Towards Formal XAI: Formally Approximate Minimal Explanations of Neural Networks

Shahaf Bassan and Guy Katz(✉)

The Hebrew University of Jerusalem, Jerusalem, Israel
{shahaf.bassan,g.katz}@mail.huji.ac.il

Abstract. With the rapid growth of machine learning, deep neural networks (DNNs) are now being used in numerous domains. Unfortunately, DNNs are "black-boxes", and cannot be interpreted by humans, which is a substantial concern in safety-critical systems. To mitigate this issue, researchers have begun working on explainable AI (XAI) methods, which can identify a subset of input features that are the cause of a DNN's decision for a given input. Most existing techniques are heuristic, and cannot guarantee the correctness of the explanation provided. In contrast, recent and exciting attempts have shown that formal methods can be used to generate provably correct explanations. Although these methods are sound, the computational complexity of the underlying verification problem limits their scalability; and the explanations they produce might sometimes be overly complex. Here, we propose a novel approach to tackle these limitations. We (i) suggest an efficient, verification-based method for finding *minimal explanations*, which constitute a *provable approximation* of the global, minimum explanation; (ii) show how DNN verification can assist in calculating lower and upper bounds on the optimal explanation; (iii) propose heuristics that significantly improve the scalability of the verification process; and (iv) suggest the use of *bundles*, which allows us to arrive at more succinct and interpretable explanations. Our evaluation shows that our approach significantly outperforms state-of-the-art techniques, and produces explanations that are more useful to humans. We thus regard this work as a step toward leveraging verification technology in producing DNNs that are more reliable and comprehensible.

1 Introduction

Machine learning (ML) is a rapidly growing field with a wide range of applications, including safety-critical, high-risk systems in the fields of health care [18], aviation [38] and autonomous driving [12]. Despite their success, ML models, and especially deep neural networks (DNNs), remain "black-boxes" — they are incomprehensible to humans and are prone to unexpected behaviour and errors. This issue can result in major catastrophes [13, 73], and also in poor decision-making due to brittleness or bias [7, 24].

In order to render DNNs more comprehensible to humans, researchers have been working on *explainable AI (XAI)*, where we seek to construct models for explaining and interpreting the decisions of DNNs [50,55–57]. Work to date has focused on heuristic approaches, which provide explanations, but do not provide guarantees about the correctness or succinctness of these explanations [14,32,44]. Although these approaches are an important step, their limitations might result in skewed results, possibly failing to meet the regulatory guidelines of institutions and organizations such as the European Union, the US government, and the OECD [51]. Thus, producing DNN explanations that are provably accurate remains of utmost importance.

More recently, the formal verification community has proposed approaches for providing formal and rigorous explanations for DNN decision making [27,31, 51,59]. Many of these approaches rely on the recent and rapid developments in DNN verification [1,8,9,39]. These approaches typically produce an *abductive explanation* (also known as a *prime implicant*, or *PI-explanation*) [31,58,59]: a minimum subset of input features, which by themselves already determine the classification produced by the DNN, regardless of any other input features. These explanations afford formal guarantees, and can be computed via DNN verification [31].

Abductive explanations are highly useful, but there are two major difficulties in computing them. First, there is the issue of scalability: computing locally minimal explanations might require a polynomial number of costly invocations of the underlying DNN verifier, and computing a globally minimal explanation is even more challenging [10,31,48]. The second difficulty is that users may sometimes prefer "high-level" explanations, not based solely on input features, as these may be easier to grasp and interpret compared to "low-level", complex, feature-based explanations.

To tackle the first difficulty, we propose here new approaches for more efficiently producing verification-based abductive explanations. More concretely, we propose a method for *provably approximating* minimum explanations, allowing stakeholders to use slightly larger explanations that can be discovered much more quickly. To accomplish this, we leverage the recently discovered dual relationship between explanations and contrastive examples [30]; and also take advantage of the sensitivity of DNNs to small adversarial perturbations [64], to compute both lower and upper bounds for the minimum explanation. In addition, we propose novel heuristics for significantly expediting the underlying verification process.

In addressing the second difficulty, i.e. the interpretability limitations of "low-level" explanations, we propose to construct explanations in terms of *bundles*, which are sets of related features. We empirically show that using our method to produce bundle explanations can significantly improve the interpretability of the results, and even the scalability of the approach, while still maintaining the soundness of the resulting explanations.

To summarize, our contributions include the following: (i) We are the first to suggest a method that formally produces sound and minimal abductive explanations that *provably approximate* the global-minimum explanation. (ii) Our three suggested novel heuristics expedite the search for minimal abductive explanations, significantly outperforming the state of the art. (iii) We suggest a novel approach for using bundles to efficiently produce sound and provable explanations that are more interpretable and succinct.

For evaluation purposes, we implemented our approach as a proof-of-concept tool. Although our method can be applied to any ML model, we focused here on DNNs, where the verification process is known to be NP-complete [39], and the scalable generation of explanations is known to be challenging [31,58]. We used our tool to test the approach on DNNs trained for digit and clothing classification, and also compared it to state-of-the-art approaches [31,32]. Our results indicate that our approach was successful in quickly producing meaningful explanations, often running 40% faster than existing tools. We believe that these promising results showcase the potential of this line of work.

The rest of the paper is organized as follows. Sec. 2 contains background on DNNs and their verification, as well as on formal, minimal explanations. Sec. 3 covers the main method for calculating approximations of minimum explanations, and Sec. 4 covers methods for improving the efficiency of calculating these approximations. Sec. 5 covers the use of *bundles* in constructing "high-level", provable explanations. Next, we present our evaluation in Sec. 6. Related work is covered in Sec. 7, and we conclude in Sec. 8.

2 Background

DNNs. A deep neural network (DNN) [46] is a directed graph composed of layers of nodes, commonly called *neurons*. In feed-forward NNs the data flows from the first (*input*) layer, through intermediate (*hidden*) layers, and onto an *output* layer. A DNN's output is calculated by assigning values to its input neurons, and then iteratively calculating the values of neurons in subsequent layers. In the case of *classification*, which is the focus of this paper, each output neuron corresponds to a specific *class*, and the output neuron with the highest value corresponds to the class the input is classified to.

Fig. 1 depicts a simple, feed-forward DNN. The input layer includes three neurons, followed by a weighted sum layer, which calculates an affine transformation of values from the input layer. Given the input $V_1 = [1,1,1]^T$, the second layers computes the values $V_2 = [6,9,11]^T$. Next comes a ReLU layer, which computes the function $\text{ReLU}(x) = \max(0,x)$ for each neuron in the preceding layer, resulting in $V_3 = [6,9,11]^T$. The final (output) layer then computes an affine transformation, resulting in $V_4 = [15,-4]^T$. This indicates that input $V_1 = [1,1,1]^T$ is classified as the category corresponding to the first output neuron, which is assigned the greater value.

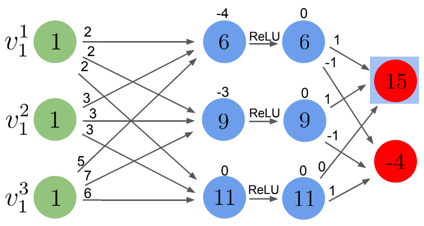


Fig. 1: A simple DNN.

DNN Verification. A DNN verification query is a tuple $\langle P, N, Q\rangle$, where N is a DNN that maps an input vector x to an output vector $y = N(x)$, P is a predicate on x, and Q is a predicate on y. A DNN verifier needs to decide whether there exists an input x_0 that satisfies $P(x_0) \wedge Q(N(x_0))$ (the `SAT` case) or not (the `UNSAT` case). Typically, P and Q are expressed in the logic of real arithmetic [49]. The DNN verification problem is known to be NP-Complete [39].

Formal Explanations. We focus here on explanations for classification problems, where a model is trained to predict a label for each given input. A classification problem is a tuple $\langle F, D, K, N\rangle$ where (i) $F = \{1, ..., m\}$ denotes the features; (ii) $D = \{D_1, D_2 ..., D_m\}$ denotes the domains of each of the features, i.e. the possible values that each feature can take. The entire feature (input) space is hence $\mathbb{F} = D_1 \times D_2 \times ... \times D_m$; (iii) $K = \{c_1, c_2, ..., c_n\}$ is a set of classes, i.e. the possible labels; and (iv) $N : F \to K$ is a (non-constant) classification function (in our case, a neural network). A classification instance is the pair (v, c), where $v \in \mathbb{F}$, $c \in K$, and $c = N(v)$. In other words, v is mapped by the neural network N to class c.

Looking at (v, c), we often wish to know why v was classified as c. Informally, an *explanation* is a subset of features $E \subseteq F$, such that assigning these features to the values assigned to them in v already determines that the input will be classified as c, regardless of the remaining features $F \setminus E$. In other words, even if the values that are *not* in the explanation are changed arbitrarily, the classification remains the same. More formally, given input $v = (v_1, ...v_m) \in \mathbb{F}$ with the classification $N(v) = c$, an explanation (sometimes referred to as an *abductive explanation*, or an *AXP*) is a subset of the features $E \subseteq F$, such that:

$$\forall (x \in \mathbb{F}). \quad [\bigwedge_{i \in E} (x_i = v_i) \to (N(x) = c)] \tag{1}$$

We continue with the running example from Fig. 1. For simplicity, we assume that each input neuron can only be assigned the values 0 or 1. It can be observed that for input $V_1 = [1, 1, 1]^T$, the set $\{v_1^1, v_1^2\}$ is an explanation; indeed, once the first two entries in V_1 are set to 1, the classification remains the same for any value of the third entry (see Fig. 2). We can prove this by encoding a verification query $\langle P, N, Q\rangle = \langle E = v, N, Q_{\neg c}\rangle$, where E is the candidate explanation, and $E = v$ means that we restrict the features in E to their values in v; and $Q_{\neg c}$ implies that the classification is not c. An `UNSAT` result for this query indicates that E is an explanation for instance (v, c).

Clearly, the set of all features constitutes a trivial explanation. However, we are interested in *smaller* explanation subsets, which can provide useful information regarding the decision of the classifier. More precisely, we search for *minimal explanations* and *minimum explanations.* A subset $E \subseteq F$ is a *minimal explanation* (also referred to as a *local-minimal explanation*, or a *subset-minimal explanation*) of instance (v, c) if it is an explanation that ceases to be an explanation if even a single feature is removed from it:

$$\begin{aligned} &(\forall (x \in \mathbb{F}).[\wedge_{i \in E} (x_i = v_i) \to (N(x) = c)]) \wedge \\ &(\forall (j \in E).[\exists (y \in \mathbb{F}).[\wedge_{i \in E \setminus j} (y_i = v_i) \wedge (N(y) \neq c)]) \end{aligned} \tag{2}$$

Fig. 3 demonstrates that $\{v_1^1, v_1^2\}$ is a minimal explanation in our running example: removing any of its features allows mis-classification.

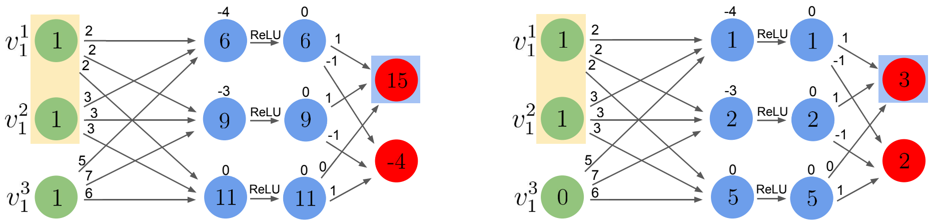


Fig. 2: $\{v_1^1, v_1^2\}$ is an explanation for input $V_1 = [1, 1, 1]^T$.

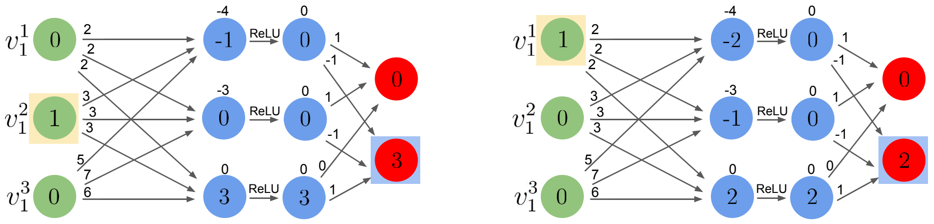


Fig. 3: $\{v_1^1, v_1^2\}$ is a minimal explanation for input $V_1 = [1, 1, 1]^T$.

A *minimum explanation* (sometimes referred to as a *cardinal minimal explanation* or a *PI-explanation*) is defined as a minimal explanation of minimum size; i.e., if E is a minimum explanation, then there does not exist a minimal explanation $E' \neq E$ such that $|E'| < |E|$. Fig. 4 demonstrates that $\{v_1^3\}$ is a minimum explanation for our running example.

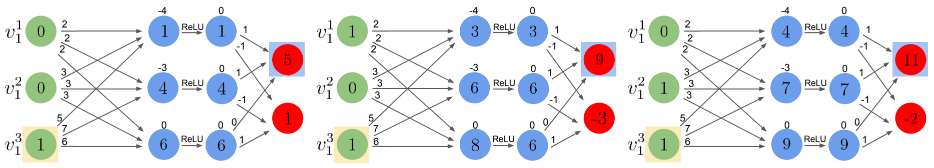


Fig. 4: $\{v_1^3\}$ is a minimum explanation for input $V_1 = [1, 1, 1]^T$.

Contrastive Example. A subset of features $C \subseteq F$ is called a *contrastive example* or a *contrastive explanation (CXP)* if altering the features in C is sufficient to cause the misclassification of a given classification instance (v, c):

$$\exists (x \in \mathbb{F}).[\wedge_{i \in F \setminus C}(x_i = v_i) \wedge (N(x) \neq c)] \tag{3}$$

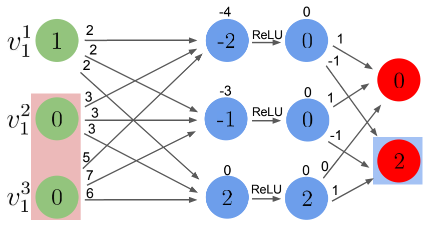


Fig. 5: $\{v_1^2, v_1^3\}$ is a contrastive example for $V_1 = [1, 1, 1]^T$.

A contrastive example for our running example is shown in Fig. 5. Notice that the question of whether a set is a contrastive example can be encoded into a verification query $\langle P, N, Q\rangle = \langle (F \setminus C) = v, N, Q_{\neg c}\rangle$, where a SAT result indicates that C is a contrastive example. As with explanations, smaller contrastive examples are more valuable than large ones. One useful notion is that of a *contrastive singleton*: a contrastive example of size one. A contrastive singleton could represent a specific pixel in an image, the alteration of which could result in misclassification. Such singletons are leveraged in "one-pixel attacks" [64] (see Fig. 16 in the appendix of the full version of this paper [11]). Contrastive singletons have the following important property:

Lemma 1. *Every contrastive singleton is contained in all explanations.*

The proof appears in Sec. A of the appendix of the full version of this paper [11]. Lemma 1 implies that each contrastive singleton is contained in all minimal/minimum explanations.

We consider also the notion of a *contrastive pair*, which is a contrastive example of size 2. Clearly, for any pair of features (u, v) where u or v are contrastive singletons, (u, v) is a contrastive pair; however, when we next refer to contrastive pairs, we consider only pairs that *do not* contain any contrastive singletons. Likewise, for every $k > 2$, we can consider contrastive examples of size k, and we exclude from these any contrastive examples of sizes $1, \ldots, k-1$ as subsets.

We state the following theorem, whose proof also appears in Sec. A of the appendix of the full version of this paper [11]:

Lemma 2. *All explanations contain at least one element of every contrastive pair.*

The theorem can be generalized to any $k > 2$; and can be used in showing that the *minimum hitting set (MHS)* of all contrastive examples is exactly the minimum explanation [29, 54] (see Sec. B of the appendix of the full version of this paper [11]). Further, the theorem implies a duality between contrastive examples and explanations [30, 34]: a minimal hitting set of all contrastive examples constitutes a minimal explanation, and a minimal hitting set of all explanations constitutes a minimal contrastive example.

3 Provable Approximations for Minimal Explanations

State-of-the-art approaches for finding minimum explanations exploit the MHS duality between explanations and contrastive examples [31]. The idea is to iteratively compute contrastive examples, and then use their MHS as an under-approximation for the minimum explanation. Finding this MHS is an NP-complete problem, and is difficult in practice as the number of contrastive examples increases [20]; and although the MHS can be approximated using maximum satisfiability (MaxSAT) or mixed integer linear programming (MILP) solvers [26,47], existing approaches tackle simpler ML models, such as decision trees [33,36], but face scalability limitations when applied to DNNs [31,58]. Further, enumerating all contrastive examples may in itself take exponential time. Finally, recall that DNN verification is an NP-Complete problem [39]; and so dispatching a verification query to identify each explanation or contrastive example is also very slow, when the feature space is large. Finding *minimal* explanations may be easier [31], but may converge to larger and less meaningful explanations, while still requiring a linear number of calls to the underlying verifier. Our approach, described next, seeks to mitigate these difficulties.

Our overall approach is described in Algorithm 1. It is comprised of two separate threads, intended to be run in parallel. The *upper bounding thread* (T_{UB}) is responsible for computing a minimal explanation. It starts with the entire feature space, and then gradually reduces it, until converging to a minimal explanation. The size of the presently smallest explanation is regarded as an upper bound (UB) for the size of the minimum explanation. Symmetrically, the *lower bounding thread* (T_{LB}) attempts to construct small contrastive sets, used for computing a lower bound (LB) on the size of the minimum explanation. Together, these two bounds allow us to compute the approximation ratio between the minimal explanation that we have discovered and the minimum explanation. For instance, given a minimal explanation of size 7 and a lower bound of size 5, we can deduce that our explanation is at most $\frac{\text{UB}}{\text{LB}} = \frac{7}{5}$ times larger than the minimum. The two threads share global variables that indicate the set of contrastive singletons (Singletons), the set of contrastive pairs (Pairs), the upper and lower bounds (UB, LB), and the set of features that were determined not to participate in the explanation and are "free" to be set to any value (Free). The output of our algorithm is a minimal explanation (F∖Free), and the approximation ratio ($\frac{\text{UB}}{\text{LB}}$). We next discuss each of the two threads in detail.

Algorithm 1 Minimal Explanation Search

```
Input N (Neural network), F (features), v (input values), c (class prediction)
1: Singletons, Pairs, Free ← ∅, UB ← $|F|$, LB ← 0        ▷ Global variables
2: Launch thread $T_{\text{UB}}$
3: Launch thread $T_{\text{LB}}$
4: return F∖Free, $\frac{\text{UB}}{\text{LB}}$
```

The Upper Bounding Thread (T_{UB}). This thread, whose pseudocode appears in Algorithm 2, follows the framework proposed by Ignatiev et al. [31]: it seeks a minimal explanation by starting with the entire feature space, and then iteratively attempting to remove individual features. If removing a feature allows misclassification, we keep it as part of the explanation; otherwise, we remove it and continue. This process issues a single verification query for each feature, until converging to a minimal explanation (lines 2–8). Although this naïve search is guaranteed to converge to a minimal explanation, it needs not to converge to a *minimum* explanation; and so we apply a more sophisticated ordering scheme, similar to the one proposed by [32], where we use some heuristic model as a way for assigning weights of importance to each input feature. We then check the "least important" input features first, since freeing them has a lower chance of causing a misclassification, and they are consequently more likely to be successfully removed. We then continue iterating over features in ascending order of importance, hopefully producing small explanations.

Algorithm 2 T_{UB}: Upper Bounding Thread

```
1: Use a heuristic model to sort $F$'s features by ascending relevance
2: for each $f \in F$ do
3:     Explanation ← F∖Free
4:     if Verify((Explanation∖{f})=v,N,$Q_{\neg c}$) is UNSAT then
5:         Free ← Free ∪ {f}
6:         UB ← UB − 1
7:     end if
8: end for
```

The Lower Bounding Thread (T_{LB}). The pseudocode for the lower bounding thread (T_{LB}) appears in Algorithm 3. In lines 1–6, the thread searches for contrastive singletons. Neural networks were shown to be very sensitive to adversarial attacks [25] — slight input perturbations that cause misclassification (e.g., the aforementioned one-pixel attack [64]) — and this suggests that contrastive sets, and in particular contrastive singletons, exist in many cases. We observe that identifying contrastive singletons is computationally cheap: by encoding Eq. 3 as a verification query, once for each feature, we can discover all singletons; and in these queries all features but one are fixed, which empirically allows verifiers to dispatch them quickly.

The rest of T_{LB} (lines 9–13) performs a similar process, but with contrastive pairs (which do not contain contrastive singletons as one of their features). We use verification queries to identify all such pairs, and then attempt to find their MHS. We observe that finding the MHS of all contrastive pairs is the 2-MHS problem, which is a reformalization of the *minimum vertex cover* problem (see Sec. B of the appendix of the full version of this paper [11]). Since this is an easier problem than the general MHS problem, solving it with MAX-SAT or MILP often converges quickly. In addition, the minimum vertex cover algorithm has a linear 2-approximating greedy algorithm, which can be used for finding a lower bound in cases of large feature spaces.

More formally, T_{LB} performs an efficient computation of the following bound:

$$\text{LB} = |\text{Singletons}| + |\text{MVC}(\text{Pairs})| \leq \text{MHS}(\text{Cxps}) = E_M \tag{4}$$

Algorithm 3 T_{LB}: Lower Bounding Thread

```
1:  for each $f \in F$ do                                   ▷ Find all singletons
2:      if Verify((F∖{f}=v,N,$Q_{\neg c}$) is SAT then
3:          Singletons ← Singletons ∪ {$f$}
4:          LB ← LB +1
5:      end if
6:  end for
7:
8:  AllPairs ← Distinct pairs of F∖Singletons
9:  for each (a,b) ∈ AllPairs do                               ▷ Find all pairs
10:     if Verify((F∖{a,b}=v,N,$Q_{\neg c}$) is SAT then
11:         Pairs ←Pairs ∪ {$(a, b)$}
12:     end if
13: end for
14: LB ← LB + MVC(Pairs)
```

where MVC is the minimum vertex cover, Cxps denotes the set of all contrastive examples, and E_M is the size of the minimum explanation.

It is worth mentioning that this approach can be extended to use contrastive examples of larger sizes ($k = 3, 4, \ldots$), as specified in Sec. C of the appendix of the full version of this paper. The fact that small contrastive examples, such as singletons, exist in large, state-of-the-art DNNs with large inputs [21, 64] suggests that useful approximations exist in large DNNs. In our experiments, we observed that using only singletons and pairs affords good approximations, without incurring overly expensive computations by the underlying verifier.

4 Finding Minimal Explanations Efficiently

Algorithm 1 is the backbone of our approach, but it suffers from limited scalability — particularly, in T_{UB}. As the execution of T_{UB} progresses, and as additional features are "freed", the quickly growing search space slows down the underlying verifier. Here we propose three different methods for expediting this process, by reducing the number of verification queries required.

Method 1: Using Information from $T_{\mathbf{LB}}$. We suggest to leverage the contrastive examples found by T_{LB} to expedite T_{UB}. The process is described in Algorithm 4. In line 3, T_{LB} is queried for the current set of contrastive singletons, which we know must be part of any minimal explanation. These are subtracted from the RemainingFeatures set (features left for T_{UB} to query), and consequently will not be added to the Free set — i.e., they are marked as part of the current explanation. In addition, for any contrastive pair (a, b) found by T_{LB}, either a or b must appear in any minimal explanation; and so, our algorithm skips checking the case where both a and b are removed from F (Line 8). (the method could also be extended to contrastive sets of greater cardinality.)

Algorithm 4 T_{UB} using information from T_{LB}

```
1: Use a heuristic model to sort $F$ by ascending relevance
2: RemainingFeatures ← F∖Singletons
3: for each f ∈ RemainingFeatures do
4:     Explanation ← F∖Free
5:     if Verify((Explanation∖{f})=v,N,$Q_{\neg c}$) is UNSAT then
6:         Free ← Free ∪ {f}
7:         UB ← UB − 1
8:         Delete all features in a pair with f from RemainingFeatures
9:     end if
10: end for
```

Method 2: Binary Search. Sorting the features being considered in ascending order of importance can have a significant effect on the size of the explanation found by Algorithm 2. Intuitively, a "perfect" heuristic model would assign the greatest weights to all features in the minimum explanation, and so traversing features in ascending order would first discover all the features that can be removed (`UNSAT` verification queries), followed by all the features that belong in the explanation (`SAT` queries). In this case, a sequential traversal of the features in ascending order is quite wasteful, and it is much better to perform a binary search to find the point where the answer flips from `UNSAT` to `SAT`.

Of course, in practice, the heuristic models are not perfect, leading to potential cases with multiple "flips" from `SAT` to `UNSAT`, and vice versa. Still, if the heuristic is good in practice (which is often the case; see Sec. 6), these flips are scarce. Thus, we propose to perform multiple binary searches, each time identifying one `SAT` query (i.e., a feature added to the explanation). Observe that each time we hit an `UNSAT` query, this indicates that all the queries for features with lower priorities would also yield `UNSAT` — because if "freeing" multiple features cannot change the classification, changing fewer features certainly cannot. Thus, we are guaranteed to find the first `SAT` query in each iteration, and soundness is maintained. This process is described in Algorithm. 6 and in Fig. 14 in the appendix of the full version of this paper [11].

Method 3: Local-Singleton Search. Let N be a DNN, and let x be an input point whose classification we seek to explain. As part of Algorithm 2, T_{UB} iteratively "frees" certain input features, allowing them to take arbitrary values, as it continues to search for features that must be included in the explanation. The increasing number of free features enlarges the search space that the underlying verifier must traverse, thus slowing down verification. We propose to leverage the hypothesis that input points nearby x that are misclassified tend to be clustered; and so, it is beneficial to *fix* the free features to "bad" values, as opposed to letting them take on arbitrary values. We speculate that this will allow the verifier to discover satisfying assignments much more quickly.

This enhancement is shown in Algorithm 5. Given a set Free of features that were previously freed, we fix their values according to some satisfying assignment previously discovered. Thus, the verification of any new feature that we consider is similar to the case of searching for contrastive singletons, which, as we already know, is fairly fast. See Fig. 15 in the appendix of the full version of this paper [11] for an illustration. The process can be improved further by fixing the freed features to small neighborhoods of the previously discovered satisfying assignment (instead of its exact values), to allow some flexibility while still keeping the query's search space small.

Algorithm 5 T_{UB} using local-singleton search

```
 1: Use a heuristic model to sort $F$ by ascending relevance
 2: RemainingFeatures ← F∖Singletons
 3: for each f ∈ RemainingFeatures do
 4:     Explanation ← F∖Free
 5:     if Verify((Explanation∖{f})=v,N,$Q_{\neg c}$) is UNSAT then
 6:         Free ← Free ∪ {f}
 7:         UB ← UB − 1
 8:     else
 9:         Extract counter example C
10:         LocalSingletons ← ∅
11:         for each $f'$ ∈ RemainingFeatures do
12:             if Verify(Explanation∖{$f'$} = $C$,N,$Q_{\neg c}$) is SAT then
13:                 LocalSingletons ← LocalSingletons ∪ {$f'$}
14:             end if
15:         end for
16:         RemainingFeatures ← RemainingFeatures ∖ LocalSingletons
17:     end if
18: end for
```

5 Minimal Bundle Explanations

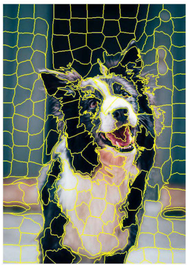


Fig. 6: Partition input's features into bundles.

So far, we presented methods for generating explanations within a given approximation ratio of the minimum explanation (Sec. 3), and for expediting the computation of these explanations (Sec. 4) — in order to improve the scalability of our explanation generation mechanism. Next, we seek to tackle the second challenge from Sec. 1, namely that these explanations may be too low-level for many users. To address this challenge, we focus on *bundles*, which is a topic well covered in the ML [63] and heuristic XAI literature [50,55] (commonly known as "super-pixels" for computer-vision tasks). Intuitively, bundles are a partitioning of the features into disjoint sets (an illustration appears in Fig. 6). The idea, which we later validate empirically, is that providing explanations in terms of bundles is often easier for humans to comprehend. As an added bonus, using bundles also curtails the search space that the verifier must traverse, expediting the process even further.

Given a feature space $F = \{1, ..., m\}$, a bundle b is just a subset $b \subseteq F$. When dealing with the set of all bundles $B = \{b_1, b_2, ...b_n\}$, we require that they form a partitioning of F, namely $F = \cup b_i$. We define a *bundle explanation* E_B for a classification instance (v, c) as a subset of bundles, $E_B \subseteq B$, such that:

$$\forall(x \in \mathbb{F}).[\wedge_{i \in \cup E_B}(x_i = v_i) \to (N(x) = c)] \tag{5}$$

The following theorem then connects bundle explanations and explicit, non-bundle explanations:

Theorem 1. *The union of features in a bundle explanation is an explanation.*

The proof directly follows from Eqs. 1 and 5. We note that this definition of bundles implies that features that are not part of the bundle explanation (i.e. features contained in *"free" bundles*) are "free" to be set to any possible value. Another possible alternative for defining bundles could be to allow features in "free" bundles to only change in the same, coordinated manner. We focus here on the former definition, and leave the alternative definition for future work.

Many of the aforementioned results and definitions for explanations can be extended to bundle explanations. In a similar manner to Eq. 5, we can define the notions of minimal and minimum bundle explanations, a contrastive bundle singleton, and contrastive bundle pairs (see Sec. D of the appendix of the full version of this paper [11]). Theorems 1 and 2 can be extended to bundle explanations in a straightforward manner. It then follows that all bundle explanations contain all contrastive singleton bundles, and that all bundle explanations contain at least one bundle of any contrastive bundle pair.

Our method from Secs. 3 and 4 can be similarly performed on bundles rather than on features, and T_{UB} would then be used for calculating a minimal bundle explanation, rather than a minimal explanation. Regarding the aforementioned approximation ratio, we discuss and evaluate two different methods for obtaining it. The first, natural approach is to apply our techniques from Sec. 3 on bundle explanations, thus obtaining a provable approximation for a *minimum bundle explanation.* The upper bound is trivially derived by the size of the bundle explanation found by T_{UB}, whereas the lower bound calculation requires assigning a cost to each bundle, representing the number of features it contains. This is done via a known notion of *minimum hitting sets of bundles (MHSB) [6]* and using minimum *weighted* vertex cover for the approximation of contrastive bundle pairs. This method, which is almost identical to the one mentioned in Sec. 3, is formalized in Sec. D of the appendix of the full version of this paper [11].

The second approach is to calculate an approximation ratio with respect to a regular, non-bundle minimum explanation. The minimal bundle explanation found by T_{UB} is an upper bound on the minimum non-bundle explanation following theorem 5. For computing a lower bound, we can analyze contrastive bundle examples; extract from them contrastive non-bundle examples; and then use the duality property, compute an MHS of these contrastive examples, and derive lower bounds for the size of the minimum explanation. We formalize techniques for performing this calculation in Sec. E of the appendix of the full version of this paper [11].

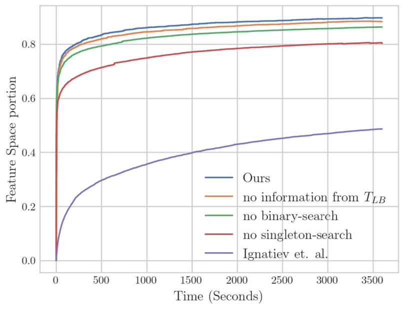
6 Evaluation

Implementation and Setup. For evaluation purposes, we created a proof-of-concept implementation of our approach as a Python framework. Currently, the framework uses the Marabou verification engine [41] as a backend, although other engines may be used. Marabou is a Simplex-based DNN verification framework that is sound and complete [5,39–41,68,69], and which includes support for proof production [35], abstraction [15, 16, 52, 60, 67, 72], and optimization [62]; and has been used in various settings, such as ensemble selection [3], simplification [22, 43] repair [23, 53], and verification of reinforcement-learning based systems [2,4,17]. For sorting features by their relevance, we used the popular XAI method LIME [55]; although again, other heuristics could be used. The MVC was calculated using the classic 2-approximating greedy algorithm. All experiments reported were conducted on x86-64 Gnu/Linux-based machines, using a single Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz core, with a 1-hour timeout.

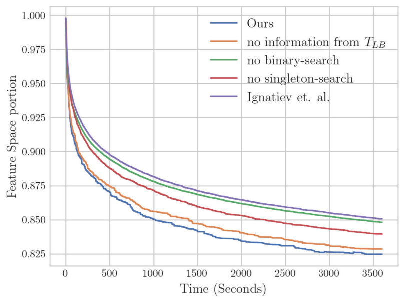
Benchmarks. As benchmarks, we used DNNs trained over the MNIST dataset for handwritten digit recognition [45]. These networks classify 28×28 grayscale images into the digits $0, \ldots, 9$. Additionally, we used DNNs trained over the Fashion-MNIST dataset [71], which classify 28×28 grayscale images into 10 clothing categories (“Dress”, “Coat”, etc.) For each of these datasets we trained a DNN with the following architecture: (i) an input layer (which corresponds to the image) of size 784; (ii) a fully connected hidden layer with 30 neurons; (iii) another fully connected hidden layer, with 10 neurons; and (iv) a final, softmax layer with 10 neurons, corresponding to the 10 possible output classes. The accuracy of the MNIST DNN was 96.6%, whereas that of the Fashion-MNIST DNN was 87.6%. (We note that we configured LIME to ignore the external border pixels of each input, as these are not part of the actual image.)

In selecting the classification instances to be explained for these networks, we targeted input points where the network was not confident — i.e., where the winning label did not win by a large margin. The motivation for this choice is that explanations are most useful and relevant in cases where the network's decision is unclear, which is reflected in lower confidence scores. Additionally, explanations of instances with lower confidence tend to be larger, facilitating the process of extensive experimentation. We thus selected the 100 inputs from the MNIST and the Fashion-MNIST datasets where the networks demonstrated the lowest confidence scores — i.e., where the difference between the winning output score and the runner-up class score was minimal.

Experiments. Our first goal was to compare our approach to that of Ignatiev et al. [31], which is the current state of the art in verification-based explainability of DNNs. Other approaches consider other ML types, such as decision trees [33,36], or focus on alternative definitions for abductive explanations [42, 70] and are thus not comparable. Because the implementation used in [31] is unavailable, we implemented their approach, using Marabou as the underlying verifier for a fair comparison. In addition, we used the same heuristic model, LIME, for sorting



(a) Average portion of features verified to participate in the explanation.



(b) Average explanation size.

Fig. 7: Our full and ablation-based results, compared to the state of the art for finding minimal explanations on the MNIST dataset.

the input features' relevance. Fig. 7 depicts a comparison of the two approaches, over the MNIST benchmarks. The Fashion-MNIST results were similar, but since the Fashion-MNIST network had lower accuracy it tended to produce larger explanations with lower run-times, resulting in less meaningful evaluations (due to space limitations, these results appear in Fig. 12 in the appendix of the full version of this paper [11]). We compared the approaches according to two criteria: the portion of input features whose participation in the explanation was verified, over time (part (a) of Fig. 7), and the average size of the presently obtained explanation over time, also presented as a fraction of the total number of input features (part (b)). The results indicate that our method significantly improves over the state of the art, verifying the participation of 40.4% additional features, on average, and producing explanations that are 9.7% smaller, on average, at the end of the 1-hour time limit. Furthermore, our method timed out on 10% fewer benchmarks. We regard this as compelling evidence of the potential of our approach to produce more efficient verification-based XAI.

We also looked into comparing our approach to heuristic, non-verification-based approaches, such as LIME itself; but these comparisons did not prove to be meaningful, as the heuristic approaches typically solved benchmarks very quickly, but very often produced incorrect explanations. This matches the findings reported in previous work [14,32].

Next, we set out to evaluate the contribution of each of the components implemented within our framework to overall performance, using an ablation study. Specifically, we ran our framework with each of the components mentioned in Sec. 4, i.e. (i) information exchange between T_{UB} and T_{LB}; (ii) the binary search in T_{UB}; and (iii) local-singleton search, turned off. The results on the MNIST benchmarks appear in Fig. 7; see Fig. 12 in the appendix of the full version of this paper [11] for the Fashion-MNIST results. Our experiments revealed that each of the methods mentioned in Sec. 4 had a favorable impact on both the average portion of features verified, and the average size of the discovered explanation, over time. Fig 7a indicates that the local-singleton search method, used for efficiently proving that features are bound to be *included* in the explanation, was the most significant in reducing the number of features remained for verifying, thus substantially increasing the portion of verified features. Moreover, Fig. 7b indicates that the binary search method, which is used for grouping `UNSAT` queries and proving the *exclusion* of features from the explanation, was the most significant for more efficiently obtaining smaller-sized explanations, over time.

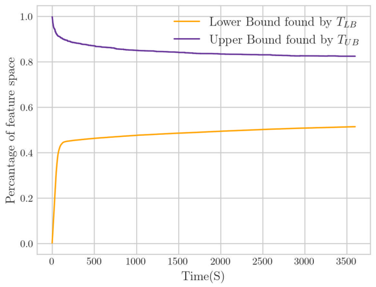
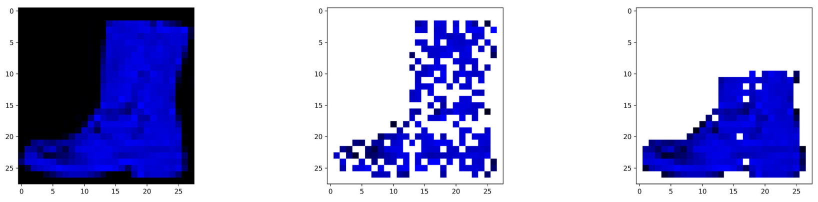


Fig. 8: Average approximation of *minimum* explanation over time.

Our second goal was to evaluate the quality of the *minimum* explanation approximation of our method (using the lower/upper bounds) over time. Results are averaged over all benchmarks of the MNIST dataset and are presented in Fig. 8 (similar results on Fashion-MNIST appear in Fig. 13 in the appendix of the full version of this paper [11]). The upper bound represents the average size of the explanation discovered by T_{UB} over time, whereas the lower bound represents the average lower bound discovered by T_{LB} over time. It can be seen that initially, there is a steep increase in the size of the lower bound, as T_{LB} discovered many contrastive singletons. Later, as we begin iterating over contrastive pairs, the verification queries take longer to solve, and progress becomes slower. The average approximation ratio achieved after an hour was 1.61 for MNIST and 1.19 for Fashion-MNIST.

For our third experiment, we set out to assess the improvements afforded by bundles. We repeated the aforementioned experiments, this time using sets of features representing bundles instead of the features themselves. The segmentation into bundles was performed using the *quickshift* method [65], with LIME again used for assigning relevance to each bundle [55]. We approximate the sizes of the bundle explanations in terms of both the minimum bundle explanation as well as the minimum (non-bundle) explanation (as mentioned in Sec. 5 and in Sec. E of the appendix of the full version of this paper [11]). The bundle configuration showed drastic efficiency improvements, with none of the experiments timing out within the 1-hour time limit, thus improving the portion of timeouts on the MNIST dataset by 84%. The efficiency improvement was obtained at the expense of explanation size, resulting in a decrease of 352% in the approximation ratios obtained for MNIST and 39% for Fashion-MNIST. Nevertheless, when calculating the approximation in terms of the *minimum bundle explanation*, an increase of 12% and 8% was obtained for MNIST and Fashion-MNIST (results are summarized in Table 1 in the appendix of the full version of this paper [11]). For a visual evaluation, we performed the same set of experiments for both bundle and non-bundle implementations, using instances with high confidence rates to obtain smaller-sized explanations that could be more easily interpreted. A



(a) Original Image (b) Explanation (c) Bundle explanation

Fig. 9: Minimal explanations and bundle explanations found by our method on the Fashion-MNIST dataset. White pixels are not part of the explanation.

sample of these results is presented in Fig. 9. Empirically, we observe that the bundle-produced explanations are less complex and more comprehensible.

Overall, we regard our results as compelling evidence that verification-based XAI can soundly produce meaningful explanations, and that our improvements can indeed significantly improve its runtime.

7 Related Work

Our work is another step in the ongoing quest for formal explainability of DNNs, using verification [19, 27, 31, 58]. Related approaches have applied enumeration of contrastive examples [30, 31], which is also an ingredient of our approach. Other approaches focus on producing abductive explanations around an epsilon environment [42, 70]. Similar work has been carried out for decision sets [33], lists [28] and trees [36], where the problem appears to be simpler to solve [36]. Our work here tackles DNNs, which are known to be more difficult to verify [39].

Prior work has also sought to produce approximate explanations, e.g., by using δ-relevant sets [37,66]. This line of work has focused on probabilistic methods for generating explanations, which jeopardizes soundness. There has also been extensive work in heuristic XAI [50, 55, 56, 61], but here, too, the produced explanations are not guaranteed to be correct.

8 Conclusion

Although DNNs are becoming crucial components of safety-critical systems, they remain "black-boxes", and cannot be interpreted by humans. Our work seeks to mitigate this concern, by providing formally correct explanations for the choices that a DNN makes. Since discovering the minimum explanations is difficult, we focus on approximate explanations, and suggest multiple techniques for expediting our approach — thus significantly improving over the current state of the art. In addition, we propose to use bundles to efficiently produce more meaningful explanations. Moving forward, we plan to leverage lightweight DNN verification techniques for improving the scalability of our approach [49], as well as extend it to support additional DNN architectures.

References

1. M. Akintunde, A. Kevorchian, A. Lomuscio, and E. Pirovano. Verification of RNN-Based Neural Agent-Environment Systems. In *Proc. 33rd AAAI Conf. on Artificial Intelligence (AAAI)*, pages 197–210, 2019.
2. G. Amir, D. Corsi, R. Yerushalmi, L. Marzari, D. Harel, A. Farinelli, and G. Katz. Verifying Learning-Based Robotic Navigation Systems, 2022. Technical Report. https://arxiv.org/abs/2205.13536.
3. G. Amir, G. Katz, and M. Schapira. Verification-Aided Deep Ensemble Selection. In *Proc. 22nd Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 27–37, 2022.
4. G. Amir, M. Schapira, and G. Katz. Towards Scalable Verification of Deep Reinforcement Learning. In *Proc. 21st Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 193–203, 2021.
5. G. Amir, H. Wu, C. Barrett, and G. Katz. An SMT-Based Approach for Verifying Binarized Neural Networks. In *Proc. 27th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 203–222, 2021.
6. E. Angel, E. Bampis, and L. Gourvès. On the Minimum Hitting Set of Bundles Problem. *Theoretical Computer Science*, 410(45):4534–4542, 2009.
7. J. Angwin, J. Larson, S. Mattu, and L. Kirchner. Machine Bias. *Ethics of Data and Analytics*, pages 254–264, 2016.
8. G. Avni, R. Bloem, K. Chatterjee, T. Henzinger, B. Konighofer, and S. Pranger. Run-Time Optimization for Learned Controllers through Quantitative Games. In *Proc. 31st Int. Conf. on Computer Aided Verification (CAV)*, pages 630–649, 2019.
9. T. Baluta, S. Shen, S. Shinde, K. Meel, and P. Saxena. Quantitative Verification of Neural Networks And its Security Applications. In *Proc. 26th ACM Conf. on Computer and Communication Security (CCS)*, pages 1249–1264, 2019.
10. P. Barceló, M. Monet, J. Pérez, and B. Subercaseaux. Model interpretability through the lens of computational complexity. *Advances in neural information processing systems*, 33:15487–15498, 2020.
11. S. Bassan and G. Katz. Towards Formally Approximate Minimal Explanations of Neural Networks, 2022. Technical Report. https://arxiv.org/abs/2210.13915.
12. M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to End Learning for Self-Driving Cars, 2016. Technical Report. http://arxiv.org/abs/1604.07316.
13. CACM. A Case Against Mission-Critical Applications of Machine Learning. *Communications of the ACM*, 62(8):9–9, 2019.
14. O.-M. Camburu, E. Giunchiglia, J. Foerster, T. Lukasiewicz, and P. Blunsom. Can I Trust the Explainer? Verifying Post-Hoc Explanatory Methods, 2019. Technical Report. http://arxiv.org/abs/1910.02065.
15. Y. Elboher, E. Cohen, and G. Katz. Neural Network Verification using Residual Reasoning. In *Proc. 20th Int. Conf. on Software Engineering and Formal Methods (SEFM)*, pages 173–189, 2022.
16. Y. Elboher, J. Gottschlich, and G. Katz. An Abstraction-Based Framework for Neural Network Verification. In *Proc. 32nd Int. Conf. on Computer Aided Verification (CAV)*, pages 43–65, 2020.

17. T. Eliyahu, Y. Kazak, G. Katz, and M. Schapira. Verifying Learning-Augmented Systems. In *Proc. Conf. of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 305–318, 2021.
18. A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean. A Guide to Deep Learning in Healthcare. *Nature Medicine*, 25(1):24–29, 2019.
19. T. Fel, M. Ducoffe, D. Vigouroux, R. Cadène, M. Capelle, C. Nicodème, and T. Serre. Don't Lie to Me! Robust and Efficient Explainability with Verified Perturbation Analysis, 2022. Technical Report. http://arXivpreprintarXiv:2202.07728.
20. A. Gainer-Dewar and P. Vera-Licona. The Minimal Hitting Set Generation Problem: Algorithms and Computation. *SIAM Journal on Discrete Mathematics*, 31(1):63–100, 2017.
21. S. Garg and G. Ramakrishnan. BAE: Bert-Based Adversarial Examples for Text Classification, 2020. Technical Report. https://arxiv.org/abs/2004.01970.
22. S. Gokulanathan, A. Feldsher, A. Malca, C. Barrett, and G. Katz. Simplifying Neural Networks using Formal Verification. In *Proc. 12th NASA Formal Methods Symposium (NFM)*, pages 85–93, 2020.
23. B. Goldberger, Y. Adi, J. Keshet, and G. Katz. Minimal Modifications of Deep Neural Networks using Verification. In *Proc. 23rd Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, pages 260–278, 2020.
24. I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples, 2014. Technical Report. http://arxiv.org/abs/1412.6572.
25. S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial Attacks on Neural Network Policies, 2017. Technical Report. http://arxiv.org/abs/1702.02284.
26. IBM. The CPLEX Optimizer, 2018.
27. A. Ignatiev. Towards Trustable Explainable AI. In *Proc. 29th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 5154–5158, 2020.
28. A. Ignatiev and J. Marques-Silva. SAT-Based Rigorous Explanations for Decision Lists. In *Proc. 24th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT)*, pages 251–269, 2021.
29. A. Ignatiev, A. Morgado, and J. Marques-Silva. Propositional Abduction with Implicit Hitting Sets, 2016. Technical Report. http://arxiv.org/abs/1604.08229.
30. A. Ignatiev, N. Narodytska, N. Asher, and J. Marques-Silva. From Contrastive to Abductive Explanations and Back Again. In *Proc. 19th Int. Conf. of the Italian Association for Artificial Intelligence (AIxIA)*, pages 335–355, 2020.
31. A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-Based Explanations for Machine Learning Models. In *Proc. 33rd AAAI Conf. on Artificial Intelligence (AAAI)*, pages 1511–1519, 2019.
32. A. Ignatiev, N. Narodytska, and J. Marques-Silva. On Validating, Repairing and Refining Heuristic Ml Explanations, 2019. Technical Report. http://arxiv.org/abs/1907.02509.
33. A. Ignatiev, F. Pereira, N. Narodytska, and J. Marques-Silva. A SAT-Based Approach to Learn Explainable Decision Sets. In *Proc. 9th Int. Joint Conf. on Automated Reasoning (IJCAR)*, pages 627–645, 2018.
34. A. Ignatiev, A. Previti, M. Liffiton, and J. Marques-Silva. Smallest MUS Extraction with Minimal Hitting Set Dualization. In *Proc. 21st Int. Conf. on Principles and Practice of Constraint Programming (CP)*, pages 173–182, 2015.

35. O. Isac, C. Barrett, M. Zhang, and G. Katz. Neural Network Verification with Proof Production. In *Proc. 22nd Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 38–48, 2022.
36. Y. Izza, A. Ignatiev, and J. Marques-Silva. On Explaining Decision Trees, 2020. Technical Report. http://arxiv.org/abs/2010.11034.
37. Y. Izza, A. Ignatiev, N. Narodytska, M. Cooper, and J. Marques-Silva. Efficient Explanations with Relevant Sets, 2021. Technical Report. http://arxiv.org/abs/2106.00546.
38. K. Julian, M. Kochenderfer, and M. Owen. Deep Neural Network Compression for Aircraft Collision Avoidance Systems. *Journal of Guidance, Control, and Dynamics*, 42(3):598–608, 2019.
39. G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Proc. 29th Int. Conf. on Computer Aided Verification (CAV)*, pages 97–117, 2017.
40. G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: a Calculus for Reasoning about Deep Neural Networks. *Formal Methods in System Design (FMSD)*, 2021.
41. G. Katz, D. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, D. Dill, M. Kochenderfer, and C. Barrett. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In *Proc. 31st Int. Conf. on Computer Aided Verification (CAV)*, pages 443–452, 2019.
42. E. La Malfa, A. Zbrzezny, R. Michelmore, N. Paoletti, and M. Kwiatkowska. On Guaranteed Optimal Robust Explanations for NLP Models, 2021. Technical Report. https://arxiv.org/abs/2105.03640.
43. O. Lahav and G. Katz. Pruning and Slicing Neural Networks using Formal Verification. In *Proc. 21st Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 183–192, 2021.
44. H. Lakkaraju and O. Bastani. "How do I Fool You?" Manipulating User Trust via Misleading Black Box Explanations. In *Proc. AAAI/ACM Conf. on AI, Ethics, and Society (AIES)*, pages 79–85, 2020.
45. Y. LeCun. The MNIST Database of Handwritten Digits, 1998. http://yann.lecun.com/exdb/mnist/.
46. Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521(7553):436–444, 2015.
47. C. Li and F. Manya. MaxSAT, Hard and Soft Constraints. In *Handbook of Satisfiability*, pages 903–927. IOS Press, 2021.
48. P. Liberatore. Redundancy in logic i: Cnf propositional formulae. *Artificial Intelligence*, 163(2):203–232, 2005.
49. C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. Kochenderfer. Algorithms for Verifying Deep Neural Networks, 2020. Technical Report. http://arxiv.org/abs/1903.06758.
50. S. M. Lundberg and S.-I. Lee. A Unified Approach to Interpreting Model Predictions. In *Proc. 31st Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.
51. J. Marques-Silva and A. Ignatiev. Delivering Trustworthy AI through formal XAI. In *Proc. 36th AAAI Conf. on Artificial Intelligence (AAAI)*, pages 3806–3814, 2022.
52. M. Ostrovsky, C. Barrett, and G. Katz. An Abstraction-Refinement Approach to Verifying Convolutional Neural Networks. In *Proc. 20th. Int. Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2022.

53. I. Refaeli and G. Katz. Minimal Multi-Layer Modifications of Deep Neural Networks. In *Proc. 5th Workshop on Formal Methods for ML-Enabled Autonomous Systems (FoMLAS)*, 2022.
54. R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987.
55. M. Ribeiro, S. Singh, and C. Guestrin. "Why should I Trust You?" Explaining the Predictions of any Classifier. In *Proc. 22nd Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 1135–1144, 2016.
56. M. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-Precision Model-Agnostic Explanations. In *Proc. 32nd AAAI Conf. on Artificial Intelligence (AAAI)*, 2018.
57. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-Cam: Visual Explanations from Deep Networks via Gradient-Based Localization. In *Proc. 20th IEEE Int. Conf. on Computer Vision (ICCV)*, pages 618–626, 2017.
58. W. Shi, A. Shih, A. Darwiche, and A. Choi. On Tractable Representations of Binary Neural Networks, 2020. Technical Report. http://arxiv.org/abs/2004.02082.
59. A. Shih, A. Choi, and A. Darwiche. A Symbolic Approach to Explaining Bayesian Network Classifiers, 2018. Technical Report. http://arxiv.org/abs/1805.03364.
60. G. Singh, T. Gehr, M. Puschel, and M. Vechev. An Abstract Domain for Certifying Neural Networks. In *Proc. 46th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*, 2019.
61. D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: Removing Noise by Adding Noise, 2017. Technical Report. http://arxiv.org/abs/1706.03825.
62. C. Strong, H. Wu, A. Zeljić, K. Julian, G. Katz, C. Barrett, and M. Kochenderfer. Global Optimization of Objective Functions Represented by ReLU Networks. *Journal of Machine Learning*, pages 1–28, 2021.
63. D. Stutz, A. Hermans, and B. Leibe. Superpixels: An Evaluation of the State-of-the-Art. *Computer Vision and Image Understanding*, 166:1–27, 2018.
64. J. Su, D. Vargas, and K. Sakurai. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.
65. A. Vedaldi and S. Soatto. Quick Shift and Kernel Methods for Mode Seeking. In *Proc. 10th European Conf. on Computer Vision (ECCV)*, pages 705–718, 2008.
66. S. Waeldchen, J. Macdonald, S. Hauch, and G. Kutyniok. The Computational Complexity of Understanding Binary Classifier Decisions. *Journal of Artificial Intelligence Research*, 70:351–387, 2021.
67. S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Formal Security Analysis of Neural Networks using Symbolic Intervals. In *Proc. 27th USENIX Security Symposium*, 2018.
68. H. Wu, A. Ozdemir, A. Zeljić, A. Irfan, K. Julian, D. Gopinath, S. Fouladi, G. Katz, C. Păsăreanu, and C. Barrett. Parallelization Techniques for Verifying Neural Networks. In *Proc. 20th Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 128–137, 2020.
69. H. Wu, A. Zeljić, G. Katz, and C. Barrett. Efficient Neural Network Analysis with Sum-of-Infeasibilities. In *Proc. 28th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 143–163, 2022.
70. M. Wu, H. Wu, and C. Barrett. VeriX: Towards Verified Explainability of Deep Neural Networks, 2022. Technical Report. https://arxiv.org/abs/2212.01051.
71. H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNist: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017. Technical Report. http://arxiv.org/abs/1708.07747.

72. T. Zelazny, H. Wu, C. Barrett, and G. Katz. On Reducing Over-Approximation Errors for Neural Network Verification. In *Proc. 22nd Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 17–26, 2022.
73. Z. Zhou and L. Sun. Metamorphic Testing of Driverless Cars. *Communications of the ACM*, 62(3):61–67, 2019.