VERIX: TOWARDS VERIFIED EXPLAINABILITY OF DEEP NEURAL NETWORKS

EXECUTIVE SUMMARY

VERIX (VERIfied eXplainability) introduces a pioneering system aimed at enhancing trustworthiness in AI systems by providing robust explanations and generating counterfactuals for deep neural networks. In safety-critical domains like autonomous driving and healthcare, understanding AI decisions is crucial for adoption and regulatory compliance. VERIX achieves this by leveraging constraint solving techniques to compute optimal explanations, ensuring they come with provable guarantees against perturbations in the input space. A feature-level sensitivity heuristic is integrated, efficiently approximating global optima, which significantly improves scalability for high-dimensional inputs and large models. Furthermore, the system automatically computes counterfactuals along decision boundaries, offering tangible examples of how changes in input features can alter model predictions. The algorithmic approach of VERIX employs an automated reasoning sub-procedure to traverse through input features, using a heuristic based on feature-level sensitivity to determine the traversal order. An off-the-shelf neural network verification tool validates specifications, ensuring the soundness and optimality of explanations. VERIX's effectiveness is demonstrated through evaluations on image recognition benchmarks and a real-world scenario involving autonomous aircraft taxiing, highlighting its potential for broader applications in critical domains.

BACKGROUND

The use of AI in critical areas like self-driving cars and healthcare requires trustworthy AI systems. A crucial aspect of trustworthiness is explainability, meaning AI should be able to explain its actions in a way humans can understand.

Early methods for explainable AI, like LIME and SHAP, create simple models around specific inputs to give explanations. However, these methods lack formal guarantees and can't be relied upon in high-risk situations. For example, a healthcare AI might give an explanation based on skin lesions for a diagnosis, but if patients with similar lesions but different skin tones get different diagnoses, the explanation can be misleading and may hide biases in the model. Additionally, these methods often need access to training data, which might not always be available due to privacy issues.

LIME

Methodology: LIME creates simple, interpretable models around specific inputs to explain AI decisions.

Local Scope: Focuses on explaining predictions for individual instances rather than the model as a whole.

Limitations: Lacks formal guarantees, can produce misleading explanations, and depends on access to training data.

SHAP

- Methodology: SHAP uses game theory to allocate contributions of each feature to a prediction across all possible permutations.
- Global Scope: Provides explanations considering the impact of each feature on predictions across the entire dataset.
- Limitations: Can be computationally expensive and may still require access to training data.

ENTER VERIX

- Recent research in the field of explainable AI aims to create explanations that are not only understandable but also trustworthy. These explanations should guarantee soundness, meaning that if certain input features are fixed, the model's prediction remains consistent.
- Some earlier approaches only considered large or "unbounded" changes to the input data. These changes might be too broad to be useful in understanding the model's behavior accurately.
- To address these limitations, newer methods introduce two types of smaller, more controlled changes or "perturbations" to the input data:

- ϵ-ball perturbations: These handle continuous changes, like adjusting the intensity of colors in an image.
- k-NN box closure: This method focuses on nearby data points, making the perturbation space discrete and manageable.

VERIX CONT:

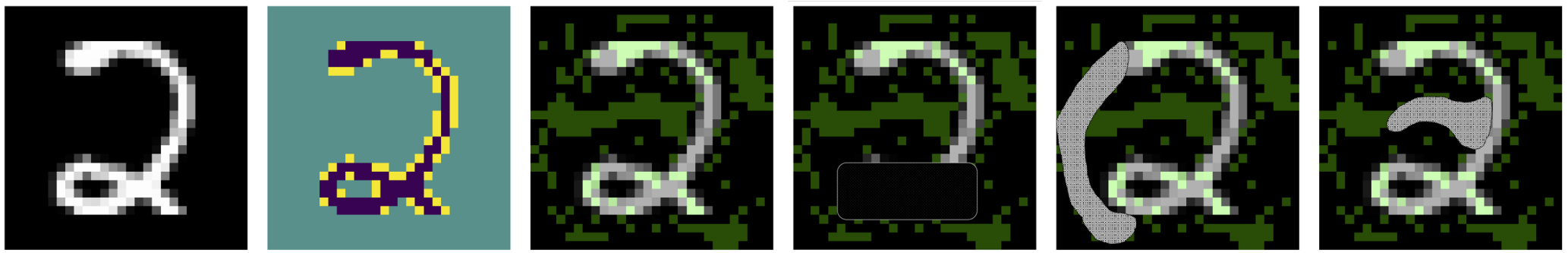
- The paper introduces VERIX (VERIfied eXplainability), a tool designed to produce robust explanations and generate counterfactuals for deep neural networks. Here's what VERIX aims to achieve:

- Constraint Solving: Utilizes mathematical techniques to ensure explanations are accurate against both continuous and infinite perturbations.
- Feature-level Sensitivity: Enhances scalability by efficiently approximating global optima for high-dimensional inputs and large models.
- Relationship with Counterfactuals: Highlights the connection between explanations and "what-if" scenarios, showing alternative outcomes at no extra cost.
- Extensive Evaluation: VERIX's performance is tested across various perception models, including real-world applications like autonomous aircraft taxiing.

WHY IS VERIX BETTER?

- Formal Guarantees: Unlike LIME and SHAP, Verix offers formal guarantees, ensuring more reliable explanations.
- Bias Detection: Verix can uncover biases in AI models by identifying discrepancies in explanations across different groups, ensuring fairer outcomes.
- Privacy-preserving: Verix operates without needing access to the raw training data, addressing privacy concerns while maintaining explainability.
- In summary, while LIME and SHAP are valuable for explainability, Verix offers enhanced reliability, fairness, and privacy, making it a superior choice for trustworthy AI in critical domains.

VERIX PROVIDES A MORE HOLISTIC APPROACH TO EXPLAINING AI DECISIONS COMPARED TO MODEL-AGNOSTIC METHODS LIKE ANCHORS. WHILE ANCHORS FOCUSES ON SELECTING PROMINENT SEGMENTS IN AN IMAGE TO EXPLAIN A MODEL'S DECISION, VERIX CONSIDERS BOTH THE SALIENT FEATURES AND BACKGROUND ELEMENTS. FOR INSTANCE, IN AN IMAGE OF THE NUMBER "2", ANCHORS MIGHT HIGHLIGHT SPECIFIC SEGMENTS LIKE THE PURPLE AND YELLOW PARTS, WHILE VERIX ALSO TAKES INTO ACCOUNT THE ABSENCE OF CERTAIN FEATURES IN THE BACKGROUND THAT COULD INFLUENCE THE PREDICTION. THIS COMPREHENSIVE APPROACH ENSURES THAT VERIX'S EXPLANATIONS ARE MORE ROBUST AND ACCURATE, CAPTURING NUANCES THAT MAY BE OVERLOOKED BY OTHER



(a) Original “2” (b) Segmentation (c) VERIX (d) “2” into “7” (e) “2” into “0” (f) “2” into “3”

$$\forall\, \mathbf{x}^{\mathbf{B}'}.\ \left\|\mathbf{x}^{\mathbf{B}} - \mathbf{x}^{\mathbf{B}'}\right\|_p \leq \epsilon \Rightarrow |f(\mathbf{x}) - f(\mathbf{x}')| \leq \delta,$$

OPTIMAL ROBUST EXPLANATIONS

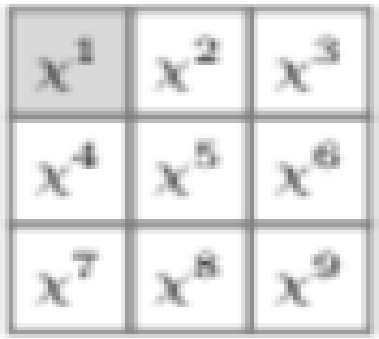
$$\forall \chi \in \mathbf{x}^{\mathbf{A}}.\ \exists\, \mathbf{x}^{\mathbf{B}'}, \chi'.\ \left\|\left(\mathbf{x}^{\mathbf{B}} \oplus \chi\right) - \left(\mathbf{x}^{\mathbf{B}'} \oplus \chi'\right)\right\|_p \leq \epsilon \wedge \left|f(\mathbf{x}) - f(\mathbf{x}')\right| > \delta,$$

- The existing work on explaining AI decisions, such as abductive explanations, prime implicants, and sufficient reasons, defines a formal explanation as the minimal subset of input features responsible for a model's decision. This means that any changes to the remaining features won't alter the prediction. Building on this, recent work introduces bounded perturbations to compute explanations for NLP models. However, our approach differs in several key aspects: we consider a uniform cost function, focus on continuous perturbations instead of discrete ones, and allow for bounded output variations to accommodate both classification and regression tasks.

- In our definition of an optimal robust explanation, given a neural network, an input, a manipulation magnitude, and a discrepancy parameter, a robust explanation is a set of input features such that any perturbations within a certain bound won't change the prediction beyond a specified limit. We introduce the concept of "irrelevant features" that can be perturbed without affecting the prediction. If an input is already robust to perturbations, it means all features are irrelevant, requiring a larger perturbation to find a meaningful explanation. Conversely, if changing any feature affects the prediction, the entire input becomes the explanation.

- Our definition aims for local optimality, finding the smallest subset of features explaining the model's decision. While a globally optimal explanation, the smallest among all local optima, is desirable, it's computationally challenging. To address this, we propose a tractable heuristic that approximates the ideal and performs well in practice. This approach is illustrated graphically in Figure 2, where variants of the input are shown to either lie on the same side of the decision boundary or lead to different classifications when perturbed.

VERIX EXAMPLE



- Imagine we have an input x with 9 features, labeled {χ1 through χ9}. We're using a classification neural network f and a perturbation limit ε with a norm $p=\infty$. The algorithm goes through each feature of the input to decide if it's important for the model's decision or not.
- The algorithm starts with both the explanation set A and the irrelevant set B empty. It goes through each feature one by one and decides whether to add it to A or B based on whether changing it affects the model's prediction.
- Let's say the algorithm starts with χ1. It checks if changing χ1 within the ε limit affects the prediction. If not, it adds χ1 to the irrelevant set B. It repeats this process for each feature until all are visited.
- Eventually, the algorithm determines that $\langle\chi4,\chi5,\chi8\rangle$ are the important features (added to A), meaning that if these features are fixed, the model's prediction won't change with any perturbations within ε on the other features. Additionally, for each feature in A, the algorithm finds a counterfactual example showing how changing that feature could alter the prediction.

VERIX ALGORITHM

1. **Algorithm Overview**: Algorithm 1 describes the VERIX procedure, which takes a neural network *f* and an input *x* as input. It outputs an optimal explanation *xA* with respect to certain parameters like perturbation magnitude *ε*, norm *p*, and discrepancy *δ*, along with a set of counterfactuals *C*.

2. **Initialization**: The procedure starts by initializing three sets: *A* (explanation index set), *B* (irrelevant feature set), and *C* (set of counterfactuals). The prediction for the input *x* is recorded as *c*, regardless of its accuracy.

3. **Feature Traversal**: It goes through each feature *χi* in the input according to a specified traversal order. For each feature, it checks if perturbing it within a certain limit *ε*, while fixing the rest of the features, leads to a consistent prediction. This is done using a formula *ϕ* that encodes these conditions.

4. **Automated Reasoning**: An automated reasoning sub-procedure called CHECK examines whether the specification holds for the network. If it does, the feature is added to the irrelevant set *B*; otherwise, it's added to the explanation index set *A*. If a feature is added to *A*, it means that feature contributes to explaining the prediction.

5. **Counterfactuals**: For features added to *A*, a counterexample *m* is generated, representing a counterfactual for that feature. This counterexample shows how changing that feature, along with the irrelevant features, could alter the prediction.

6. **Result**: The procedure continues until all features are traversed. At the end, it returns the optimal explanation *xA* and the set of counterfactuals *C*.

.

Algorithm 1 VERIX (VERIfied eXplainability)

Input: neural network f and input $\mathbf{x} = \langle \chi^1, \ldots, \chi^d \rangle$
Parameter: ϵ-perturbation, norm p, and discrepancy δ
Output: optimal robust explanation $\mathbf{x}^{\mathbf{A}}$, counterfactuals $\mathbf{C}$ for each $\chi \in \mathbf{x}^{\mathbf{A}}$

1: **function** VERIX($f, \mathbf{x}$)
2: $\quad \mathbf{A}, \mathbf{B}, \mathbf{C} \mapsto \emptyset, \emptyset, \emptyset$
3: $\quad c \mapsto f(\mathbf{x})$
4: $\quad \hat{c} \mapsto f(\hat{\mathbf{x}})$
5: $\quad \pi \mapsto$ TRAVERSALORDER($\mathbf{x}$)
6: $\quad$ **for** i in π **do**
7: $\quad\quad \mathbf{B}^+ \mapsto \mathbf{B} \cup \{i\}$
8: $\quad\quad \phi \mapsto (\left\| \hat{\chi}^{\mathbf{B}^+} - \chi^{\mathbf{B}^+} \right\|_p \leq \epsilon)$
9: $\quad\quad \phi \mapsto \phi \wedge (\hat{\chi}^{\Theta \setminus \mathbf{B}^+} = \chi^{\Theta \setminus \mathbf{B}^+})$
10: $\quad\quad (\texttt{HOLD}, m) \mapsto$ CHECK($f, \phi \Rightarrow |\hat{c} - c| \leq \delta$)
11: $\quad\quad$ **if** HOLD **then** $\mathbf{B} \mapsto \mathbf{B}^+$
12: $\quad\quad$ **else** $\mathbf{A} \mapsto \mathbf{A} \cup \{i\}$; $\mathbf{C} \mapsto \mathbf{C} \cup \{m\}$
13: $\quad$ **return** ($\mathbf{x}^{\mathbf{A}}$, $\mathbf{C}$)

FEATURE-LEVEL SENSITIVITY TRAVERSAL

1. **Feature-Level Sensitivity**: This heuristic is based on the concept of feature-level sensitivity. Given an input *x* and a neural network *f*, the sensitivity of a feature *χi* is calculated by comparing the model's output with and without the feature. This is done by applying a transformation *T* to the feature and observing how it affects the model's confidence.
2. **Transformation Types**: Common transformations include deletion, where the feature is removed (set to 0), and reversal, where the feature's value is negated (subtracted from itself). These transformations help understand how the presence or absence of a feature influences the model's prediction.
3. **Ranking Features**: Once sensitivity values are obtained for each feature with respect to a chosen transformation, the features are ranked based on their sensitivity. Features with the least sensitivity are considered first in the traversal order, while those with the highest sensitivity are considered last

THANK YOU

Sohaib Ashraf (20k-0488)
Muhammad Shahzaib (20k-1067)
Hassan Malik (20k-0380)