Formally Explaining Neural Network Classification

Tomáš Kolárik[1], Grigory Fedyukovich[1,3], Faezeh Labbaf[1], Fabrizio Leopardi[1], Natasha Sharygina[1], Michael Wand[2]

[1] University of Lugano (USI), Lugano, Switzerland
[2] SUPSI, IDSIA, Lugano, Switzerland
[3] Florida State University, United States of America

Abstract. Neural networks (NNs) are the core of AI-based technologies. However, the degree of reliability in performing the task is an open problem. The explainability of a central task of NNs, classification, is of immense importance. While at the rise of AI-based reasoning, explainability of the NN classification has mostly been done using statistical methods, nowadays, a more reliable trend of formal logic-based methods is gaining popularity. The advantage of the formal approach is that it gives strict and provable guarantees of the classification. Formal methods is a mature field that has delivered a number of efficient computational solutions already applied in the analysis of software and hardware systems. Formal explainability methods naturally have the ability to reuse existing techniques and tools for a newly emerging field of formal explainability of NN classification.

This paper surveys existing efforts to compute explanations of neural network classification based on logical abductive reasoning. The abduction approach is crucial for generalizing the results, capturing the underlying behavior of the classifier. We present the existing techniques as instances of a general formalization that allows contrasting them against each other. In addition, we discuss the issue of the quality of explanations, focusing on their key metrics and factors. As an illustrative example, the paper also presents a practical framework, SpEXplAIn, which automatically computes Space Explanations, the most general abduction-based explanations for classifying NNs with provable guarantees of the behavior of the network in continuous areas of the input feature space. The tool leverages an SMT solver compatible with a range of flexible Craig interpolation algorithms and unsatisfiable core generation, and is applicable to a wide range of applications.

1 Introduction

Classifying neural networks (NNs) are of immense interest and importance in the research and industry. Nonetheless, despite the enormous complexity, these machine learning models resemble black boxes and the reasons behind the classification remain unclear. Explainable artificial intelligence (XAI) reasons through the classification process and provides human-understandable interpretations:

Given a NN and a sample point, that is, an assignment of all features of the NN to concrete values, the problem is to explain the classification outcome computed by the NN. Many approaches to XAI have been developed over the years:

- The analysis at the *unit (neuron) level* [23,76], which, however, fails to capture global properties or dependencies between units.
- *Gradient-based* methods [68,4], which locally describe the network behavior around a particular sample.
- *Inversion* methods [49], which provide global, but only approximate, views.
- *Model-agnostic* methods [5,48], including SHAP [43], LIME [61] and Anchors [62], which, however, often yield logically inconsistent explanations [51].

While useful and widely used these methods *lack rigor*. In this paper, we focus on alternative rigorous XAI approaches that are based on first-order logic. Relying on constraint solving and Satisfiability Modulo Theories [56,57] (SMT), they deliver explanations accompanied by *provable guarantees* regarding their correctness. These XAI tools leverage computational engines like Marabou [41] or SMT solvers [6,16,10,13,21,12]. The explanations are obtained through *abductive reasoning*, which originates from program verification [19,1,59,58]. The key insight is that *changing* the classification outcome for the input sample point changes the *satisfiability* of the logical formula that encodes the XAI-problem. Thus, if a relaxation of the input constraints in the formula does not affect the unsatisfiability of the formula, then this relaxation gives a *logically guaranteed generalization* of the explanation. Early methods in this domain perform rather naïve relaxation of input constraints based on exhaustive enumeration of features and without taking the proof of unsatisfiability into account [32,67,65,51,73,50]. They yield explanations where the behavior outside the space of the underlying dataset or samples generated from the model remains unknown. In contrast, interval-based explanations [35] are represented by hyperrectangular spaces, but they still cannot reflect feature relationships to precisely approximate decision boundaries. Thus, all existing approaches to logic-based XAI are more restrictive than what the original notion of abductive reasoning could in theory offer.

A breakthrough in logic-based XAI has been recently enabled by the concept of *Space Explanations* [44] that exploits the proofs of unsatisfiability of logical formulas as opposed to the structure of these formulas. It yields explanations where subsets of the feature space associated with the formulas are correct by construction and, in practice, much more general than the ones produced by the earlier methods. There is, however, no unifying framework to formulate the outcome expectations of the logic-based approaches, given the diversity of logic-reasoning tools that are used at the background. Consequently, even the notion of abduction is often misused. In our overview, we bridge the gaps in existing formalizations of logic-based methods to obtain explanations and bring them under the same umbrella. This paper uses a unified formalization for logical abduction-based explanations of NNs, allows a comprehensive comparison of the different approaches, provides a unified encoding for the corresponding explainability problems, gives an overview of existing algorithms solving the addressed

problems, and provides an extensive discussion on the quality of abduction-based explanations.

Moreover, the paper demonstrates the first tool to automatically generate space explanations called SpEXplAIn[4]. At the heart of SpEXplAIn, *Craig interpolation* [15] extracts a reason why the given input cannot change the classification from the logical proof. The tool is flexible and accepts optional configurations from the user, allowing them to tune various properties of the explanations, such as the logical strength or the size of the formula, and it can generalize existing space explanations. Notably, the tool can provide human-recognizable interpretations by plugging in different visualization methods. SpEXplAIn is applicable to a wide range of applications which the paper illustrates on several real-life case studies and showcases that the explanations can be visualized and provide meaningful feedback to the user.

2 Background

A classification task involves mapping input data into a predefined set of classes. In this context, we are given a set of *features* $\mathcal{F} = \{1, \ldots, m\}$ and a finite set of classes $\mathcal{K} = \{c_1, c_2, \ldots, c_K\}$. Each feature $i \in \mathcal{F}$ takes values from a discrete or continuous *domain* $\mathcal{D}_i$. The set of domains is represented by $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_m\}$. The *feature space* is defined as $\mathbb{F} = \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_m$.

The notation $\mathbf{x} = (x_1, \ldots, x_m)$ is used for an arbitrary point in feature space, where each x_i is a *feature variable* taking values from $\mathcal{D}_i$. Where clear from context, we will abbreviate feature variables as features. The notation $\mathbf{s} = (s_1, ..., s_m)$ represents a specific *sample point* in the feature space (i.e., $\mathbf{s} \in \mathbb{F}$), where each s_i is a constant representing one concrete value from $\mathcal{D}_i$.

A *classifier* $\mathcal{M} = (\mathcal{F}, \mathcal{D}, \mathbb{F}, \mathcal{K}, \kappa)$ is characterized by a *classification function* $\kappa : \mathbb{F} \to \mathcal{K}$ that maps the feature space $\mathbb{F}$ into the set of classes $\mathcal{K}$. An *instance* is denoted as a pair $(\mathbf{s}, c)$, where $\mathbf{s} \in \mathbb{F}$ and $c \in \mathcal{K}$ is a prediction $c = \kappa(\mathbf{s})$. The classification function κ is usually estimated from a set of sample points for which the correct class c is known (the *training data*).

(Classifying) Neural Networks are a specific class of general classifiers. A (feed-forward) neural network is represented by a graph $G = (E, V)$ with the set of edges E and the set of vertices V, representing the neurons and their weighted connections, respectively. The neurons are partitioned into layers $V_0, \ldots, V_L$ where each V_l contains neurons $v_i^{(l)}$, $i \in \{1, \ldots, n^{(l)}\}$. Layer V_0 is the *input* layer, V_L is the *output* layer, and layers $V_1, \ldots, V_{L-1}$ are *hidden* layers. The edges correspond to the pairs connecting every neuron of layer V_l to every neuron in the next layer V_{l+1}, for all $0 \leq l < L$, i.e., edge $e_{i,j}^{(l)}$ connects i-th neuron in layer l with j-th neuron in layer $l + 1$. Every edge is assigned a numerical weight $w_{i,j}^{(l)}$. Moreover, every neuron $v_i^{(l)}$ in a hidden layer is assigned a bias $b_i^{(l)}$. We assume that each input neuron $v_i^{(0)}$ corresponds to feature $i \in \mathcal{F}$, i.e., is represented

[4] https://github.com/usi-verification-and-security/spexplain

by feature variable $x_i \in \mathcal{D}_i$, and that each output neuron $v_i^{(L)}$ corresponds to class c_i. We assume that each domain $\mathcal{D}_i$ is either a finite set or a closed interval $\mathcal{D}_i = [L_i, U_i]$ where both the lower and upper bounds are finite. This is always true as long as the (training) data set is finite. The restriction to finite intervals is required since a neural network is always trained on a finite set of training data, thus it will learn to expect inputs in a finite range; passing an input (far) outside this range will inevitably cause unpredictable behavior.

Given an input sample $\mathbf{s} = (s_1, \ldots, s_m)$, the activations $x_i^{(l)}$ for each neuron $v_i^{(l)}$ are computed as follows. In the case of the input layer, $x_i^{(0)} = s_i$; for hidden layers, $y_i^{(l)} = \sum_j x_j^{(l-1)} w_{j,i}^{(l-1)} + b_i^{(l)}$ and $x_i^{(l)} = ReLU(y_i^{(l)})$, where $ReLU(x) = \max\{x, 0\}$ is the rectifier function. In the case of the output layer, no activation function is applied[5] . The class c_i assigned to input sample $\mathbf{s}$ is given by the output neuron with maximum activation, formally, $\kappa(\mathbf{s}) = c_i \Leftrightarrow i = \arg\max_j \{y_j^{(L)}\}$.

Example 1. A simple neural network maps features $\mathbf{x} = (x_1, x_2, x_3) \in [0, 4]^3$ to classes $\{c_1, c_2\}$. It has 3 input neurons, 2 hidden neurons with *ReLU* activation, and 2 output neurons. Using fixed weights and zero biases, they are defined by

$$x_1^{(1)} = ReLU(2x_1 + x_3) \wedge o_{c_1} = y_1^{(2)} = x_1^{(1)} - 4x_2^{(1)}$$
$$x_2^{(1)} = ReLU(-x_1 + x_2 - x_3) \wedge o_{c_2} = y_2^{(2)} = -x_1^{(1)} + 4x_2^{(1)}.$$

The network predicts c_1 if $o_{c_1} \geq o_{c_2}$, otherwise c_2. For instance, with sample $\mathbf{s} = (1, 1, 3)$, we get $o_{c_1} = 5$, $o_{c_2} = -5$, so $\mathbf{s}$ is classified as c_1.

3 Abduction-based Explanations

The explanations of NNs studied in this paper are obtained through abductive reasoning which, given an observation O, infers an explanation φ that entails O. Formally, for vectors of variables $\mathbf{x}$ and $\mathbf{y}$, and two formulas $T(\mathbf{x}, \mathbf{y})$ and $O(\mathbf{x}, \mathbf{y})$, the *abduction problem* is to find a formula $\varphi(\mathbf{x})$, such that: (1) $\varphi(\mathbf{x}) \wedge T(\mathbf{x}, \mathbf{y}) \not\Rightarrow \bot$, and (2) $\varphi(\mathbf{x}) \wedge T(\mathbf{x}, \mathbf{y}) \Rightarrow O(\mathbf{x}, \mathbf{y})$. For example, if $T \equiv x > 0$ and $O \equiv x+y > 0$, then one possible solution for the abduction problem is $\varphi_1 \equiv y \geq 0$ (the weakest), and another one is $\varphi_2 \equiv y = 10$ (a stronger). In our context, φ is an explanation, vectors $\mathbf{x}$ and $\mathbf{y}$ are the input and output variables of the network, respectively. T encodes the network and the constraints over the input variables, and O contains the conditions over the output, e.g., the classification. *Weaker* formulas give more information about the classifier. However, since the explanations serve for humans to better understand the behavior of the NN, it is essential that the explanations are also meaningful and interpretable—refer to Section 6 for more details. Explanation methods are distinguished mainly in how they solve the abduction problem and the type of solutions they provide.

Abduction-based explanations have been studied by several works in different forms [35,44,32,34][6]. To unify all different forms of such explanations, this section

[5] During the training, usually the softmax function is used in the last layer; since this is a monotonic function, it can be ignored here.

[6] Refer to Section 5 for other formalisms beyond abductive reasoning.

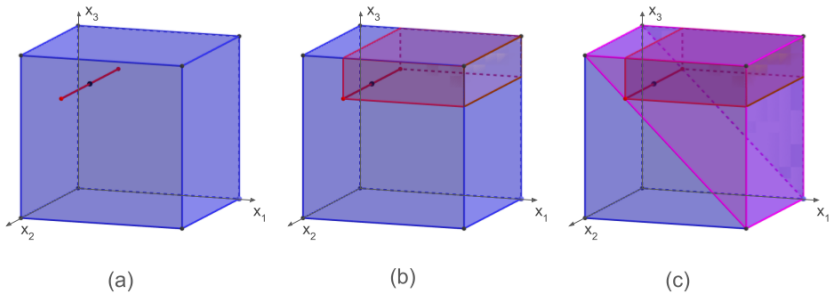

Fig. 1. Impact spaces associated to different explanations in domain $[0, 4]^3$

starts with a general formalization of explanations introduced in [44] and called *space explanations*, which directly stems from the abduction problem. Then, the section discusses special cases, some of which are widely used in the literature.

Definition 1 (Space Explanation). *Given a classifier $\mathcal{M}$ computing a classification function κ from feature space $\mathbb{F}$, and a class c, a* space explanation *of c is a formula φ s.t. $\forall\, \mathbf{x} \in \mathbb{F} \,.\, \varphi(\mathbf{x}) \implies \kappa(\mathbf{x}) = c$ holds.*

This formalism can be viewed as a set of *rules*, where a space explanation represents a sufficient condition for the classification: if the condition holds for the given features, then it must be classified to class c. The set representation of a space explanation φ is its *impact space* $\mathbb{F}_\varphi := \{\mathbf{x} \in \mathbb{F} \mid \varphi(\mathbf{x})\}$, and $\mathbb{F}_\varphi \subseteq \{\mathbf{x} \in \mathbb{F} \mid \kappa(\mathbf{x}) = c\}$.

Given the network in Example 1, a space explanation for class c_1 is e.g. $\varphi \equiv x_1 + x_3 \geq 4$. Figure 1 (c) shows the impact space $\mathbb{F}_\varphi$ as a pink prism.

The above formalization is general enough to cover any other abduction-based explanation definition by possibly restricting formula φ. In the following, we introduce them as special cases of space explanations.

Explanations as a Subset of Features The concept of abductive explanations was first introduced by [32] as a subset of conditions over the input feature variables that guarantee the output conditions, i.e., the classification. This abductive explanation, that is restricted compared to the original notion of abduction, can be formalized as a special case of space explanation as follows:

Definition 2. *An* abductive explanation *$\varphi_\mathcal{X}$ is a space explanation for a classifier $\mathcal{M}$ and a class c such that $\varphi_\mathcal{X} := \bigwedge_{i \in \mathcal{X}} x_i = s_i$ where $\mathcal{X} \subseteq \mathcal{F}$ is a subset of features. $\varphi_\mathcal{X}$ is* subset-minimal *iff $\forall\, \mathcal{X}' \subsetneq \mathcal{X} \,.\, \varphi_{\mathcal{X}'}$ does not constitute a space explanation anymore.*

That is, any input sample that coincides with $\mathbf{s}$ on features from $\mathcal{X}$ but can have arbitrary values for features from $\mathcal{F} \setminus \mathcal{X}$ is also classified to class c. In the case $\varphi_\mathcal{X}$ is subset-minimal, subset $\mathcal{X}$ is required to be irreducible.

For example, given the NN in Example 1 and instance $((1, 1, 3), c_1)$, a subset-minimal abductive explanation is $\varphi_\mathcal{X}$ with $\mathcal{X} = \{x_1, x_3\}$. Thus, if $x_1 = 1$ and

$x_3 = 3$, then for any value assigned to x_2 from the domain, the classification is still c_1. However, if any feature from $\mathcal{X}$ is removed, then neither $x_1 = 1$ nor $x_3 = 3$ alone guarantees the classification anymore. The corresponding impact space is a line shown in Figure 1 (a).

Explanations as a Set of Intervals Another special case of space explanations specifies an interval for selected input features (which is a subset of the corresponding domain interval), offering higher granularity over Definition 2.

Definition 3 (Interval Explanation). *An* interval explanation φ_I *is a space explanation for a classifier* $\mathcal{M}$ *and a class* c *s.t.* $\varphi_I := \bigwedge_{i \in \mathcal{X}} l_i \leq x_i \leq u_i$ *where* $\mathcal{X} \subseteq \mathcal{F}$ *and* $I = \{[l_i, u_i] \mid i \in \mathcal{X}, l_i, u_i \in \mathcal{D}_i\}$.

Thus, $[l_i, u_i] \subseteq \mathcal{D}_i$ for all $i \in \mathcal{X}$. The definition does not allow φ_I to represent multiple discontinuous intervals (which could together still be a subset of $\mathcal{D}_i$). The associated impact space is a hyperrectangle expressed as the Cartesian product of intervals over each feature i: $\mathbb{F}_{\varphi_I} = \prod_{i \in \mathcal{F}} [l_i, u_i]$, $[l_i, u_i] = \mathcal{D}_i$, $i \in \mathcal{F} \setminus \mathcal{X}$.

For example, given the NN in Example 1, an interval explanation for class c_1 with $I = \{[1, 4], [3, 4]\}$ and $\mathcal{X} = \{1, 3\}$ is $\varphi \equiv (1 \leq x_1 \leq 4) \wedge (3 \leq x_3 \leq 4)$. The impact space of the explanation is shown in Figure 1 (b).

A notion of maximal interval explanations depends on multicriterial optimization, resulting in ambiguity of what should be maximized—considering both the size of $\mathcal{X}$ and of particular intervals. Also, given a subset $\mathcal{X}$, many combinations of differently sized intervals would exist because it would highly depend on the order and the extent of the particular expansions. A possibility would be to require the resulting volume to be maximal.

There is a similar notion of interval explanations called *inflated explanations* [35]. However, the definition differs in that it uses a general concept of *maximal sets*, meaning that it also admits discrete domains or subsets of continuous space (which are not necessarily continuous, i.e., connected)[7], and that they cannot be expanded further. Moreover, they explicitly restrict the explanations to origin from a subset-minimal abductive explanation. Still, in all cases, the resulting explanations fall into space explanations.

Distance-Restricted Explanations Literature often considers explanations that focus only on a small vicinity around the sample point rather than the entire feature space [73,34,31]. They are defined as follows.

Definition 4. *A* distance-restricted abductive explanation $\varphi_{\mathcal{X},\epsilon}$ *is a space explanation for a classifier* $\mathcal{M}$ *and an instance* $(\mathbf{s}, c)$ *s.t.* $\varphi_{\mathcal{X},\epsilon} := \|\mathbf{x} - \mathbf{s}\|_p \leq \epsilon \wedge \bigwedge_{i \in \mathcal{X}} x_i = s_i$ *where* ϵ *is a distance value and* $\|.\|_p$ *is a norm.*

Distance-restricted abductive explanations do not fall into abductive explanations: effectively, such an explanation forms a space in the feature space and no

[7] Despite the general definition, [35] incorrectly considers only intervals in the continuous domain.

longer depends solely on the features. Nonetheless, if the uniform norm is used ($p = \infty$), they can be expressed as interval explanations, which can naturally restrict the neighborhood around the sample point, resulting in a hyperrectangular space. However, this is not possible with other norms, such as the Euclidean norm ($p = 2$) where the resulting space corresponds to a hypersphere. In all cases, distance-restricted abductive explanations still fall into space explanations.

Summary Abductive, interval, and inflated explanations are special cases of space explanations, where the formula φ is restricted. This constrains the shape of the corresponding impact space. While a general space explanation can have an arbitrarily shaped impact space, the impact space of both abductive and interval explanations takes the form of an affine rectangle. Abductive explanations also always keep some of the features fixed to a constant. Figure 1 compares impact spaces for the three types of explanations discussed above.

4 Technical Foundations

The computation of abduction-based explanations for NNs follows a structured two-phase methodology. First, the target ML model is encoded in first-order logic, e.g., real arithmetic (LRA) in Satisfiability Modulo Theories (SMT). Second, abductive reasoning is applied to identify a formula that entails the model's prediction for a given data instance. Coming up with such a formula involves one or more entailment queries where tools like SMT solvers or specialized verifiers act as oracles to verify the entailment. Different methods vary in how they approach the abduction problem and in the types of tools used to compute explanations. We start with a possible encoding for various approaches that yield different types of abduction-based explanations. Then, we survey the approach that leverages Craig interpolation to efficiently produce highly general explanations, followed by approaches that generate more structured and restricted forms of explanations using NN verification tools.

4.1 Encoding

While all theoretical foundations hold in both cases, we instantiate all the techniques in the continuous domain and omit the discrete domains. All approaches to NN explainability use the following encoding schema. Section 2 gives the definition of all layers of a NN, where all neurons are variables and the weights and biases are constants. The encoding of this NN is $\psi_{\mathcal{M}}$. The encoding of $ReLU(x)$ depends on the concrete approach: in the most precise scenario, it is an ITE-term; some approaches, typically NN verifiers (e.g. [70]), apply abstractions [69], but at the cost of additional verification calls to check that the correctness has not been affected. The domains $\mathcal{D}$ of the features are encoded as $\psi_{\mathcal{D}}$, which is a conjunction of intervals for possible values of every feature.

Let the formula $\neg\psi_c \equiv \bigvee_{i \in \mathcal{K}} y_i^{(K)} > y_{j_c}^{(K)}$ represent a constraint that the outcome of the classification is *not* class $c \in \mathcal{K}$, then with no additional restrictions,

formula $\psi \equiv \psi_{\mathcal{M}} \wedge \psi_{\mathcal{D}} \wedge \neg\psi_c$ is *satisfiable*. Now suppose a sample point $\mathbf{s} \in \mathbb{F}$ classified as class c. Using the encoding $\varphi_{\mathbf{s}} \equiv \mathbf{x} = \mathbf{s}$, formula $\varphi_{\mathbf{s}} \wedge \psi$ is in turn *unsatisfiable*. For explainability, this fact lets to weaken formula $\varphi_{\mathbf{s}}$ while still preserving the unsatisfiability.

4.2 Craig Interpolation for Explanations

Given an unsatisfiable formula $A \wedge B$, a *Craig interpolant* is a formula I such that $A \Rightarrow I$, $I \wedge B$ is unsatisfiable, and I uses only the common variables of A and B. We denote interpolant I computed by an interpolation procedure *Itp* from formulas A and B by $I = \mathit{Itp}(A, B)$. Hence, given encoding ψ and a space explanation φ of c, interpolant $I = \mathit{Itp}(\varphi, \psi)$ is also a space explanation of c satisfying $\varphi \implies I$ (and hence $\mathbb{F}_{\varphi} \subseteq \mathbb{F}_I$). This offers a universal means of generalization of existing space explanations (e.g. trivially a sample $\varphi_{\mathbf{s}}$), still guaranteeing the classification.

Several interpolation algorithms, e.g. [2,8] for quantifier-free LRA, typically yield explanations in the form of bounds on linear combinations of features such as $\bigwedge_j \sum_{i \in \mathcal{F}} \alpha_{j,i} x_i \leq \beta_j$ ($\alpha_{j,i}, \beta_j \in \mathbb{Q}$). However, these algorithms are "blind" in the context of neural network classification and they do not allow more granular control of the form of the interpolants, such as specific shapes. While post-processing and simplifying the formula is often beneficial, several methods (cf. Section 6) exist to interpret even complex formulas in a user-friendly manner.

Furthermore, interpolation algorithms can benefit from special cases [26] where given $\mathit{Itp}(A, B)$, formula A can be partitioned, leaving a part of A unchanged. For example, if a space explanation can be split into two parts $\varphi_A \wedge \varphi_B$, and given a Craig interpolation procedure *Itp* with $I = \mathit{Itp}(\varphi_A, \varphi_B \wedge \psi)$, then $I \wedge \varphi_B$ is a logically weaker space explanation. Hence, it is possible to, for example, focus the interpolation only on selected features that appear only in φ_A, excluding the rest appearing only in φ_B from the expansion.

Different Craig interpolation procedures offer various properties of the interpolants, in particular, their *strength* [20,63]. Moreover, the *dual* approach Itp' to an existing procedure *Itp* s.t. $\mathit{Itp}'(A, B) = \neg\mathit{Itp}(B, A)$ can be used to produce (usually) weaker explanations. The algorithms usually range over either the propositional part or a specific theory. An example of interpolation rules over the Boolean abstraction of the formula is *McMillan* algorithm [53] (Itp_M) that typically yields conjunctions of constraints, while its weaker dual interpolation scheme (Itp'_M) typically yields disjunctions ($\mathit{Itp}_M(A, B) \implies \mathit{Itp}'_M(A, B)$).

In the case of quantifier-free LRA, given a system of unsatisfiable linear inequalities, a widely used technique is based on Farkas' lemma, producing *Farkas interpolants* (Itp_F), or their logically stronger *decomposed* variants [8] (Itp_{DF}), where Itp_F yields a single inequality, while Itp_{DF} a conjunction of inequalities. The duals of these algorithms Itp'_F and Itp'_{DF} yield weaker interpolants and Itp'_{DF} yields a disjunction of inequalities. Since these algorithms are orthogonal to McMillan algorithms, we can even produce conjunctions of disjunctions or vice versa. Furthermore, there is an infinite family of interpolants [2] (Itp_f) with

the logical strength in between Itp_F and Itp'_F interpolants, parametrized by a rational factor $f \in [0, 1]$.

Hence, a number of flexible interpolation algorithms are available, offering different strengths and structures of the resulting formula. The presented arithmetic algorithms exhibit the following relationship [8,2]:

$$Itp_{DF} \implies Itp_F \implies Itp_f \implies Itp'_F \implies Itp'_{DF}$$

such that each is applied to the same arguments (A, B).

Unsatisfiable Cores Given formulas A and B such that $A = \bigwedge_{i\in\mathcal{I}} a_i$ and $A \wedge B$ is unsatisfiable, an *unsatisfiable core* of A is a formula $U = \bigwedge_{i\in\mathcal{J}} a_i$, $\mathcal{J} \subseteq \mathcal{I}$ such that $U \wedge B$ is still unsatisfiable. We denote unsatisfiable core U computed by an algorithm *Ucore* from formulas A and B by $U = Ucore(A, B)$. If $Ucore(A, B)$ produces a formula over the common variables to A and B, it is a special case of Craig interpolation.

Most modern SMT solvers [6,13,30,16] implement computation of unsatisfiable cores. However, different techniques offer different trade-offs between the computational effort and size of the core. Some techniques [14] add very little overhead on top of proving unsatisfiability but offer no guarantees about the size of the core. Other techniques compute *irreducible* unsatisfiable cores [27,39], but are typically much more computationally intensive.

Generalizing Existing Explanations The concept of Craig interpolation allows the *iterative construction* of space explanations using a sample point or existing abductive, interval, or inflated explanation as a starting point. [44] lists several strategies and suggests some combinations in order to control the logical strength of the resulting explanations, as well as their form, including possible simplifications using the unsatisfiable core extraction.

4.3 Verification for Explanations

Given a specification of a neural network f of the form $\{precondition\}\ \mathbf{r} = f(\mathbf{x})\ \{postcondition\}$, where *precondition* is a predicate over variables $\mathbf{x}$, and *postcondition* is a predicate over variables $\mathbf{x}$ and $\mathbf{r}$, the verification problem consists of checking if for all the values of variables for which *precondition* holds, *postcondition* also holds. Many state-of-the-art NN explanation methods use verification tools to compute conditions over input features that guarantee the output [73,34,32,7]. The explanation approach is reduced to a sequence of verification problems with different preconditions until a suitable (and in practice, weaker) one is found. In a nutshell, these techniques perform a naive relaxation of input constraints based on exhaustive enumeration of features, heavily depending on selecting a specific order of the features. Each verification problem is defined by *precondition* as a conjunction of an input φ (starting with the sample point $\varphi_{\mathbf{s}}$) and the domain constraints $\psi_{\mathcal{D}}$, and the *postcondition* as the constraint that the classification is not class c, $\neg\psi_c$ (recall Sect. 4.1).

NN verification and abstraction literature is thoroughly reviewed in [40]. It classifies the existing approaches into either "Reduction" class [3], "Branch-and-Bound" class [74], or classes that apply abstractions to the input domains [46,69] or the NN architecture [22]. Reduction approaches solve the verification problem by reducing it to an instance of another problem, such as SMT (most closely related to abduction, to be discussed in the next paragraph), MILP, Convex Optimization, and BDD. Branch-and-bound techniques work by splitting the input or activation space into independent branches. Since each branch is verified separately, this often increases the likelihood of finding a proof or counterexample. One part of abstraction-based approaches use "abstract interpretation" (e.g. [69]) to over-approximate the input domains (e.g., to intervals, zonotopes, or polytopes) and propagate the properties through the layers of the NN. Other approaches instead perform NN's structural abstraction (e.g. [22]) by reducing the size of the NN while maintaining guarantees of the verification, either by exploiting syntactic or semantic information regarding the network's topology.

Among existing tools for NN verification, $\alpha, \beta-$CROWN [74,70], VeriNet [28], Marabou [41,71], MN-BaB [24] are most used. A survey paper [42] gives details about other tools. Many of these approaches use an SMT solver under the hood. For instance, Marabou uses the CVC5 [6] SMT solver to check that every assignment of variables for which the *precondition* holds can only be extended to an assignment for which *postconditon* holds. In order to solve the verification problem, it loops by trying to satisfy the currently unsatisfied constraints at each iteration. The main verification loop exploits the well-known simplex algorithm.

The explanation algorithms discussed in this section leverage a *robustness oracle* to answer the robustness query: whether a constrained adversarial example exists when certain features are fixed to their original values [73,34,32,7]. Formally speaking, given a classifier $\mathcal{M}$ computing a classification function κ from feature space $\mathbb{F}$, and constraint $\varphi(\mathbf{x})$, a *constrained adversarial example* for instance $(\mathbf{s}, c)$ exists if $\exists\, \mathbf{x} \in \mathbb{F} \,.\, \kappa(\mathbf{x}) \neq c \wedge \varphi(\mathbf{x})$. If an adversarial example does not exist, then the classifier is *constrained robust*. Robustness is a special case of the verification problem.

Finding a constrained adversarial example in the vicinity of a sample point is equivalent to solving the negation of the problem in Definition 1, searching for a counterexample $\mathbf{x}$ with $\varphi = \varphi_{\mathcal{X},\epsilon}$ from Definition 4.

Compared to Craig-interpolation-based methods, the methods discussed in this subsection *restrict* the *shape* of the impact space associated with the explanations, disallowing the approximation of arbitrarily complex decision boundaries. Consequently, the former methods can *capture relationships* among features, offering meaningful insights into the *reasons* behind a classification. On the other hand, interpolation methods are in a sense "blind" (i.e., unaware of their application context), and their output still benefits from further post-processing.

Computing explanation as a subset of features Methods that aim at subsets of features typically use brute force or binary search, with optimizations.

Brute-Force Methods A straightforward approach to solve the problem is by assuming that all features are initially fixed to their original values. That is, if $\mathcal{X}$ initially contains all features $\mathcal{F}$, then no adversarial example exists. A brute-force algorithm then loops through each feature i in the set of all features $\mathcal{F}$. In each iteration, it tentatively removes the current feature i from the set of fixed features $\mathcal{X}$. It then makes a call to a robustness oracle (like Marabou) to check if a constrained adversarial example exists when only the features remaining in the set $\mathcal{X}$ are fixed. If the oracle returns true (an adversarial example exists), it means feature i was necessary to maintain the original prediction . The algorithm then adds feature i back to the set $\mathcal{X}$. If the oracle returns false (no adversarial example exists when i is unfixed), it means feature i was not necessary to prevent the adversarial example, and it can be safely left unfixed. After iterating through all features, the remaining set $\mathcal{X}$ is a subset-minimal abductive explanation. The algorithm needs $\Theta(|\mathcal{F}|)$ calls to the robustness oracle. To potentially find smaller subset-minimal explanations, heuristic models of feature importance can be employed, checking less important features first. Variants of this method are studied in recent works, for example, VeriX [73] or [29].

Binary Search Methods from another group do not check each feature sequentially, but instead use binary search on the set of features being considered ($\mathcal{W}$). It maintains a set $\mathcal{X}$ of features already identified as part of the abductive explanation and a set $\mathcal{W}$ of features still under consideration. In an inner loop, it performs a binary search over the features in $\mathcal{W}$ (which are assumed to be ordered). In each step of the binary search, it checks a *chunk* of features. Specifically, it calls the oracle to see if an adversarial example exists when the features in $\mathcal{X}$ plus a subset of features from the beginning of $\mathcal{W}$ (e.g., $\mathcal{W}_{1..t}$) are fixed. Based on whether the oracle returns true (adversarial example exists) or false (no adversarial example), the binary search narrows down the range within $\mathcal{W}$ that contains the relevant feature. Once the inner binary search loop finishes, a relevant feature is found and moved from $\mathcal{W}$ to $\mathcal{X}$. This process repeats in an outer loop until all features initially in $\mathcal{W}$ have been examined. This method is studied in [7,34,72].

This binary search structure reduces the number of oracle calls within the inner loop to $O(\log |\mathcal{W}|)$, where $|\mathcal{W}|$ is the number of features currently being searched. If the largest abductive explanation has size kM, the outer loop runs $O(k_M)$ times. Therefore, the total number of sequential oracle calls is reduced to $O(k_M \log m)$, where m is the total number of features $|\mathcal{F}|$. This can offer better performance than the greedy algorithm if the largest abductive explanation is considerably smaller than the total number of features.

Viewing the algorithm as analyzing *chunks* of features highlights a possibility for parallelization, where checks on different chunks could be run simultaneously. This idea is further developed in the *SwiftXplain* algorithm [34].

Optimizations The robustness queries used in the mentioned algorithms can be expensive due to the complexity of neural networks, especially when many

features are set to free. Various techniques have been proposed to optimize the explanation process.

Ordering features heuristically [7,72,73] The order of processing features influences both explanation size and time. Prioritizing features that are more likely to appear in the final explanation generally improves efficiency and makes the explanations smaller. Analyzing the network to establish a heuristic ordering of features can significantly enhance the explanation process. Scoring methods such as LIME [61] and SHAP [48], and a feature-level sensitivity method [73] can be employed as a preprocessing step.

Using contrastive examples [7] A *contrastive example* (or *contrastive explanation*) is a subset of features $\mathcal{C} \subset \mathcal{F}$ s.t. altering them is sufficient to cause a misclassification of a given instance $(\mathbf{s}, c)$: $\exists\, \mathbf{x} \in \mathbb{F} \,.\, \kappa(\mathbf{x}) \neq c \wedge \bigwedge_{i \in \mathcal{F} \setminus \mathcal{C}} x_i = s_i$. The technique identifies contrastive singletons—features that, if changed individually, cause a misclassification—and immediately marks them as part of the explanation (i.e., fixes them), thus avoiding unnecessary robustness queries. Contrastive examples of size two are also exploited: if one feature in such a pair is removed from the explanation (i.e., freed), the other should be included.

Local-Singleton Search [7] Freeing features enlarges the search space for the underlying verifier and thus slows down the process. The core observation is that input points near the original input that cause misclassification tend to be clustered. To take advantage of this, instead of letting the values of previously freed features be arbitrary during verification queries, the method fixes them to the values from a previously found counterexample. Then, it searches for local singletons: features whose individual perturbation leads to misclassification, assuming all other features stay fixed at the counterexample values.

Computing Interval and Inflated Explanations Inflated explanations can be computed using a greedy algorithm that starts from a subset-minimal abductive explanation. In the continuous domain, it systematically expands the associated interval for each feature, strictly following a selected order while maintaining the same classification outcome. The resulting explanations represent maximal intervals. However, the intervals associated with the features appearing at the beginning of the ordering are usually larger than the latter, and the volume of the resulting impact space is typically not maximal. This is due to the greedy expansion of the former intervals in the ordering which is too restrictive with respect to the remaining features, leaving lower flexibility for their expansion.

Moreover, a modified version of Algorithm 3 from [35] can be used for computing interval explanations (Definition 3) if the stopping condition is relaxed to e.g. a predefined number of expansion steps. Also, since we assume that all domains are bounded (Section 2), the linear search in the algorithm should be replaced by binary, as suggested by the authors. The resulting explanations can be easier to compute and may yield higher volumes as well, although without guarantees of the maximality of the intervals or the volume. Such an algorithm can also be generalized so that it accepts any interval explanation as input.

5 Related Work

Explanation methods for neural networks range from gradient-based and attribution techniques to formal logic-based approaches. This section expands on methods that are not based on logic or abduction.

Approaches Not Based on Logic Classical approaches to explain classifying NNs mostly fall into one of the following categories: *Analysis at the unit (neuron) level* [23,76], which however fails to capture global properties or interdependencies between units, *gradient-based methods* [68,4], which describe the network behavior around a particular sample; and inversion methods [49], which are computationally efficient and can provide global views, but are still only approximating. There is also a range of *Model-agnostic* methods [5,61,48,62], which can be applied not only to NNs, but in principle to any classifier. Bottleneck-based approaches [75,64] address explainability through architectural constraints. Bottleneck architectures constrain information flow through compressed intermediate layers: either by forcing the model to focus on the most prediction-relevant input information, or by using human-defined concept layers. These models improve explainability by making the model's decision more transparent in the intermediate representations.

Other Logic-based Explanations Beyond abduction-based explanations, which aim to identify sufficient reasons for a classifier's decision, *contrastive* or *counterfactual* explanations [54] have also gained attention. It explains why another decision was not made—also known as a necessary reason. Several works have explored counterfactual reasoning in the context of AI systems [55,18,60,36]. There are other logic-based approaches that are not abduction-based, for instance, [25] introduces an approach that maps the neural activations to semantic concepts, which are then used to induce logic-based theories. Furthermore, in the context of image classification, [11] introduced *compositional* explanations as the set $\mathcal{E}$ of superpixels (i.e., clusters of adjacent pixels considered as a unique feature) that are the most *relevant* for the classification task. Last, but not least, recent work has addressed the practical computation of probabilistic abductive explanations for various families of classifiers [33,37].

Other surveys A few surveys have been done on formal AI explanation and verification. [45] surveys early works on utilizing formal methods for the explanation, verification, and testing of machine learning systems. [52] provides a technical survey of logic-based XAI, discussing its origins, the current topics of research, and emerging future research areas. [43] surveys NN verification tools from a machine learning perspective, while [40] reviews and classifies the literature on NN verification, which we used in Section 4.3.

6 Quality of Space Explanations

Since based on logic, space explanations enable rigorous measurement and comparisons of the quality of explanations, which typically depends on multiple properties of the underlying formula and is application-specific. Such quality metrics

are based on comparing formulas that represent explanations, which often have different shapes. For example, interval explanations are easy to interpret, but can sometimes be too restricted given the complexity of the feature space. We identify two main aspects of the quality that are often contradictory: (human) *interpretability*, and *generality* of the explanations. More general explanations may convey richer information about the network [44] but often come at the cost of reduced readability. The challenges of practical human interpretation of complex explanations, particularly in high-dimensional feature spaces, and their comparison, were addressed in [47] by applying classical methods from geometry, demonstrating the usability of the explanations for domain experts without the need for expertise in logic.

6.1 Interpretability

Explanations can be communicated in various ways, mainly by reading the formula, or using a visualization.

Formula Readability Human readability of the formula is, for example, important in the domains of natural sciences where an expert in the domain verifies whether the obtained rules are compatible with their knowledge. Explanations that constrain individual features (Definitions 2–4) result in short and structurally simple formulas that are easy to interpret. In contrast, non-restricted space explanations can be harder to grasp. The readability depends on the size of the formula, which can be defined as the number of terms, number of features involved, number of characters, etc.

Visualization Visualization can provide intuitive insights for the classification even for complex formulas. However, it may be difficult to quantify or compare them with other visualizations. A direct way is to visualize the impact space which is usually high-dimensional. There are several ways to overcome this, such as using *slices* or *projections* for selected pairs or triplets of features, resulting in a 2D or 3D space. Slices fix all other dimensions, providing a precise view into the selected dimensions, but conclusions on the excluded dimensions are very limited. Projections can be computed by quantifier elimination, which is expensive but can also be approximated using linear programming. They offer an intuitive understanding of the geometrical differences between impact spaces. They still do not capture the complete information contained in higher-dimensional spaces: they may result in too large area because the information from the other dimensions is collapsed. An example of such visual comparisons is shown in Figure 2: using

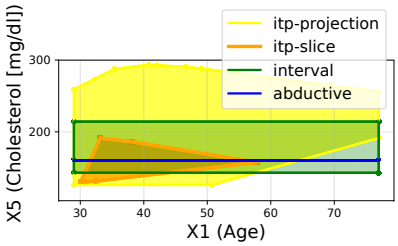

Fig. 2. Projection and slice visualization of three types of explanations

two visualization methods, projection and slice, it shows three types of explanations from [44] for a heart-attack risk model, focusing on two features of the patients (age and cholesterol). For interval and abductive explanations, both visualization methods result in the same area because the impact space of these explanations is always a hyperrectangle.

There are other means of dimensionality reduction as well [9], such as PCA, that exploits the data distribution within a dataset of sample points. A special case of visualization applies to the domain of image recognition, where particular features represent pixels. There, the explanations can be visualized using some projections on particular pixels (i.e., features), and the user can then decide which ones identified as responsible for the classification make more sense. An example of such visual comparisons appeared in [73] for abductive explanations.

Following [47], further means of visual interpretation include *extraction of the maximum diameter*, identifying a significant linear relation between features, and *extraction of the* (Euclidean) *minimum norm point*, which illustrates the flexibility of particular pixels in explanations for image-classification datasets: the darker pixels compared to the original image, the higher flexibility given by the explanation, reflecting the relevance of the pixels for classification, and also the size of the impact space (see an example in Figure 7). Both methods compute objects that are easy to visualize but at the same time represent correct explanations that are subsets of the original explanation.

6.2 Generality

The generality is characterized by the size of the impact space, reflecting the ability to cover multiple sample points. The size of the continuous space is estimated based on selected factors such as the volume. Moreover, some applications may favour the importance of certain input features over others. The choice of the used metric is application- or even explanation-specific: for example, volume estimation is robust and comprehensive, but only if the accuracy is meaningful and given that the computation for the explanation is feasible.

Subset Checks A direct comparison of explanations is to check whether one is a subset of another (e.g., using implication checks). While providing rigorous comparisons, explanations often intersect but are not entirely subsumed by one another. As a result, many explanations remain incomparable [44].

Number of Fixed Features A rough aspect of the generality is based on the number of features that are fixed to a single value by the formula, not allowing any flexibility. For example, a sample point fixes all features, resulting in the least general explanation possible, and abductive explanations (Definition 2) only constrain features to be fixed or entirely free within the domain.

Volume Computing a relative volume, that is, a volume after normalizing particular domains, estimates the size of an impact space. Particular domains can also be adjusted according to the importance of some features. While the volume

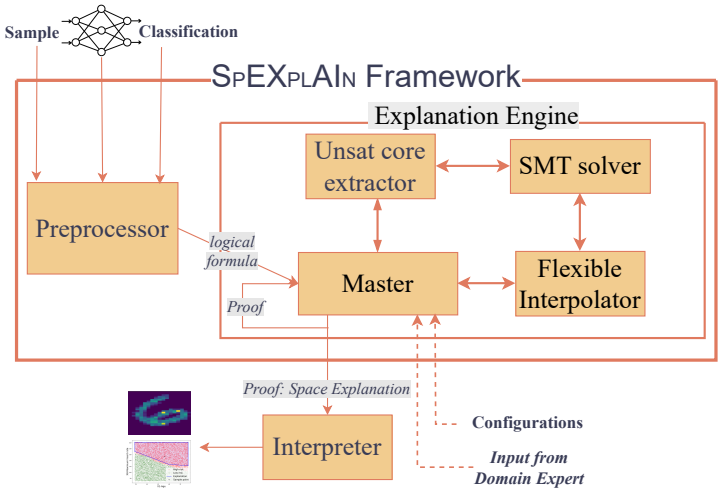

Fig. 3. Space explanations computation scheme

is easily computable for hyperrectangles, it is non-trivial for more general high-dimensional spaces, even if they are convex. Approximations can be obtained based on symbolic reasoning [66] or Monte Carlo methods [47], but may still be expensive. The volume is zero if any feature is fixed in the explanation, but one can compare the number of fixed features and compute the volume only within the rest. Since computing the volume in 2D or 3D spaces is feasible, another possibility is to exhaustively compare volumes of all slices or projections (cf. visualization) for all or selected pairs or triples of features.

7 Illustration of a Practical Framework SpEXplAIn

The SpEXplAIn framework builds on the most general abduction-based concept of *Space Explanations*. The tool is available online[8][9] with detailed instructions on how to build and run the tool. Based on SMT, it automatically generates explanations of NN classification in the form of logical formulas with provable guarantees about the classification and covering complex-shaped subsets of the feature space, enabling close decision boundary approximation. Starting from a sample point or an existing space explanation, the tool delivers explanations from logical proofs about the classification constructed on the fly. *Craig interpolation* utilizes the proof for expanding the space around the sample point and capturing the reasons behind the classification. Importantly, it preserves the guarantees about the classification, and can iterate for generalizing the explanation further. Figure 3 gives the framework overview with three main parts:

[8] https://github.com/usi-verification-and-security/spexplain
[9] https://verify.inf.usi.ch/spexplain

- **Preprocessor** encodes the NN, the sample point, and its classification into a logical formula in SMT, as decribed in Section 4.1.
- **Explanation Engine** receives the logical encoding and generalizes the constraints on the input features while preserving guarantees on the classification. Following Section 4.2, it outputs a proof that represents a space explanation. Internally, the Master subcomponent instructs the Interpolator and Unsat-core extractor subcomponents for iterative explanation generalization. The engine is optionally configurable to tailor the explanation to the specific needs of the user.
- **Interpreter** automatically transforms the resulting space explanation into a human-recognizable form, for example, using a visualization (cf. Section 6).

The engine contains four components:

- **Master** coordinates the user inputs and configuration with other components. and also handles the actual explanations and proofs (pass and receive them from the other components).
- **SMT solver** is the core constraint solver, available on demand, answering queries from other components. It automatically determines whether logical formulas are satisfiable, and can generate proofs when they are not.
- **Interpolator** takes an unsatisfiable formula of the form $\psi \wedge \varphi$ and uses Craig interpolation to weaken φ. Initially, the interpolator lets φ be $\varphi_{\mathbf{s}}$ and produces a generalization over the sample point. Optionally, it continues to weaken the generalization iteratively.
- **Unsat core extractor** receives an unsatisfiable conjunctive formula represented as a set of constraints and extracts a smaller unsatisfiable subset in the form of an unsatisfiable core. This results in a shorter and weaker formula, meaning that the explanation is generalized and simplified.

The engine is instantiated by the OpenSMT2 solver [10,30] which integrates a range of techniques into a unified framework, combining standard SMT solving with the computation of unsatisfiable cores and Craig interpolants of varying sizes and logical strengths. It also provides a naive implementation of computing feature-based explanations (Definition 2 and 3).

Users may optionally specify configurations and a sequence of strategies based on their domain knowledge. This provides flexibility to tailor the explanation to their specific needs, but it is not required for the framework to function. The available strategies—detailed below—can be applied in sequence when provided.

- **Generalize (G).** Generalizes an arbitrary space explanation (including e.g. $\varphi_{\mathbf{s}}$, abductive or interval-based explanations) by calling the Interpolator. The strategy is parametric in the interpolation method with different logical strengths. We use arithmetic interpolation algorithms from Section 4.2: `weak` $\mapsto Itp'_F$, `mid` $\mapsto Itp_f$ using $f = 0.5$, and `strong` $\mapsto Itp_F$, all integrated with McMillan's propositional interpolation algorithm Itp_M.
- **Reduce (R).** Calls the Unsat core extractor to simplify an explanation. When applied on top of strategy **G** (denoted as $\mathbf{R} \circ \mathbf{G}$), the resulting explanation is shorter and more readable. Optionally, minimal reduction ($\mathbf{R}_{min}$) arrives at an irreducible explanation, possibly with significant overhead.

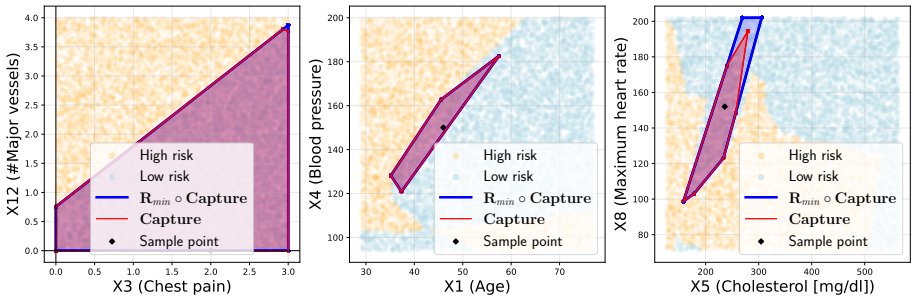

Fig. 4. Approximation of decision boundaries and feature relations using strategies $\mathbf{C}$ and $\mathbf{R}_{min} \circ \mathbf{C}$ within selected pairs of features.

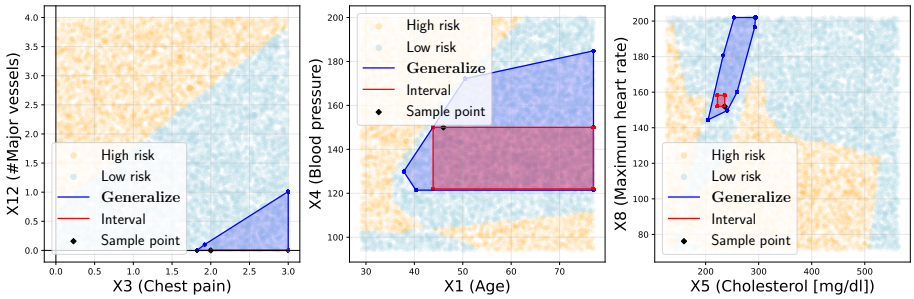

Fig. 5. Generalizing an interval explanation

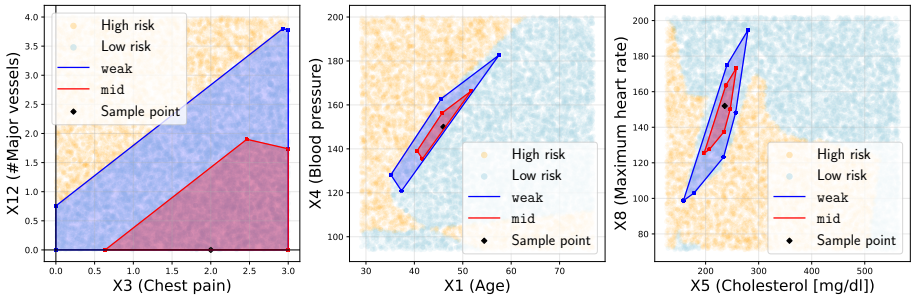

Fig. 6. Explanations produced by $\mathbf{G}$ with different logical strengths

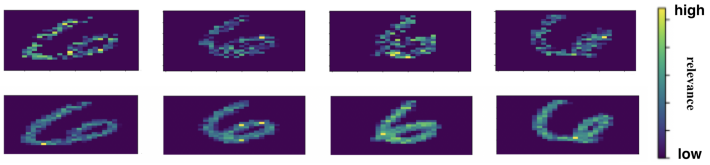

Fig. 7. Visualization of space explanations of MNIST dataset computed by strategy $\mathbf{G}$ with algorithm `weak` (above) and `strong` (below).

– **Capture (C).** Similarly to $\mathbf{G}$, calls the Interpolator but only on the part of the initial formula $\varphi_{\mathbf{s}}$ with constraints over selected features, hence capturing

their mutual relationships. Moreover, using **Reduce** is more efficient on top of strategy **C** than **G** (i.e., **R**∘**C** vs. **R**∘**G**) since the interpolation is focused.

The interpreter transforms the resulting space explanation into a user-friendly representation that also captures meaningful insights into the classification. We demonstrate several visualizations of the generated space explanations in the experiments below.

Evaluation SpEXplAIn has been evaluated [44] and compared to the state-of-the-art abduction-based XAI tool VeriX[10] on test cases of different sizes and different nature, such as medical domain datasets ("heart-attack risk" and "obesity risk") and image classification (MNIST). Thanks to the efficiency of SMT solvers, SpEXplAIn applies in realistic settings. Using the direct encoding of the NN (cf. Section 4.1) with no further optimizations, [44] demonstrated the usefulness of the explanations generated by SpEXplAIn but also the scalability challenges, showing that while using this encoding, SpEXplAIn is less sensitive to the number of input features than VeriX, but more sensitive to the number of hidden layers in the NN. In this paper, we present a snapshot of the experiments to illustrate the benefits of SpEXplAIn, focusing on the quality of explanations. More details, such as the runtimes, are available in [44]. Notably, its main concept of space explanations enables the preservation of important domain information. Moreover, the flexibility of tuning the algorithms to deal with logic formulas accommodates the specific needs of the user. The tutorial accompanying the paper will show further experiments and comparisons[11], using also other models and datasets.

We illustrate the application of SpEXplAIn on *tabular data* using the Heart-attack risk dataset [38]. The task is to predict the risk of a heart attack based on various medical indicators of patients. It contains 13 input features and 2 possible classification outcomes: high or low risk. We trained the NN using one hidden layer with 50 neurons and used a dataset with 303 sample points. Below, we demonstrate that our explanations are meaningful to the user and often confirm the intuition behind the relations of the medical indicators.

SpEXplAIn automatically approximates decision boundaries among all pairs of input features. Figure 4 illustrates the relation between selected pairs (e.g. age and blood pressure) using strategy **Capture** and `weak` interpolation algorithm. Notably, the decision boundaries that can even be non-convex are well approximated while preserving the relationship between the features (cf. yellow and light-blue colors). The figure shows feature relations that resemble real-world phenomena (e.g., chest pain and #major vessels) and at the same time stem from the behavior of the NN with provable guarantees. Moreover, the same experiment illustrates that applying minimal **Reduce** on top of **Capture** (i.e.,

[10] VeriX showed it produces (sound) explanations that involve fewer pixels compared to SHAP and Anchors, but using a significantly higher computation time.

[11] See more experiments with optimizations and more layers or neurons in the NNs at: https://verify.inf.usi.ch/content/experiments-spexplain

$\mathbf{R}_{min} \circ \mathbf{C}$) is useful to improve the accuracy of the obtained explanations (red and blue areas in Figure 4). Another benefit of performing the reduction is that the resulting explanations are more compact and thus easier to communicate.

Furthermore, SpEXplAIn can generalize already existing space explanations, as demonstrated in Figure 5. In this experiment, an interval explanation (cf. Section 3) is first computed based on techniques from related work [35] (see red rectangles) and then widened using strategy **Generalize** (see blue polygons), again with `weak` algorithm. Notably, the generalized explanations approximate the behavior of the NN much more closely than the prior work which is restricted to simplistic feature-based explanations.

Figure 6 illustrates the capabilities of SpEXplAIn to generate explanations of various logical strengths. The use of weaker interpolation algorithms (e.g. `weak`, blue area) results in explanations with larger space compared to stronger ones (e.g. `mid`, red area). Note that the shapes are similar for all sizes.

Overall, these experiments confirm that SpEXplAIn captures feature relations meaningfully, which could be useful for the user.

The other illustration uses the *image classification dataset* MNIST [17], which represents benchmarks with a different kind of sample points, where particular features correspond to pixels arranged into a matrix and are often related locally. The MNIST dataset collects grayscale images of handwritten digits with 784 pixels ranging from 0 to 255. We trained the model using a hidden layer with 200 neurons and experimented on a dataset with 50 sample points.

For this domain, the common task is to identify pixels that are relevant for classification. We used SpEXplAIn to experiment with this task by identifying explanations of different relevance for the visualization of digits. The SpEXplAIn capability of generating explanations of different strengths is very useful for this task. For example, here is a snapshot of experimentation with digit 6 for which we use strategy **G**. For each explanation, Figure 7 illustrates visualizations for particular points extracted using the minimum Euclidean norm, that is, the point closest to the original image (cf. Section 6.1). The figure enables the comparison of interpolation algorithms `weak` and `strong`. The lighter the color, the higher participation in the explanation it represents for particular pixels of the digits. Notably, while both approaches accurately resemble the actual intended digit, applying algorithm `weak` results in explanations that are more fuzzy upon visualization compared to algorithm `strong`. Such observations suggest that stronger interpolation algorithms suit image classification better because they yield more compact, smoother, and easier-to-interpret explanations.

8 Conclusion

This paper surveys the state of the art in logical abduction-based explanations, a concept of provably correct explanations of NN classification w.r.t. a formal specification of the model. In this area, it establishes a unified theoretical framework and encoding and list the state-of-the-art approaches, allowing their comprehensive comparison. Furthermore, it suggests multiple metrics and factors for

estimation of the quality of explanations, allowing their comparison as well. Finally, it illustrates SpEXplAIn, a practical framework for computing most general abduction-based explanations for neural networks. SpEXplAIn preserves provable guarantees of the classification and generates substantially more informative explanations than prior work. Based on SMT solving techniques such as Craig interpolation, it approximates complex decision boundaries and identifies relations between input features that are visualized in a user-friendly way. We illustrate several real-world experiments from different domains with different requirements, highlighting the usefulness of the explanations and the flexibility of their generation.

Acknowledgement This work was conducted as part of the "Formal Reasoning on Neural Networks" project funded by the Hasler Foundation, Switzerland.

References

1. Albarghouthi, A., Dillig, I., Gurfinkel, A.: Maximal specification synthesis. SIGPLAN Not. **51**(1), 789–801 (Jan 2016), https://doi.org/10.1145/2914770.2837628
2. Alt, L., Hyvärinen, A.E.J., Sharygina, N.: LRA interpolants from no man's land. In: Strichman, O., Tzoref-Brill, R. (eds.) HVC. pp. 195–210. Springer International Publishing, Cham (2017)
3. Amir, G., Wu, H., Barrett, C., Katz, G.: An SMT-based approach for verifying binarized neural networks. In: Groote, J.F., Larsen, K.G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 203–222. Springer International Publishing, Cham (2021)
4. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. PLOS One **10**(7) (2015)
5. Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Müller, K.R.: How to Explain Individual Classification Decisions. Journal of Machine Learning Research **11** (2010)
6. Barbosa, H., Barrett, C., Brain, M., Kremer, G., Lachnitt, H., Mann, M., Mohamed, A., Mohamed, M., Niemetz, A., Nötzli, A., Ozdemir, A., Preiner, M., Reynolds, A., Sheng, Y., Tinelli, C., Zohar, Y.: cvc5: A versatile and industrial-strength SMT solver. In: Fisman, D., Rosu, G. (eds.) TACAS. pp. 415–442. Springer International Publishing, Cham (2022)
7. Bassan, S., Katz, G.: Towards Formal XAI: Formally Approximate Minimal Explanations of Neural Networks. In: TACAS. pp. 187 – 207 (2023)
8. Blicha, M., Hyvärinen, A.E.J., Kofroň, J., Sharygina, N.: Decomposing Farkas interpolants. In: Vojnar, T., Zhang, L. (eds.) TACAS. pp. 3–20. Springer International Publishing, Cham (2019)
9. Bohn, B., Garcke, J., Griebel, M.: Algorithmic mathematics in machine learning. SIAM (2024)
10. Bruttomesso, R., Pek, E., Sharygina, N., Tsitovich, A.: The OpenSMT Solver. In: Esparza, J., Majumdar, R. (eds.) TACAS. LNCS, vol. 6015, pp. 150–153. Springer (2010), https://doi.org/10.1007/978-3-642-12002-2_12

11. Chockler, H., Kelly, D.A., Kroening, D., Sun, Y.: Causal explanations for image classifiers. arXiv preprint arXiv:2411.08875 (2024)
12. Christ, J., Hoenicke, J., Nutz, A.: SMTInterpol: An interpolating SMT solver. In: Donaldson, A.F., Parker, D. (eds.) SPIN. Lecture Notes in Computer Science, vol. 7385, pp. 248–254. Springer (2012). https://doi.org/10.1007/978-3-642-31759-0_19, https://doi.org/10.1007/978-3-642-31759-0_19
13. Cimatti, A., Griggio, A., Schaafsma, B., Sebastiani, R.: The MathSAT5 SMT Solver. In: Piterman, N., Smolka, S. (eds.) TACAS. LNCS, vol. 7795, pp. 93–107. Springer (2013)
14. Cimatti, A., Griggio, A., Sebastiani, R.: Computing small unsatisfiable cores in satisfiability modulo theories. Journal of Artificial Intelligence Research **40**, 701–728 (2011)
15. Craig, W.: Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. In: J. of Symbolic Logic. pp. 269–285 (1957)
16. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: TACAS. pp. 337–340. TACAS'08/ETAPS'08, Springer-Verlag, Berlin, Heidelberg (2008), https://doi.org/10.1007/978-3-540-78800-3_24
17. Deng, L.: The MNIST Database of Handwritten Digit Images for Machine Learning Research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012). https://doi.org/10.1109/MSP.2012.2211477
18. Dhurandhar, A., Chen, P.Y., Luss, R., Tu, C.C., Ting, P., Shanmugam, K., Das, P.: Explanations based on the missing: towards contrastive explanations with pertinent negatives. In: NIPS. p. 590–601. NIPS'18, Curran Associates Inc., Red Hook, NY, USA (2018)
19. Dillig, I., Dillig, T., Li, B., McMillan, K.L.: Inductive invariant generation via abductive inference. In: Hosking, A.L., Eugster, P.T., Lopes, C.V. (eds.) OOPSLA. pp. 443–456. ACM (2013). https://doi.org/10.1145/2509136.2509511, https://doi.org/10.1145/2509136.2509511
20. D'Silva, V., Kroening, D., Purandare, M., Weissenbacher, G.: Interpolant strength. In: Barthe, G., Hermenegildo, M. (eds.) VMCAI. pp. 129–145. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
21. Dutertre, B.: Yices 2.2. In: Biere, A., Bloem, R. (eds.) CAV. pp. 737–744. Springer International Publishing, Cham (2014)
22. Elboher, Y.Y., Gottschlich, J., Katz, G.: An abstraction-based framework for neural network verification. In: Lahiri, S.K., Wang, C. (eds.) CAV. pp. 43–65. Springer International Publishing, Cham (2020)
23. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing Higher-layer Features of a Deep Network. Tech. rep., University of Montreal (2009)
24. Ferrari, C., Muller, M.N., Jovanovic, N., Vechev, M.: Complete verification via multi-neuron relaxation guided branch-and-bound. arXiv preprint arXiv:2205.00263 (2022)
25. Ferreira, J., de Sousa Ribeiro, M., Gonçalves, R., Leite, J.: Looking Inside the Black-Box: Logic-based Explanations for Neural Networks. In: KR. pp. 432–442 (8 2022). https://doi.org/10.24963/kr.2022/45, https://doi.org/10.24963/kr.2022/45
26. Gurfinkel, A., Rollini, S.F., Sharygina, N.: Interpolation properties and SAT-based model checking. In: Van Hung, D., Ogawa, M. (eds.) ATVA. pp. 255–271. Springer International Publishing, Cham (2013)
27. Guthmann, O., Strichman, O., Trostanetski, A.: Minimal unsatisfiable core extraction for SMT. In: FMCAD. pp. 57–64 (2016). https://doi.org/10.1109/FMCAD.2016.7886661

28. Henriksen, P., Lomuscio, A.: Efficient neural network verification via adaptive refinement and adversarial search. Ph.D. thesis, Ph. D. Dissertation. Imperial College London (2019)
29. Huang, X., Marques-Silva, J.: From robustness to explainability and back again. arXiv preprint arXiv:2306.03048 (2023)
30. Hyvärinen, A.E.J., Marescotti, M., Alt, L., Sharygina, N.: OpenSMT2: An SMT Solver for Multi-core and Cloud Computing. In: Creignou, N., Berre, D.L. (eds.) SAT. LNCS, vol. 9710, pp. 547–553. Springer (2016), https://doi.org/10.1007/978-3-319-40970-2_35
31. Ignatiev, A., Narodytska, N., Asher, N., Marques-Silva, J.: From contrastive to abductive explanations and back again. In: AIxIA. p. 335–355. Springer-Verlag, Berlin, Heidelberg (2020), https://doi.org/10.1007/978-3-030-77091-4_21
32. Ignatiev, A., Narodytska, N., Marques-Silva, J.: Abduction-based explanations for machine learning models. In: AAAI. AAAI'19/IAAI'19/EAAI'19, AAAI Press (2019), https://doi.org/10.1609/aaai.v33i01.33011511
33. Izza, Y., Huang, X., Ignatiev, A., Narodytska, N., Cooper, M., Marques-Silva, J.: On computing probabilistic abductive explanations. International Journal of Approximate Reasoning **159**, 108939 (2023), https://www.sciencedirect.com/science/article/pii/S0888613X23000701
34. Izza, Y., Huang, X., Morgado, A., Planes, J., Ignatiev, A., Marques-Silva, J.: Distance-restricted explanations: theoretical underpinnings & efficient implementation. In: KR. KR '24 (2024), https://doi.org/10.24963/kr.2024/45
35. Izza, Y., Ignatiev, A., Stuckey, P.J., Marques-Silva, J.: Delivering inflated explanations. In: AAAI. AAAI'24/IAAI'24/EAAI'24, AAAI Press (2024), https://doi.org/10.1609/aaai.v38i11.29170
36. Izza, Y., Marques-Silva, J.: Efficient contrastive explanations on demand. CoRR **abs/2412.18262** (2024). https://doi.org/10.48550/ARXIV.2412.18262, https://doi.org/10.48550/arXiv.2412.18262
37. Izza, Y., Meel, K.S., Marques-Silva, J.: Locally-minimal probabilistic explanations (2024), https://arxiv.org/abs/2312.11831
38. Janosi, A., Steinbrunn, W., Pfisterer, M., Detrano, R.: Heart Disease. UCI Machine Learning Repository (1989), DOI: https://doi.org/10.24432/C52P4X
39. Junker, U.: QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In: National Conference on Artifical Intelligence. p. 167–172. AAAI'04, AAAI Press (2004)
40. Kanav, S., Křetínský, J., Rieder, S.: A Literature Review on Verification and Abstraction of Neural Networks Within the Formal Methods Community. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-75778-5_3, https://doi.org/10.1007/978-3-031-75778-5_3
41. Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., Dill, D.L., Kochenderfer, M.J., Barrett, C.: The marabou framework for verification and analysis of deep neural networks. In: Dillig, I., Tasiran, S. (eds.) CAV. pp. 443–452. Springer International Publishing, Cham (2019)
42. König, M., Bosman, A.W., Hoos, H.H., van Rijn, J.N.: Critically assessing the state of the art in neural network verification. Journal of Machine Learning Research **25**(12), 1–53 (2024), http://jmlr.org/papers/v25/23-0119.html
43. König, M., Bosman, A.W., Hoos, H.H., Van Rijn, J.N.: Critically assessing the state of the art in neural network verification. J. Mach. Learn. Res. **25**(1) (Jan 2024)

44. Labbaf, F., Kolárik, T., Blicha, M., Fedyukovich, G., Wand, M., Sharygina, N.: Space explanations of neural network classification. In: Piskac, R., Rakamaric, Z. (eds.) CAV. Springer International Publishing (2025)
45. Larsen, K., Legay, A., Nolte, G., Schlüter, M., Stoelinga, M., Steffen, B.: Formal methods meet machine learning (f3ml). In: ISoLA, Part III. p. 393–405. Springer-Verlag, Berlin, Heidelberg (2022), https://doi.org/10.1007/978-3-031-19759-8_24
46. Laurel, J., Yang, R., Singh, G., Misailovic, S.: A dual number abstraction for static analysis of clarke jacobians. Proc. ACM Program. Lang. **6**(POPL) (Jan 2022), https://doi.org/10.1145/3498718
47. Leopardi, F., Labbaf, F., Kolárik, T., Wand, M., Sharygina, N.: Interpreting logical explanations of classifying neural networks. In: ESANN (2026), to appear
48. Lundberg, S.M., Lee, S.I.: A Unified Approach to Interpreting Model Predictions. In: NIPS (2017)
49. Mahendran, A., Vedaldi, A.: Understanding Deep Image Representations by Inverting Them. In: CVPR. pp. 5188 – 5196 (2015)
50. Malfa, E.L., Michelmore, R., Zbrzezny, A.M., Paoletti, N., Kwiatkowska, M.: On guaranteed optimal robust explanations for NLP models. In: IJCAI. pp. 2658 – 2665 (2021)
51. Marques-Silva, J.: Logic-Based Explainability in Machine Learning. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-31414-8_2, https://doi.org/10.1007/978-3-031-31414-8_2
52. Marques-Silva, J.: Logic-based explainability: Past, present and future. In: ISoLA, Part IV. p. 181–204. Springer-Verlag, Berlin, Heidelberg (2024), https://doi.org/10.1007/978-3-031-75387-9_12
53. McMillan, K.L.: Interpolation and SAT-based model checking. In: Hunt, W.A., Somenzi, F. (eds.) CAV. pp. 1–13. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
54. Miller, T.: Contrastive explanation: a structural-model approach. Knowl. Eng. Rev. **36**, e14 (2021). https://doi.org/10.1017/S0269888921000102, https://doi.org/10.1017/S0269888921000102
55. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: ACM FAT. p. 607–617. FAT* '20, Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3351095.3372850
56. de Moura, L., Bjørner, N.: Satisfiability Modulo Theories: Introduction and Applications. Communications of the ACM **54**(9), 69 – 77 (2011)
57. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: from an abstract Davis–Putnam–Logemann-Loveland procedure to DPLL(T). Journal of the ACM (JACM) **53**(6), 937–977 (2006), https://doi.org/10.1145/1217856.1217859
58. Prabhu, S., D'Souza, D., Chakraborty, S., Venkatesh, R., Fedyukovich, G.: Weakest precondition inference for non-deterministic linear array programs. In: Finkbeiner, B., Kovács, L. (eds.) TACAS, Part II. Lecture Notes in Computer Science, vol. 14571, pp. 175–195. Springer (2024). https://doi.org/10.1007/978-3-031-57249-4_9, https://doi.org/10.1007/978-3-031-57249-4_9
59. Prabhu, S., Fedyukovich, G., D'Souza, D.: Maximal quantified precondition synthesis for linear array loops. In: Weirich, S. (ed.) ESOP, Part II. Lecture Notes in Computer Science, vol. 14577, pp. 245–274. Springer (2024). https://doi.org/10.1007/978-3-031-57267-8_10, https://doi.org/10.1007/978-3-031-57267-8_10

60. Prabhushankar, M., Kwon, G., Temel, D., AlRegib, G.: Contrastive explanations in neural networks. CoRR **abs/2008.00178** (2020), https://arxiv.org/abs/2008.00178
61. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?": Explaining the predictions of any classifier. In: KDD. p. 1135–1144. KDD '16, Association for Computing Machinery, New York, NY, USA (2016), https://doi.org/10.1145/2939672.2939778
62. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-Precision Model-Agnostic Explanations. In: AAAI (2018)
63. Rollini, S.F., Sery, O., Sharygina, N.: Leveraging interpolant strength in model checking. In: Madhusudan, P., Seshia, S.A. (eds.) Computer Aided Verification. pp. 193–209. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
64. Saxe, A.M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B.D., Cox, D.D.: On the information bottleneck theory of deep learning. In: ICLR. OpenReview.net (2018), https://openreview.net/forum?id=ry_WPG-A-
65. Seshia, S.A., Sadigh, D., Sastry, S.S.: Towards Verified Artificial Intelligence. Communications of the ACM **65**(7), 46 – 55 (2022)
66. Shaw, A., Sarkar, U., Meel, K.S.: Efficient Volume Computation for SMT Formulas. In: Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning. pp. 544–554 (10 2025), https://doi.org/10.24963/kr.2025/53
67. Shih, A., Choi, A., Darwiche, A.: A Symbolic Approach to Explaining Bayesian Network Classifiers. In: IJCAI. pp. 5103 – 5111 (2018)
68. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In: Workshop Proceedings of the ICLR (2014)
69. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proc. ACM Program. Lang. **3**(POPL) (Jan 2019), https://doi.org/10.1145/3290354
70. Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.J., Kolter, J.Z.: Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. Advances in neural information processing systems **34**, 29909–29921 (2021)
71. Wu, H., Isac, O., Zeljić, A., Tagomori, T., Daggitt, M., Kokke, W., Refaeli, I., Amir, G., Julian, K., Bassan, S., et al.: Marabou 2.0: a versatile formal analyzer of neural networks. In: CAV. pp. 249–264. Springer (2024)
72. Wu, M., Li, X., Wu, H., Barrett, C.: Better verified explanations with applications to incorrectness and out-of-distribution detection (2024), https://arxiv.org/abs/2409.03060
73. Wu, M., Wu, H., Barrett, C.: VeriX: Towards verified explainability of deep neural networks. In: Oh, A., Neumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems. vol. 36, pp. 22247–22268. Curran Associates, Inc. (2023)
74. Xu, K., Zhang, H., Wang, S., Wang, Y., Jana, S., Lin, X., Hsieh, C.J.: Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. arXiv preprint arXiv:2011.13824 (2020)
75. Yamaguchi, S., Nishida, K.: Explanation bottleneck models. CoRR **abs/2409.17663** (2024). https://doi.org/10.48550/ARXIV.2409.17663, https://doi.org/10.48550/arXiv.2409.17663
76. Zeiler, M.D., Fergus, R.: Visualizing and Understanding Convolutional Networks. In: ECCV. pp. 818 – 833 (2014)