Efficiently Computing Compact Formal Explanations

Min Wu[1], **Xiaofu Li**[1], **Haoze Wu**[2 3], **Clark Barrett**[1]
[1]Stanford University
[2]Amherst College
[3]VMware Research
{minwu, lixiaofu, barrettc}@stanford.edu, hwu@amherst.edu

Abstract

Building on VERIX (VERIfied eXplainability), a system for producing *optimal verified explanations* for machine learning models, we present VERIX+, which significantly improves both the *size* and the generation *time* of formal explanations. We introduce a bound propagation-based sensitivity technique to improve the size, and a binary search-based traversal with confidence ranking for improving time—the two techniques are orthogonal and can be used independently or together. We also show how to adapt the QuickXplain algorithm to our setting to provide a trade-off between size and time. Experimental evaluations on standard benchmarks demonstrate significant improvements on both metrics, e.g., a size reduction of 38% on the GTSRB dataset and a time reduction of 90% on MNIST. We demonstrate that our approach is scalable to transformers and real-world scenarios such as autonomous aircraft taxiing and sentiment analysis. We conclude by showcasing several novel applications of formal explanations.

Code — github.com/NeuralNetworkVerification/VeriX+
Extended version — arxiv.org/abs/2409.03060

1 Introduction

Explainable AI aims to extract a set of reasoning steps from the decision-making processes of otherwise opaque AI systems, thus making them more understandable and trustworthy to humans. Well-known work on explainable AI includes model-agnostic explainers, such as LIME (Ribeiro, Singh, and Guestrin 2016), SHAP (Lundberg and Lee 2017), and Anchors (Ribeiro, Singh, and Guestrin 2018), which provide explanations for neural networks by constructing a local model around a given input or identifying a subset of input features as "anchors" that (ideally) ensure a model's decision. While such methods can produce explanations efficiently, they do not provide *formal* guarantees and thus may be inadequate in high-stakes scenarios, e.g., when transparency or fairness are paramount.

Formal explainable AI (Marques-Silva and Ignatiev 2022) aims to compute explanations that *verifiably* ensure the invariance of a model's decision. One such explanation is a minimal set of input features with the property that regardless of how the *remaining* features are perturbed, the prediction remains unchanged. The intuition is that these features capture an amount of (explicit or implicit) information in the input that is sufficient to preserve the current decision. The simplest approaches allow *unbounded* perturbations (Ignatiev, Narodytska, and Marques-Silva 2019; Shih, Choi, and Darwiche 2018; Darwiche and Hirth 2020), which may be overly lenient in some cases, potentially leading to explanations that are too course-grained to be useful. As an alternative, (La Malfa et al. 2021) and (Wu, Wu, and Barrett 2023) generalize these approaches by allowing both *bounded* and unbounded perturbations, computing explanations with respect to such perturbations for natural language processing and perception models, respectively. The former primarily perturbs each word in a text with a finite set of its k closest neighbors and thus has a discrete perturbation space; the latter considers ϵ-ball perturbations over continuous and dense input spaces. The algorithm presented in (Wu, Wu, and Barrett 2023) uses a well-known but naive approach: it simply traverses the features one by one and checks formally whether any of them can be discarded. This approach sometimes yields overly conservative explanations that are inefficient to compute.

In this paper, we explore methods for computing better explanations by significantly improving both the *size* and the generation *time*. We also demonstrate novel applications that illustrate the usefulness of such explanations in practice. Our contributions can be summarized as follows.

- We utilize *bound propagation*-based techniques to obtain more fine-grained feature-level sensitivity information, leading to better traversal orders, which in turn produce smaller explanation sizes.
- We propose a *binary search*-inspired traversal approach to enable processing features in a batch manner, thus significantly reducing the *time* required to generate explanations. We also incorporate a simple but efficient *confidence ranking* strategy to further reduce time.
- We adapt the QuickXplain algorithm (Junker 2004) to provide a *trade-off* between explanation *size* and generation *time*, and note that our adaptation is an optimization of (Huang and Marques-Silva 2023).
- We demonstrate how explanations can be used in several applications, including analyzing the effects of adversarial training, as well as detecting out-of-distribution and misclassified samples.

2 VERIX+: Verified eXplainability plus

Let f be a neural network and $\mathbf{x}$ an input consisting of m-dimensional features $\langle x_1, \ldots, x_m \rangle$. The set of feature indices $\{1, \ldots, m\}$ is written as $\Theta(\mathbf{x})$, or simply Θ, when the context is clear. To denote a subset of indices, we use $\mathbf{A} \subseteq \Theta(\mathbf{x})$, and $\mathbf{x_A}$ denotes the features that are indexed by the indices in $\mathbf{A}$. We write $f(\mathbf{x}) = c$ for both regression models (c is a single quantity) and classification models ($c \in C$ is one of a set of possible labels). For the latter case, we use $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$ to denote confidence values for each label in C, e.g., the predicted class $c = \arg\max(\mathbf{y})$. For image classification tasks, $\mathbf{x}$ is an image of m pixels, the values in $\mathbf{y}$ represent the confidence values for each of the n labels in C, and y_c is the maximum value in $\mathbf{y}$, where c is the predicted label.

2.1 Optimal Verified Explanations

We adopt the definition of *optimal robust explanations* from (La Malfa et al. 2021; Wu, Wu, and Barrett 2023) (see also (Shih, Choi, and Darwiche 2018; Ignatiev, Narodytska, and Marques-Silva 2019; Darwiche and Hirth 2020)). For a network f and an input $\mathbf{x}$, we compute a minimal subset of $\mathbf{x}$, denoted by $\mathbf{x_A}$, such that any ϵ-perturbations imposed on the remaining features do not change the model's prediction.

Definition 2.1 (Optimal Verified Explanation (Wu, Wu, and Barrett 2023))**.** Given a neural network f, an input $\mathbf{x}$, a manipulation magnitude ϵ, and a discrepancy δ, a *verified explanation* with respect to norm $p \in \{1, 2, \infty\}$ is a set of input features $\mathbf{x_A}$ such that if $\mathbf{B} = \Theta(\mathbf{x}) \setminus \mathbf{A}$, then

$$\forall\, \mathbf{x'_B}.\ \|\mathbf{x_B} - \mathbf{x'_B}\|_p \leq \epsilon \implies |f(\mathbf{x}) - f(\mathbf{x'})| \leq \delta, \quad (1)$$

where $\mathbf{x'_B}$ is some perturbation on the *irrelevant* features $\mathbf{x_B}$ and $\mathbf{x'}$ is the input variant combining $\mathbf{x_A}$ and $\mathbf{x'_B}$. We say that the verified explanation $\mathbf{x_A}$ is *optimal* if

$$\forall\, x \in \mathbf{x_A}.\ \exists\, x', \mathbf{x'_B}.\ \|(x \cup \mathbf{x_B}) - (x' \cup \mathbf{x'_B})\|_p \leq \epsilon \wedge |f(\mathbf{x}) - f(\mathbf{x'})| > \delta, \quad (2)$$

where x' is some perturbation of $x \in \mathbf{x_A}$ and $\cup$ denotes concatenation of features.

Such explanations are both *sound* and *optimal* by Equations (1) and (2), respectively (Wu, Wu, and Barrett 2023). Note that the optimality we define here is *local* as it computes a minimal (not minimum) subset. The minimum subset is called a *global* optimum, also known as the cardinality-minimal explanation (Ignatiev, Narodytska, and Marques-Silva 2019). Finding the global optimum is often too computationally expensive to be practically useful, as it requires searching over an exponential number of local optima.

2.2 Workflow of VERIX+ in A Nutshell

We present the overall workflow of our VERIX+ framework in Figure 1. Starting from the left, the inputs are a network f and an input $\mathbf{x}$. The first step is to obtain a sensitivity map of all the input features and, by ranking their individual sensitivity, produce a traversal order. We introduce a new bound propagation-based technique (Algorithm 1) for obtaining more meaningful sensitivity maps and thus better traversal orders, which, in turn, reduce explanation sizes.

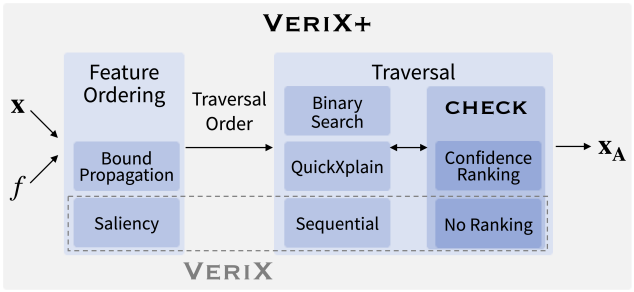


Figure 1: The VERIX+ framework, where $\mathbf{x}$ is the input, f the neural network, and $\mathbf{x_A}$ the optimal verified explanation.

The traversal order is then passed to the main traversal algorithm, which computes optimal verified explanations. We propose two new optimizations, one based on binary search (Algorithm 2) and one adapted from the well-known QuickXplain algorithm (Junker 2004) (Algorithm 4). Comparing to the existing sequential method, which precesses features one at a time, the two new algorithms significantly reduce computation time by processing features in *batches*. The key difference is that Algorithm 2 reduces computation time without affecting the explanation size, as it retains the original traversal order. In contrast, Algorithm 4 not only reduces computation time but also reduces the explanation size by adaptively adjusting the traversal order on the fly. This adaptive strategy introduces some overhead compared to Algorithm 2, but it remains significantly faster than the sequential method, resulting in a trade-off between explanation size and computation time.

The CHECK procedure (Algorithm 3) is used by the traversal methods to formally check the soundness of a candidate explanation. We also add a simple but efficient confidence ranking algorithm which further reduces generation time. The confidence ranking is orthogonal to the other optimizations and benefits all three traversal approaches.

3 Methodological Advances for Explanation Size and Generation Time

In this section, we discuss in detail several optimizations for improving both the explanation *size* and the generation *time*. The bound propagation-based sensitivity ranking discussed in Section 3.1 improves size, and the binary search-based traversal discussed in Section 3.2 improves time, as does the confidence ranking discussed in Section 3.3. Section 3.4 discusses how an adaptation of the QuickXplain algorithm (Junker 2004) enables an additional size-time trade-off.

3.1 Improving size: A Bound Propagation-based Traversal Order

In (Wu, Wu, and Barrett 2023), a traversal order is computed based on a heuristic that measures how sensitive a model's confidence is to each individual feature x_i by imposing simple feature *deletion* or *reversal*. For example, for images, after normalizing pixel values from $[0, 255]$ to $[0, 1]$, deletion of feature i sets $x_i = 0$ and reversal sets x_i to $1 - x_i$. Although

Algorithm 1: TRAVERSALORDER

Input: neural network f and input $\mathbf{x}$
Parameter : ϵ-perturbation
Output: traversal order π

1 **function** TRAVERSALORDER($f, \mathbf{x}$)
2 $\quad c \mapsto f(\mathbf{x})$
3 $\quad \hat{\mathbf{x}} \mapsto newVars()$
4 $\quad$ **for** $x_i \in \mathbf{x}$ **do**
5 $\quad\quad \hat{x}_i \mapsto [x_i - \epsilon, x_i + \epsilon]$
6 $\quad\quad \hat{x}_j \mapsto [x_j, x_j]$ where $j \neq i$
7 $\quad\quad$ lower $\mapsto$ COMPUTEBOUND($f, \hat{\mathbf{x}}$)
8 $\quad\quad \epsilon$-bounds[i] $\mapsto$ lower$_c$
9 $\quad \pi \mapsto \arg\text{sort}(\epsilon\text{-bounds}, \text{descending})$
10 $\quad$ **return** π

this produces a reasonable ranking of the input indices, it is not ideal for explanations that are based on ϵ-perturbations. We show how utilizing the perturbation information at the sensitivity phase can produce better traversal orders.

Procedure TRAVERSALORDER in Algorithm 1 computes *ϵ-bounds* for each feature of the input and then ranks these bounds to obtain an order on feature indices. We introduce variables $\hat{\mathbf{x}}$ which represent symbolic inputs to f (Line 3). Then, for[1] each individual feature x_i in $\mathbf{x}$, we set a constraint on its corresponding variable $\hat{x}_i$, requiring it to be in the *interval* $[x_i - \epsilon, x_i + \epsilon]$ (Line 5). Each of the remaining feature variables $\hat{x}_j$, with $j \neq i$ and $j \in \{1, \ldots, m\}$, is constrained to be equal to its corresponding input x_j (Line 6).[2] In Line 7, we pass $\hat{\mathbf{x}}$ (with only its i-th feature $\hat{x}_i$ allowed to change) and the model f to a bounds-analysis procedure, which computes lower and upper bounds on the outputs of $f(\hat{\mathbf{x}})$.

Note that, since f could be highly nonlinear, computing these bounds is not straightforward. That is why, instead of performing simple model inferences, we utilize existing analyses such as IBP (interval bounded propagation) (Gowal et al. 2019) and CROWN (Zhang et al. 2018) to compute the output bounds. We found (empirically) that retaining only the lower bound of the predicted class lower$_c$ provides the most effective ranking. We store the lower bound in Line 8. Once we have all the bounds, we sort them in a descending order (Line 9) – as input features producing higher lower bounds are more likely to be irrelevant to the explanation. Intuitively, the higher the lower bound, the less the output is affected by perturbing the input. Traversing less relevant features first leads to smaller explanations.

3.2 Improving Time: A Binary Search-based Traversal

Given a traversal order, the algorithm of (Wu, Wu, and Barrett 2023) simply processes the features one by one in a sequential order. Here, we propose an improvement that processes the features using an alternative approach inspired by binary search. The new algorithm searches for *batches* of *consecutive* irrelevant features. It simultaneously checks the whole batch to see whether it is irrelevant. If so, there is no need to process the features in the batch one by one. We note that this approach does not change the traversal order, so the computed explanation is the same as that computed by the original sequential algorithm.

Algorithm 2: BINARYSEQUENTIAL

Input: neural network f and input $\mathbf{x}$
Parameter : ϵ-perturbation, norm p
Output: explanation $\mathbf{x_A}$ and irrelevant set $\mathbf{x_B}$

1 $\mathbf{x_A}, \mathbf{x_B} \mapsto \emptyset, \emptyset$
2 $\mathbf{x_A}, \mathbf{x_B} \mapsto$ BINARYSEQUENTIAL($f, \mathbf{x}$)
3 **function** BINARYSEQUENTIAL($f, \mathbf{x_\Theta}$)
4 $\quad$ **if** $|\mathbf{x_\Theta}| = 1$ **then**
5 $\quad\quad$ **if** CHECK($f, \mathbf{x}, \mathbf{x_B} \cup \mathbf{x_\Theta}$) **then**
6 $\quad\quad\quad \mathbf{x_B} \mapsto \mathbf{x_B} \cup \mathbf{x_\Theta}$
7 $\quad\quad\quad$ **return**
8 $\quad\quad$ **else**
9 $\quad\quad\quad \mathbf{x_A} \mapsto \mathbf{x_A} \cup \mathbf{x_\Theta}$
10 $\quad\quad\quad$ **return**
11 $\quad \mathbf{x_\Phi}, \mathbf{x_\Psi} = \text{split}(\mathbf{x_\Theta}, 2)$
12 $\quad$ **if** CHECK($f, \mathbf{x}, \mathbf{x_B} \cup \mathbf{x_\Phi}$) **then**
13 $\quad\quad \mathbf{x_B} \mapsto \mathbf{x_B} \cup \mathbf{x_\Phi}$
14 $\quad\quad$ **if** CHECK($f, \mathbf{x}, \mathbf{x_B} \cup \mathbf{x_\Psi}$) **then**
15 $\quad\quad\quad \mathbf{x_B} \mapsto \mathbf{x_B} \cup \mathbf{x_\Psi}$
16 $\quad\quad$ **else**
17 $\quad\quad\quad$ **if** $|\mathbf{x_\Psi}| = 1$ **then**
18 $\quad\quad\quad\quad \mathbf{x_A} \mapsto \mathbf{x_A} \cup \mathbf{x_\Psi}$
19 $\quad\quad\quad$ **else**
20 $\quad\quad\quad\quad$ BINARYSEQUENTIAL($f, \mathbf{x_\Psi}$)
21 $\quad$ **else**
22 $\quad\quad$ **if** $|\mathbf{x_\Phi}| = 1$ **then**
23 $\quad\quad\quad \mathbf{x_A} \mapsto \mathbf{x_A} \cup \mathbf{x_\Phi}$
24 $\quad\quad$ **else**
25 $\quad\quad\quad$ BINARYSEQUENTIAL($f, \mathbf{x_\Phi}$)
26 $\quad\quad$ BINARYSEQUENTIAL($f, \mathbf{x_\Psi}$)

Algorithm 2 globally updates the explanation set $\mathbf{x_A}$ and the irrelevant set $\mathbf{x_B}$ throughout. Initially, these two sets are initialized to $\emptyset$. After executing BINARYSEQUENTIAL($f, \mathbf{x}$), the input $\mathbf{x}$ is partitioned into disjoint $\mathbf{x_A}$ and $\mathbf{x_B}$, i.e., $\mathbf{x_A} \cup \mathbf{x_B} = \mathbf{x}$. In function BINARYSEQUENTIAL($f, \mathbf{x_\Theta}$), where $\mathbf{x_\Theta}$ are candidate features, the first **if** condition (Lines 4-10) considers the base case when there is only a single feature left in $\mathbf{x_\Theta}$. If CHECK($f, \mathbf{x}, \mathbf{x_B} \cup \mathbf{x_\Theta}$) returns `True` (Line 5), i.e., perturbing the current $\mathbf{x_B}$ and $\mathbf{x_\Theta}$ does not change model's decision (same c for classification or $|c - c'| \leq \delta$ for regression), then $\mathbf{x_\Theta}$ is put into $\mathbf{x_B}$ (Line 6). Otherwise, it is added to $\mathbf{x_A}$ (Line 9). In the non-base case, $\mathbf{x_\Theta}$ has more than just one feature. For this case, we split $\mathbf{x_\Theta}$ into two sets $\mathbf{x_\Phi}$ and $\mathbf{x_\Psi}$ with similar sizes (Line 11). If CHECK($f, \mathbf{x}, \mathbf{x_B} \cup \mathbf{x_\Phi}$) returns `True` (Line 12-20), then $\mathbf{x_\Phi}$ belongs to the irrelevant set $\mathbf{x_B}$ (Line 13) and the procedure

[1]The for-loop is only used to present the functionality in a clear way; in our implementation, the ϵ-bounds for all the input features are computed in *parallel* for time efficiency.

[2]As a further optimization (not shown), if the input data are known to be bounded as part of the problem definition, we intersect the interval $[x_i - \epsilon, x_i + \epsilon]$ with the known bound.

continues by checking if $\mathbf{x}_\Psi$ is irrelevant (Line 14): if `True`, $\mathbf{x}_\Psi$ is also added to $\mathbf{x_B}$ (Line 15). Otherwise, if $\mathbf{x}_\Psi$ contains only one feature (Line 17), we know it must be part of the explanation feature set and directly add it to $\mathbf{x_A}$ (Line 18). (Note that this check avoids unnecessary execution of the very first **if** condition (Lines 4-10).) If not, we recursively call BINARYSEQUENTIAL$(f, \mathbf{x}_\Psi)$ to search for batches of consecutive irrelevant features in $\mathbf{x}_\Psi$ (Line 20). Finally, if CHECK$(f, \mathbf{x}, \mathbf{x_B} \cup \mathbf{x}_\Phi)$ (Line 12) returns `False` (Lines 21-26), we similarly process $\mathbf{x}_\Phi$. And when $\mathbf{x}_\Phi$ is done, we call BINARYSEQUENTIAL$(f, \mathbf{x}_\Psi)$ to process $\mathbf{x}_\Psi$ (Line 26).

Theorem 3.1 (Time Complexity)**.** *Given a neural network f and an input $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$ where $m \geq 2$, the* time complexity *of* BINARYSEQUENTIAL$(f, \mathbf{x})$ *is* 2 *calls of* CHECK *for the* best *case (all features are irrelevant) and $k_{2m} = 2 \cdot k_m + 1$ or $k_{2m+1} = k_{m+1} + k_m + 1$, for which $k_2 = 2$ and $k_3 = 4$ are the base cases, calls of* CHECK *for the* worst *case (all features are explanatory). When $m = 1$, it is obvious that only one* CHECK *call is needed. The proof is given in Appendix A.1 of (Wu et al. 2025).*

Despite the fact that the worst-case complexity is worse than that of the naive algorithm (which always requires m calls to CHECK), we show in Section 4 that BINARYSEQUENTIAL performs better in practice, and remark that smaller explanations lead to more prominent advantage of our algorithm.

3.3 Improving Time: A Confidence Ranking

The CHECK procedure checks whether a model's decision is invariant under ϵ-perturbations of a specified subset of input features. That is, it must check whether the output c is in an allowable range for regression, i.e., $|c - c'| \leq \delta$, or whether the confidence of the predicted class y_c is always the greatest among all classes, i.e., $y_c = \max(\mathbf{y})$. In the latter case, this means that in the worst case, $|\mathbf{y}| - 1$ separate checks are needed to ensure that y_c is the largest. In previous work (Wu, Wu, and Barrett 2023), these checks are done naively without any insight into what order should be used. In this work, we propose *ranking* these checks based on the confidence values of the corresponding classes. We then proceed to do the checks from the *most* to the *least* likely classes. If a check fails, i.e., decision invariance is not guaranteed, this is typically because we can perturb the inputs to produce one of the next most likely classes. Thus, by checking the most likely classes first, we avoid the need for additional checks if one of those checks already fails.

Algorithm 3 shows the CHECK procedure which checks whether imposing ϵ-perturbations on certain features $\mathbf{x}_\Theta$ of input $\mathbf{x}$ maintains the decision of model f: if yes, it returns `True`, meaning that these features are irrelevant. To start, we create variables $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ to represent the inputs and outputs of the model and set $\mathbf{y}$ to be the logits and c to be the predicted class. In Line 4, we rank the confidence values of all the classes in a descending order, prioritizing classes that are most likely. We allow ϵ-perturbation on features in $\mathbf{x}_\Theta$ while fixing the others (Lines 6–9). For each class j in the sorted ranking (excluding c as this is the predicted class to be compared with), we call SOLVE to examine whether the specification $\phi \Rightarrow \hat{y}_c < \hat{y}_j$ holds, i.e., whether the input constraints ϕ allow a prediction change with $\hat{y}_c$ smaller than $\hat{y}_j$ (Line 11). If the exitCode of SOLVE is `UNSAT`, then it means the specification is unsatisfiable in the sense that $\hat{y}_c$ will always be greater than or equal to $\hat{y}_j$, i.e., the prediction cannot be manipulated into class j. The **for** loop (Lines 10–15) examines each class in ranking $\setminus\, c$, and if all checks are `UNSAT` (algorithm returns `True` in Line 16), then $\hat{y}_c$ is ensured to be the greatest among $\hat{\mathbf{y}}$, i.e., prediction invariance is guaranteed. Otherwise, if SOLVE returns `SAT` or `Unknown` for any $\hat{y}_j$, the algorithm returns `False`. The key insight is that SOLVE is more likely to return `SAT` for classes with higher confidence, and once it does, the algorithm terminates. In practice, SOLVE can be instantiated with off-the-shelf neural network verification tools (Singh et al. 2019; Müller et al. 2022; Katz et al. 2019; Wang et al. 2021; Henriksen and Lomuscio 2020; Wu et al. 2024; Xu et al. 2020; Gehr et al. 2018; Katz et al. 2017).

Algorithm 3: CHECK with confidence ranking

Input: neural network f, input $\mathbf{x}$, and candidate features $\mathbf{x}_\Theta$
Parameter: ϵ-perturbation
Output: `True`/`False`

```
1  function CHECK(f, x, x_Θ)
2      x̂, ŷ ↦ newVars()
3      c, y ↦ f(x)
4      ranking ↦ arg sort(y, descending)
5      φ ↦ True
6      for i ∈ Θ do
7          φ ↦ φ ∧ (‖x̂_i − x_i‖_p ≤ ε)
8      for i ∈ Θ(x) \ Θ do
9          φ ↦ φ ∧ (x̂_i = x_i)
10     for j ∈ ranking \ c in order do
11         exitCode ↦ SOLVE(f, φ ⇒ ŷ_c < ŷ_j)
12         if exitCode == UNSAT then
13             continue
14         else
15             break
16     return exitCode == UNSAT
```

3.4 A Size-Time *Trade-off*: QuickXplain

In previous sections, we propose approaches to orthogonally improve explanation size and generation time; in this section, we adapt the QuickXplain algorithm (Junker 2004) and optimize it (Huang and Marques-Silva 2023) to provide an additional *trade-off* between these two metrics. We remark that the QuickXplain-based approach works as an alternative to the binary search-based method, i.e., given a traversal order from the first step of our workflow, we can either use binary search or QuickXplain. The former improves time but does not affect size, whereas the latter affects both. In practice, QuickXplain tends to produce smaller explanations at the cost of requiring more time. Confidence ranking benefits both techniques, as it accelerates CHECK.

We present our QUICKXPLAIN adaptation in Algorithm 4. The function QXP$(\mathbf{x}_\alpha, \mathbf{x}_\beta, \mathbf{x}_\Theta)$ itself is recursive with three arguments: (1) the current explanation $\mathbf{x}_\alpha$; (2) the current

Algorithm 4: QUICKXPLAIN

Input: neural network f and input $\mathbf{x}$
Parameter: ϵ-perturbation, norm p
Output: explanation $\mathbf{x_A}$ and irrelevant set $\mathbf{x_B}$

1 $\mathbf{x_A}, \mathbf{x_B} \mapsto \text{QXP}(\emptyset, \emptyset, \mathbf{x})$
2 **function** $\text{QXP}(\mathbf{x}_\alpha, \mathbf{x}_\beta, \mathbf{x}_\Theta)$
3 **if** $|\mathbf{x}_\Theta| = 1$ **then**
4 **if** $\text{CHECK}(f, \mathbf{x}, \mathbf{x}_\beta \cup \mathbf{x}_\Theta)$ **then**
5 **return** $\emptyset, \mathbf{x}_\beta \cup \mathbf{x}_\Theta$
6 **else**
7 **return** $\mathbf{x}_\Theta, \mathbf{x}_\beta$
8 $\mathbf{x}_\Phi, \mathbf{x}_\Psi = \text{split}(\mathbf{x}_\Theta, 2)$
9 **if** $\text{CHECK}(f, \mathbf{x}, \mathbf{x}_\beta \cup \mathbf{x}_\Phi)$ **then**
10 **return** $\text{QXP}(\mathbf{x}_\alpha, \mathbf{x}_\beta \cup \mathbf{x}_\Phi, \mathbf{x}_\Psi)$
11 **else if** $\text{CHECK}(f, \mathbf{x}, \mathbf{x}_\beta \cup \mathbf{x}_\Psi)$ **then**
12 **return** $\text{QXP}(\mathbf{x}_\alpha, \mathbf{x}_\beta \cup \mathbf{x}_\Psi, \mathbf{x}_\Phi)$
13 **else**
14 **if** $|\mathbf{x}_\Phi| = 1$ **then**
15 $\mathbf{x}'_\Phi, \mathbf{x}'_\beta \mapsto \mathbf{x}_\Phi, \mathbf{x}_\beta$
16 **else**
17 $\mathbf{x}'_\Phi, \mathbf{x}'_\beta \mapsto \text{QXP}(\mathbf{x}_\alpha \cup \mathbf{x}_\Psi, \mathbf{x}_\beta, \mathbf{x}_\Phi)$
18 **if** $|\mathbf{x}_\Psi| = 1$ **then**
19 $\mathbf{x}'_\Psi, \mathbf{x}''_\beta \mapsto \mathbf{x}_\Psi, \mathbf{x}'_\beta$
20 **else**
21 $\mathbf{x}'_\Psi, \mathbf{x}''_\beta \mapsto \text{QXP}(\mathbf{x}_\alpha \cup \mathbf{x}'_\Phi, \mathbf{x}'_\beta, \mathbf{x}_\Psi)$
22 **return** $\mathbf{x}'_\Phi \cup \mathbf{x}'_\Psi, \mathbf{x}''_\beta$

irrelevant set $\mathbf{x}_\beta$; and (3) the current (sub)set of input features that need to be analyzed, $\mathbf{x}_\Theta$. These three sets always form a partition of the full set of features. To start with, $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$ are initialized to $\emptyset$, and when QXP proceeds, irrelevant features are added into $\mathbf{x}_\beta$ in a monotonically increasing way; finally, after all features are done, $\mathbf{x}_\alpha$ is returned as the optimal explanation $\mathbf{x_A}$ and $\mathbf{x}_\beta$ as the irrelevant set $\mathbf{x_B}$. Now we walk through the algorithm. Lines 3–7 cover the base case when $\mathbf{x}_\Theta$ has a single feature as in Algorithm 2. When there is more than one feature in $\mathbf{x}_\Theta$, it is split into two subsets $\mathbf{x}_\Phi$ and $\mathbf{x}_\Psi$ (Line 8). In Lines 9–12, we check if the subset $\mathbf{x}_\Phi$ or $\mathbf{x}_\Psi$ belongs to the irrelevant set: if `True`, then we add it into $\mathbf{x}_\beta$ when calling QXP to process the other subset. If neither $\mathbf{x}_\Phi$ nor $\mathbf{x}_\Psi$ is irrelevant, we take turns processing $\mathbf{x}_\Phi$ and $\mathbf{x}_\Psi$ in Lines 13–22: the first **if** condition analyzes $\mathbf{x}_\Phi$, and the second **if** processes $\mathbf{x}_\Psi$. If either of them has only a single feature, then we know it must be included as an explanation feature ($\mathbf{x}'_\Phi$ on Line 15 or $\mathbf{x}'_\Psi$ on Line 19). This avoids the unnecessary execution of Lines 3–7 in a recursive call since we already know the feature cannot be irrelevant.[3] If either of them has more than one feature, then QXP is called recursively: $\mathbf{x}_\Phi$ is processed in Line 17, with $\mathbf{x}_\Psi$ as part of the new $\mathbf{x}_\alpha$, and $\mathbf{x}_\Psi$ is processed in Line 21 with $\mathbf{x}'_\Phi$ as part of the new $\mathbf{x}_\alpha$. Finally, both $\mathbf{x}'_\Phi$ and $\mathbf{x}'_\Psi$ are returned as explanatory features, and $\mathbf{x}''_\beta$ as the irrelevant set (Line 22).

[3]This enables Algorithm 4 to have *fewer* calls to CHECK (i.e., oracle calls) than the similar adaptation of QuickXplain in (Huang and Marques-Silva 2023).

(a) MNIST (b) GTSRB (c) CIFAR10

Figure 2: Explanations (pixels highlighted in green) from VERIX (left) and VERIX+ (right). (a) MNIST: size 187 vs. 137; time 635 vs. 51. (b) GTSRB: size 276 vs. 117; time 703 vs. 49. (c) CIFAR10: size 208 vs. 146; time 482 vs. 57. Statistical significance for size and time reduction is in Table 4 of (Wu et al. 2025).

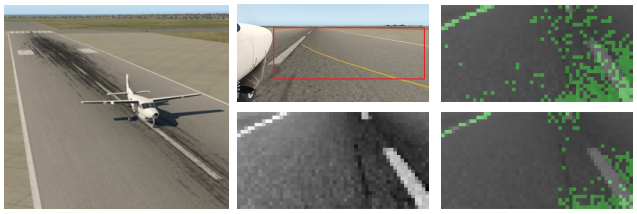


Figure 3: A real-world aircraft taxiing scenario (Julian, Lee, and Kochenderfer 2020). Pictures are taken from the camera fixed on the right wing of the aircraft. Explanation (green) from the transformer model (bottom right) is more compact than its fully-connected counterpart (top right).

Theorem 3.2 (Time Complexity)**.** *Given a neural network f and an input $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$ where $m \geq 2$, the* time complexity *of* $\text{QXP}(\emptyset, \emptyset, \mathbf{x})$ *is* $\lfloor \log_2 m \rfloor + 1$ *calls of* CHECK *for the* best *case (all features are irrelevant) and* $(m-1) \times 2$ *calls of* CHECK *for the* worst *case (all features are explanatory). When $m = 1$, it is obvious that only one* CHECK *call is needed. The proof is given in Appendix A.2 of (Wu et al. 2025).*

4 Experimental Results

We have implemented the VERIX+ framework in Python. To realize the COMPUTEBOUND analysis in Algorithm 1, we utilize the bound-propagation (Mathiesen 2022) package for fully-connected models and the auto_LiRPA (Xu et al. 2020) library for convolutional models. While the latter also supports dense layers, the former computes tighter IBP bounds which, in our case, lead to smaller explanations. Therefore, we delegate different types of models to them separately. We use Marabou (Wu et al. 2024), a neural network verification tool, to perform the SOLVE function in Algorithm 3. Our framework can accommodate other existing tools as long as they can perform the COMPUTEBOUND or the SOLVE functionality. We trained fully-connected and convolutional networks on standard image benchmarks including MNIST (LeCun, Cortes, and Burges 2010), GTSRB (Stallkamp et al. 2012), and CIFAR10 (Krizhevsky et al. 2009), as well as transformers (Vaswani et al. 2017) for real-world applications. Details on the model structures are in Appendix C of (Wu et al. 2025). Experiments were performed on a workstation equipped with 16 AMD Ryzen™ 7 5700G CPUs and 32GB memory running Fedora 37.

Table 1: Average explanation *size* (number of pixels) and generation *time* (seconds) for both VERIX and VERIX+ on fully-connected (-FC), convolutional (-CNN), and transformer (-Transformer) models. ϵ is set to $5\%, 1\%$, and 1% for the MNIST, GTSRB, and CIFAR10 datasets, respectively. We also include transformer results on TaxiNet and IMDB with $\epsilon = 1\%$ and 0.05. For images, we use Chebyshev distance to mimic natural distortions; for texts, we use Euclidean distance on their embeddings.

approaches	VERIX		**VERIX+**											
traversal order	saliency		saliency						*bound propagation*					
traversal procedure	sequential		*conf ranking*		*binary search*		*QuickXplain*		*conf ranking*		*binary search*		*QuickXplain*	
metrics	size	time	size	time	size	time	size	time	size	time	size	time	size	time
MNIST-FC	186.7	92.3	186.7	81.4	186.7	29.6	186.3	32.1	179.5	70.6	179.5	**28.4**	**179.0**	32.2
MNIST-CNN	105.7	439.2	105.7	398.9	105.7	46.3	105.7	51.5	100.8	295.8	100.8	**42.0**	**100.6**	47.4
GTSRB-FC	529.4	614.9	529.4	488.0	529.4	233.8	485.0	286.6	333.6	437.7	333.6	**149.7**	**333.4**	196.1
GTSRB-CNN	569.0	1897.6	569.0	1312.7	569.0	394.0	506.0	466.4	355.8	1430.8	355.8	**251.0**	**355.3**	330.1
CIFAR10-FC	588.9	438.0	588.9	357.7	588.9	292.2	582.4	383.5	465.8	354.9	465.8	**238.8**	**465.7**	316.9
CIFAR10-CNN	664.8	1617.0	664.8	1033.5	664.8	448.2	652.9	567.8	553.5	1152.8	553.5	**371.1**	**552.8**	482.2
TaxiNet-Transformer	150.2	749.2	–	–	150.2	661.6	150.2	687.0	–	–	91.2	**377.5**	**91.0**	404.6
IMDB-Transformer	20.0	161.2	–	–	20.0	30.6	19.3	44.3	–	–	15.0	**37.3**	**15.0**	42.3

4.1 Improvements for Explanation *Size* and Generation *Time*

We report improvements in explanation *size* and generation *time* for image benchmarks in Table 1, along with example explanations in Figure 2, and statistically significant reductions in Table 4 of (Wu et al. 2025). For each data point, we collect "valid" explanations (i.e., excluding examples that are robust to ϵ-perturbation) for the first 100 test images (to avoid selection bias) and take the average time and size.

The overall observation is that, for the same traversal order (e.g., "saliency"), confidence ranking reduces *generation time* compared to the baseline sequential method, and the binary search-based traversal procedure further decreases the time substantially. For example, on the MNIST-CNN model, the explanation time is reduced from 439.2 to 398.9, and further to 46.3, achieving an overall speedup of approximately $10\times$. An example explanation for an MNIST image is shown in Figure 2a. For the digit "2", the pixels highlighted in green are explanatory, as modifying them—particularly those in the central region—from black to white can change the model's prediction to "8". As for *explanation size*, using the same traversal procedure (e.g., "binary search"), the bound propagation-based traversal order yields significantly smaller explanations compared to saliency-based ordering. For instance, on the GTSRB-CNN model, the average explanation size is reduced from 569.0 to 355.8, a reduction of approximately 38%. In Figure 2b of the traffic sign "30 mph", our explanation is more focused, as it is almost entirely contained in the central region containing "30". Finally, in nearly all cases, QuickXplain achieves a slight additional reduction in explanation size, but at the cost of increased computation time—representing a trade-off between size and time.

4.2 Explanations for Transformers in Real-world Applications

We show that our approach can also be applied to *transformer* models (Vaswani et al. 2017) and real-world applications by looking at two additional applications: a control system called TaxiNet (Julian, Lee, and Kochenderfer 2020) for autonomous aircraft taxiing and IMDB (Maas et al. 2011) movie review sentiment analysis. TaxiNet uses pictures from a camera fixed on the right wing of an aircraft to adjust steering controls to keep it on the taxiway. Figure 3 shows an example explanation. For IMDB reviews, the task is to analyze a review's sentiment (positive or negative). An example explanation would be that by fixing `intelligent` and `movies` in `[CLS] one of the more intelligent children's movies to hit theaters this year`, any perturbations on the embeddings of the other words will never flip the sentiment. As IMDB are real-world reviews, their length varies, so we padded all reviews out to 1000 tokens. In order to focus on harder problems, we report results on reviews longer than 500 tokens (before padding). To focus on meaningful explanations, we also exclude problems whose explanations are 50 tokens or more. Figure 6 of (Wu et al. 2025) shows an example IMDB review with token-level saliency. For both tasks, we trained transformer-based models and applied our techniques to generate formal explanations. Table 1 shows that, also for these models, our advantages are consistent—generation time is reduced by using the binary search-inspired traversal, and explanation size is reduced by using the bound propagation-based ranking. Note that, *conf ranking* does not apply here, as the output of these two transformers is a single value. For TaxiNet, it is the cross-track distance, whereas for IMDB, it uses sigmoid (i.e., positive if ≥ 0.5 and negative otherwise).

4.3 Comparison to Existing Formal and Non-formal Explanation Approaches

We compare VERIX+ to existing formal explainers in Table 2. Across all six models, we achieve *smaller* explanation sizes compared to (Izza et al. 2024), as they essentially employ saliency ranking for feature ordering. Their generation time is greater than ours, as their Dichotomic search includes *unnecessary* CHECK calls, whereas our binary search-inspired traversal avoids this inefficiency. We also compare

Table 2: Comparison between VERIX+ and other formal explainable AI approaches—(Izza et al. 2024) and (Huang and Marques-Silva 2023)—in terms of explanation size (number of pixels), generation time (seconds), and number of CHECK calls.

approaches	VERIX+			(Izza et al. 2024)			(Huang and Marques-Silva 2023)			(Huang and Marques-Silva 2023)		
traversal order	bound propagation			saliency			naive sequential			bound propagation		
traversal procedure	QuickXplain			Dichotomic search			QuickXplain (no optimization)			QuickXplain (no optimization)		
metrics	size	time	# CHECK	size	time	# CHECK	size	time	# CHECK	size	time	# CHECK
MNIST-FC	167.8	35.6	437.0	175.9	82.3	1203.4	265.3	42.7	607.7	167.8	45.0	577.4
MNIST-CNN	82.4	34.6	211.0	85.7	61.3	556.7	158.2	68.4	385.8	82.4	82.4	280.5
GTSRB-FC	251.6	140.0	511.2	464.3	1021.1	3825.6	461.2	190.2	706.1	251.6	211.3	761.3
GTSRB-CNN	275.1	251.0	559.3	479.2	1672.6	4040.3	464.7	372.4	736.9	275.1	275.1	832.7
CIFAR10-FC	378.9	251.9	764.7	464.7	1233.4	3911.5	462.0	227.7	705.2	378.9	621.4	1142.2
CIFAR10-CNN	488.6	478.9	984.9	609.3	2451.6	5218.9	558.2	430.9	881.1	488.6	708.3	1471.9

Table 3: Comparing VERIX+ explanation sizes (number of pixels) for normally (MNIST-FC) and adversarially (MNIST-FC-ADV) trained MNIST-FC models. ϵ is set to 5%, and ϵ-robust denotes the percentage of ϵ-robust samples.

samples	MNIST-FC					MNIST-FC-ADV				
	accuracy	correct		incorrect		accuracy	correct		incorrect	
		ϵ-robust	size	ϵ-robust	size		ϵ-robust	size	ϵ-robust	size
original	93.76%	4%	177.20	0.3%	398.85	92.85%	52.3%	128.22	2%	311.14
malicious	23.66%	0%	466.30	0%	562.07	82.66%	6.7%	298.57	0.3%	536.41

with (Huang and Marques-Silva 2023). There is no indication that they use a special traversal order (perhaps because their experiments involve networks with only five features). Therefore, we first run their algorithm using a simple sequential order, and as expected, our method produces smaller explanations. Ensuring a fair comparison, we run their algorithm using our optimized bound propagation-based traversal order, and see that our adaptation of QuickXplain is more efficient than theirs in terms of *fewer* CHECK calls, and, consequently, *less* generation time. We emphasize that this is not only supported by empirical evidence but also follows from a purely algorithmic analysis (see footnote 3).

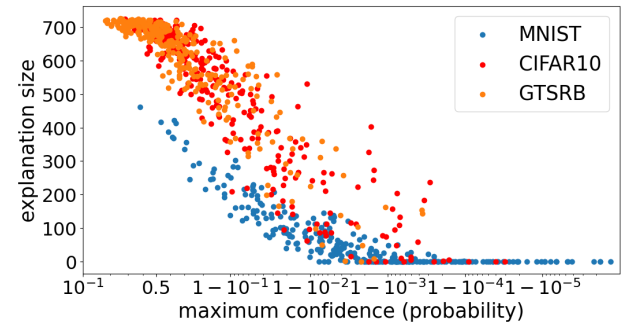
We also did a detailed comparison with non-formal explainers such as LIME (Ribeiro, Singh, and Guestrin 2016) and Anchors (Ribeiro, Singh, and Guestrin 2018) in Tables 5 and 6 of Appendix B.4 in (Wu et al. 2025). We observe that our explanations are much smaller. More significantly, we provide provable guarantees on the robustness of the produced explanations, as shown by "robust wrt # perturbations." The trade-off is that providing such guarantees requires more time.

4.4 Explanations from Adversarial Training

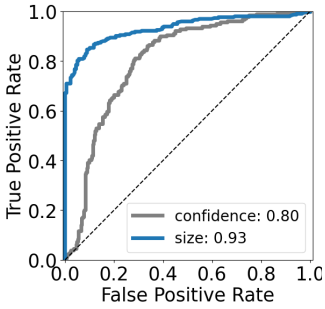
In Table 3, we compare our explanations for normally and *adversarially* trained MNIST-FC models on original and *malicious* samples. We describe the adversarial training process in Appendix B.5 of (Wu et al. 2025). We produce explanations for both *correct* and *incorrect* samples. For each table entry, we collected 300 samples and report their average explanation size (excluding ϵ-robust samples). Overall, we observe that both models have smaller explanations for original samples than for malicious samples, and for correct predictions than for incorrect predictions. Notably, the MNIST-FC-ADV model produces smaller explanations than MNIST-FC under all conditions. For instance, for original, correct samples, MNIST-FC-ADV produces 27.64% smaller explanations (128.22 vs. 177.20). This suggests that with adversarial training, the model learns more implicit information about the samples, e.g., which pixels likely contain the key information for prediction and which are subject to trivial perturbations; thus, it only needs to pay attention to fewer pixels to make a correct decision. In Figure 13 of Appendix B.5 in (Wu et al. 2025), we show examples of both original and malicious inputs. A similar phenomenon is observable in the ϵ-robust rate, which increases from 4% to 52.3% after adversarial training, since now the model has learned to focus on the principal features in the input and to become less sensitive to perturbations.

4.5 Using Explanation Size to Detect OOD and Misclassified Examples

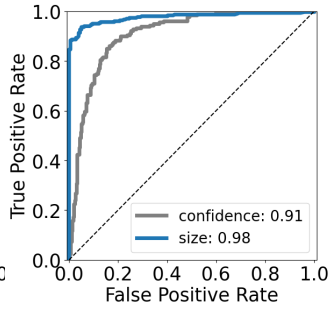
We also show that explanation size can help detect *out-of-distribution* (OOD) samples. Consider a scenario in which CIFAR10 and GTSRB images are fed into the MNIST-CNN model as OOD samples. We crop the images so they have the same size as MNIST images and use the OpenCV (Bradski 2000) library to convert color images to grayscale. The goal is to preserve the primary semantic meanings at the center of these images. We collected 900 samples in total—300 each from MNIST, CIFAR10, and GTSRB, and plot their maximum softmax probability and explanation size, as shown in Figure 4a (see also Appendix B.6 of (Wu et al. 2025)). We observe a significant separation between the in- and out-of-distribution samples, suggesting that smaller explanation sizes are associated with in-distribution inputs. A standard technique from previous work (Hendrycks and Gimpel 2017)



(a) MNIST, CIFAR10, GTSRB: confidence & size



(b) CIFAR10: AUROC



(c) GTSRB:AUROC

Figure 4: Detecting *out-of-distribution* examples from CIFAR10 and GTSRB for the MNIST-CNN model. (a) Explanation size and maximum confidence (*log* scale, see *linear* scale in Figure 7a of (Wu et al. 2025). (b)(c) ROC curves and AUROC values for OOD samples from CIFAR10 and GTSRB, respectively.

uses the maximum softmax probabilities for OOD detection. We compare using explanation size and confidence to detect OOD samples from CIFAR10 and GTSRB. From Figures 4b and 4c, we see that explanation size yields better ROC curves for both datasets and also achieves higher AUROC values, i.e., 93% on CIFAR10 and 98% on GTSRB. We perform similar OOD detection on the MNIST-FC model in Appendix B.6 of (Wu et al. 2025). We remark that this section aims to illustrate the utility of formal explanations, not to outperform state-of-the-art OOD detection methods, which are beyond this paper's scope. In Appendix B.7 of (Wu et al. 2025), we also show that explanation size is a useful proxy for detecting *misclassified* samples.

5 Related Work

Several approaches to formal explanations (Marques-Silva and Ignatiev 2022) have been explored recently. (Ignatiev, Narodytska, and Marques-Silva 2019) first proposed using abductive reasoning to compute formal explanations for neural networks by encoding them into a set of constraints and then deploying automated reasoning systems such as SMT solvers to solve the constraints. (La Malfa et al. 2021) brings in bounded perturbations, k-nearest neighbors and ϵ-balls to produce distance-restricted explanations for natural language models. Adapting the ϵ-perturbations to perception models for which inputs tend to be naturally bounded, (Wu, Wu, and Barrett 2023) proposes a feature-level saliency to obtain a ranking of the input features and thus produce empirically small explanations. In this paper, we utilize bound propagation-based techniques to obtain more fine-grained feature-level sensitivity. Compared to the existing saliency from (Wu, Wu, and Barrett 2023) and later adopted by (Izza et al. 2024), we obtain ϵ-perturbation-dependent traversal orders that lead to even smaller explanation sizes. To reduce generation time, (Huang and Marques-Silva 2023) mimics the QuickXplain algorithm (Junker 2004) to avoid traversing all the features in a linear way (experiments only include the linear case though). Our optimization of this QuickXplain adaptation further reduces the number of oracle calls; we also perform a complete experimental comparison between the linear and non-linear approaches. Additionally, we introduce binary search-based traversals to further improve time. The Dichotomic search method from (Izza et al. 2024) is also inspired by binary search, but it includes unnecessary oracle calls, as it restarts the search from the beginning every time it finds a "transition feature." Therefore, the number of oracle calls needed by their Dichotomic search in their experiments is "always larger than" the number of input features. In contrast, our binary search-based algorithm does not have such redundant oracle calls and thus achieves much reduced time. Finally, our confidence ranking strategy accelerates the CHECK procedure, which benefits all such oracle calls. To the best of our knowledge, this is the *first* time that this strategy has been proposed.

6 Discussion of Limitations

While our approach achieves improved scalability compared to existing formal explainers, we note that—with strict soundness and completeness guarantees—scalability is still limited. In particular, our method does not yet scale to contemporary large models, including large language models. We emphasize that this limitation arises from the inherent complexity of the underlying problem. Finding ways to help address this gap remains an important direction for future work. Moreover, although we use explanation size (e.g., number of pixels for images or tokens for texts) as a proxy for quality, there is no consensus on whether this is the most appropriate metric. Exploring alternative evaluation methods, such as human-in-the-loop assessments, would be a promising avenue for further investigation.

7 Conclusion and Future Work

We have presented the VERIX+ framework for computing optimal verified explanations with improved size and generation time. Future work could explore further techniques for improving the performance of verified explanation generation, perhaps by adapting parallel techniques from (Izza et al. 2024) or by finding ways to introduce approximations in order to gain scalability. It would also be interesting to evaluate explanations using a controlled setting (Kim et al. 2018) rather than simply size. Finally, we would like to explore additional applications, especially in areas where safety or fairness is crucial.

Acknowledgments

This work was funded in part by IBM as a founding member of the Stanford Institute for Human-centered Artificial Intelligence (HAI), a Ford Alliance Project (316280), the National Science Foundation (grant number 2211505), the Stanford CURIS program, and the Stanford Center for AI Safety.

References

Bradski, G. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Darwiche, A.; and Hirth, A. 2020. On the reasons behind decisions. In *Proceedings of the 24th European Conference on Artificial Intelligence*.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, 3–18.

Gowal, S.; Dvijotham, K. D.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T.; and Kohli, P. 2019. Scalable verified training for provably robust image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4842–4851.

Hendrycks, D.; and Gimpel, K. 2017. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *International Conference on Learning Representations*.

Henriksen, P.; and Lomuscio, A. R. 2020. Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*, 2513–2520.

Huang, X.; and Marques-Silva, J. 2023. From robustness to explainability and back again. *arXiv preprint arXiv:2306.03048*.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019. Abduction-based explanations for machine learning models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1511–1519.

Izza, Y.; Huang, X.; Morgado, A.; Planes, J.; Ignatiev, A.; and Marques-Silva, J. 2024. Distance-Restricted Explanations: Theoretical Underpinnings & Efficient Implementation. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning (KR 2024)*, 475–486. Association for the Advancement of Artificial Intelligence.

Julian, K. D.; Lee, R.; and Kochenderfer, M. J. 2020. Validation of image-based neural network controllers through adaptive stress testing. In *2020 IEEE 23rd international conference on intelligent transportation systems (ITSC)*, 1–7. IEEE.

Junker, U. 2004. Quickxplain: Preferred explanations and relaxations for over-constrained problems. In *Proceedings of the 19th national conference on Artifical intelligence*, 167–172.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International conference on computer aided verification*, 97–117. Springer.

Katz, G.; Huang, D. A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljić, A.; et al. 2019. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, 443–452.

Kim, B.; Wattenberg, M.; Gilmer, J.; Cai, C.; Wexler, J.; Viegas, F.; et al. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, 2668–2677. PMLR.

Krizhevsky, A.; et al. 2009. Learning multiple layers of features from tiny images. Technical report.

La Malfa, E.; Michelmore, R.; Zbrzezny, A. M.; Paoletti, N.; and Kwiatkowska, M. 2021. On Guaranteed Optimal Robust Explanations for NLP Models. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2658–2665.

LeCun, Y.; Cortes, C.; and Burges, C. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 4768–4777.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning Word Vectors for Sentiment Analysis. In Lin, D.; Matsumoto, Y.; and Mihalcea, R., eds., *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. Portland, Oregon, USA: Association for Computational Linguistics.

Marques-Silva, J.; and Ignatiev, A. 2022. Delivering Trustworthy AI through Formal XAI. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 12342–12350.

Mathiesen, F. B. 2022. Bound Propagation. *GitHub repository*.

Müller, M. N.; Makarchuk, G.; Singh, G.; Püschel, M.; and Vechev, M. 2022. PRIMA: general and precise neural network certification via scalable convex hull approximations. *Proceedings of the ACM on Programming Languages*, 6(POPL): 1–33.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: high-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, 1527–1535.

Shih, A.; Choi, A.; and Darwiche, A. 2018. A symbolic approach to explaining Bayesian network classifiers. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 5103–5111.

Singh, G.; Gehr, T.; Püschel, M.; and Vechev, M. 2019. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL): 1–30.

Stallkamp, J.; Schlipsing, M.; Salmen, J.; and Igel, C. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32: 323–332.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 6000–6010. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964.

Wang, S.; Zhang, H.; Xu, K.; Lin, X.; Jana, S.; Hsieh, C.-J.; and Kolter, Z. 2021. Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Neural Network Robustness Verification. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, volume 34, 29909–29921.

Wu, H.; Isac, O.; Zeljić, A.; Tagomori, T.; Daggitt, M.; Kokke, W.; Refaeli, I.; Amir, G.; Julian, K.; Bassan, S.; et al. 2024. Marabou 2.0: a versatile formal analyzer of neural networks. In *International Conference on Computer Aided Verification*, 249–264. Springer.

Wu, M.; Li, X.; Wu, H.; and Barrett, C. 2025. Efficiently Computing Compact Formal Explanations. *arXiv preprint arXiv:2409.03060*.

Wu, M.; Wu, H.; and Barrett, C. 2023. VeriX: Towards Verified Explainability of Deep Neural Networks. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 22247–22268. Curran Associates, Inc.

Xu, K.; Shi, Z.; Zhang, H.; Wang, Y.; Chang, K.-W.; Huang, M.; Kailkhura, B.; Lin, X.; and Hsieh, C.-J. 2020. Automatic perturbation analysis for scalable certified robustness and beyond. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713829546.

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018. Efficient neural network robustness certification with general activation functions. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 4944–4953.