupan, editors, *Artificial Intelligence in Medicine*, pages 480–489, Cham, 2025. Springer Nature Switzerland.

[WZX+21] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *Advances in Neural Information Processing Systems*, 34, 2021.

[Xpr] FICO Xpress Optimization. URL: https://www.fico.com/en/products/fico-xpress-optimization.

[XTJ18] Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson. Output reachable set estimation and verification for multilayer neural net-

works. *IEEE Transactions on Neural Networks and Learning Systems*, 29(11):5777–5783, 2018. doi:10.1109/TNNLS.2018.2808470.

[XTRJ18] Weiming Xiang, Hoang-Dung Tran, Joel A. Rosenfeld, and Taylor T. Johnson. Reachable set estimation and safety verification for piecewise linear systems with neural network controllers. In *2018 Annual American Control Conference (ACC)*, pages 1574–1579, 2018. doi:10.23919/ACC.2018.8431048.

[XZW+21] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and Complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2021. URL: https://openreview.net/forum?id=nVZtXBI6LNn.

[YF25] A. Yinusa and M. Faezipour. A multi-layered defense against adversarial attacks in brain tumor classification using ensemble adversarial training and feature squeezing. *Scientific Reports*, 15:16804, 2025. doi:10.1038/s41598-025-00890-x.

[YHPC18] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018. doi:10.1109/MCI.2018.2840738.

[YSX+22] Weijie Yu, Zhongxiang Sun, Jun Xu, Zhenhua Dong, Xu Chen, Hongteng Xu, and Ji-Rong Wen. Explainable legal case matching via inverse optimal transport-based rationale extraction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pages 657–668. Association for Computing Machinery, 2022. doi:10.1145/3477495.3531974.

[ZCX+20] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane S. Boning, and Cho-Jui Hsieh. 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, 2020.

[ZHD+23] Haobo Zhang, Junyuan Hong, Fan Dong, Steve Drew, Liangjie Xue, and Jiayu Zhou. A privacy-preserving hybrid federated learning framework for financial crime detection, 2023. URL: https://arxiv.org/abs/2302.03654, arXiv:2302.03654.

[ZHW+18] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics. *Phys. Rev. Lett.*, 120:143001, Apr 2018. URL: https://link.aps.org/doi/10.1103/PhysRevLett.120.143001, doi:10.1103/PhysRevLett.120.143001.

[ZKR+17] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017)*, pages 3391–3401, 2017. URL: https://papers.nips.cc/paper/6931-deep-sets.

[ZMN22] Lei V. Zhang, Afshin Marani, and Moncef L. Nehdi. Chemistry-informed machine learning prediction of compressive strength for alkali-activated materials. *Construction and Building Materials*, 316:126103, 2022. doi:10.1016/j.conbuildmat.2021.126103.

[ZQMQ20] Yi Zeng, Han Qiu, Gerard Memmi, and Meikang Qiu. A data augmentation-based defense method against adversarial attacks in neural networks. In Meikang Qiu, editor, *Algorithms and Architectures for Parallel Processing*, pages 274–289, Cham, 2020. Springer International Publishing.

[ZWC+18] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4944 – 4953. Curran Associates Inc., 2018.

[ZYW19] Weiting Zhang, Dong Yang, and Hongchao Wang. Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE Systems Journal*, 13(3):2213–2227, 2019. doi:10.1109/JSYST.2019.2905565.