Unfooling Perturbation-Based Post Hoc Explainers

Zachariah Carmichael, Walter J. Scheirer

University of Notre Dame
zcarmich@nd.edu, walter.scheirer@nd.edu

Abstract

Monumental advancements in artificial intelligence (AI) have lured the interest of doctors, lenders, judges, and other professionals. While these high-stakes decision-makers are optimistic about the technology, those familiar with AI systems are wary about the lack of transparency of its decision-making processes. Perturbation-based post hoc explainers offer a model agnostic means of interpreting these systems while only requiring query-level access. However, recent work demonstrates that these explainers can be fooled adversarially. This discovery has adverse implications for auditors, regulators, and other sentinels. With this in mind, several natural questions arise – how can we audit these black box systems? And how can we ascertain that the auditee is complying with the audit in good faith? In this work, we rigorously formalize this problem and devise a defense against adversarial attacks on perturbation-based explainers. We propose algorithms for the detection (`CAD-Detect`) and defense (`CAD-Defend`) of these attacks, which are aided by our novel conditional anomaly detection approach, `KNN-CAD`. We demonstrate that our approach successfully detects whether a black box system adversarially conceals its decision-making process and mitigates the adversarial attack on real-world data for the prevalent explainers, `LIME` and `SHAP`. The code for this work is available at https://github.com/craymichael/unfooling.

Introduction

As a result of the many recent advancements in artificial intelligence (AI), a significant interest in the technology has developed from high-stakes decision-makers in industries such as medicine, finance, and the legal system (Lipton 2018; Miller and Brown 2018; Rudin 2019). However, many modern AI systems are black boxes, obscuring undesirable biases and hiding their deficiencies (Szegedy et al. 2014; Hendrycks et al. 2021; Lipton 2018). This has resulted in unexpected consequences when these systems are deployed in the real world (O'Neil 2016; Buolamwini and Gebru 2018; McGregor 2021). Accordingly, regulatory and legal ordinance has been proposed and implemented (EU and Parliament 2016; U.S.-EU TTC 2022; European Commission 2021).

All of this naturally leads to the question: how can we audit opaque algorithms? Post hoc explanation methods offer a way of understanding black box decision-making processes by estimating the influence of each variable on the decision value. Unfortunately, there is a medley of incentives that may motivate an organization to withhold this information, whether it is financial, political, personal, or otherwise. Indeed, these explainers are demonstrably deceivable (Slack et al. 2020; Baniecki 2022) — if an organization is aware that its algorithms are under scrutiny, it is capable of falsifying how its algorithms operate. This begs the question — how do we know that the auditee is complying faithfully?

In this work, we explore the problem of employing perturbation-based post hoc explainers to audit black box algorithms. To the best of our knowledge, this is the first paper to address the aforementioned questions and provide a solution. We explore a multi-faceted problem setting in which the auditor needs to ascertain that the algorithms of an organization: 1) do not violate regulations or laws in their decision-making processes and 2) do not adversarially mask said processes. Our contributions are as follows:

- We formalize real-world adversarial attack and defense models for the auditing of black box algorithms with perturbation-based post hoc explainers. We then formalize the general defense problem against adversarial attacks on explainers, as well as against the pragmatic scaffolding-based adversarial attack on explainers (Slack et al. 2020).
- We propose a novel unsupervised conditional anomaly detection algorithm based on k-nearest neighbors: `KNN-CAD`.
- We propose adversarial detection and defense algorithms for perturbation-based explainers based on any conditional anomaly detector: `CAD-Detect` and `CAD-Defend`, respectively.
- Our approach is evaluated on several high-stakes real-world data sets for the popular perturbation-based explainers, `LIME` (Ribeiro, Singh, and Guestrin 2016) and `SHAP` (Lundberg and Lee 2017). We demonstrate that the detection and defense approaches using `KNN-CAD` are capable of detecting whether black box models are adversarially concealing the features used in their decision-making processes. Furthermore, we show that our method exposes the features that the adversaries attempted to mask, mitigating the scaffolding-based attack.

- We introduce several new metrics to evaluate the fidelity of attacks and defenses with respect to explanations and the black box model.
- We conduct analyses of the explanation fidelity, hyperparameters, and sample-efficiency of our approach.

Background

Local Black Box Post Hoc Explainers Of particular relevance to auditing an algorithm is in understanding individual decisions. Explainable AI (XAI) approaches afford transparency of otherwise uninterpretable algorithms (Barredo Arrieta et al. 2020) and are conducive to auditing (Akpinar et al. 2022; Zhang, Cho, and Vasarhelyi 2022). Local post hoc explainers, notably `LIME` and `SHAP`, do so by estimating the contribution of each feature to a decision value. These particular explainers are classified as model agnostic and black box, which satisfies the conditions of an audit — the auditor has no *a priori* knowledge of the model class and has only query-level access. Both approaches produce explanations by fitting linear models to a dataset generated by perturbing the neighborhood about a sample. Let $\boldsymbol{\mathcal{D}} = (\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}) = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ be the data set where each data sample $\mathbf{x}_i \in \mathbb{R}^F$ has F features and each label $y_i \in \mathbb{N}_0$ represents one of C classes encoded as an integer in the range $[0..C)$. We denote the black box classifier as $f : \boldsymbol{\mathcal{X}} \rightarrow \boldsymbol{\mathcal{Y}}$ and the explainer as g. The general problem these explainers solve in order to explain an instance $\mathbf{x}_i$ is given by Eq. (1)

$$\underset{g_{\mathbf{x}_i} \in \mathcal{G}}{\operatorname{argmin}} \sum_{\mathbf{x}_j^{(g)} \in \boldsymbol{\mathcal{X}}_i^{(g)}} \left(f(\mathbf{x}_j^{(g)}) - g_{\mathbf{x}_i}(\mathbf{x}_j^{(g)}) \right)^2 \pi_{\mathbf{x}_i}(\mathbf{x}_j^{(g)}) + \Omega(g_{\mathbf{x}_i}) \tag{1}$$

The minimization objective is a function of the linear model $g_{\mathbf{x}_i}$ from the set of all linear models $\mathcal{G}$, the neighborhood function $\pi_{\mathbf{x}_i}$, and the regularization function Ω. With a slight abuse of notation, $\pi_{\mathbf{x}_i}$ both generates the neighborhood of $\mathbf{x}_i$ ($\boldsymbol{\mathcal{X}}_i^{(g)}$) and gives the proximity of each $\mathbf{x}_j^{(g)}$ to $\mathbf{x}_i$. The latter two functions are defined using game-theoretic means for `SHAP` and empirical means for `LIME`. Both explainers produce explanations as a set of feature contributions $\mathcal{E}_i = \{a_{ij}\}_{j=1}^{F}$ that describes the contribution, or importance, of each feature to the decision value y_i. In our notation, a_{ij} indicates the explained contribution of the j^{th} feature to y_i.

Adversarial Attacks on Explainers Szegedy et al. were the first to discover the phenomena of *adversarial examples* (Szegedy et al. 2014) — human-indistinguishable manipulation of the input space that results in misclassification. It is worth noting that naturally-occurring adversarial examples have been shown to exist (Hendrycks et al. 2021), though the focus of this work concerns human adversaries. In contrast to the literature on neural network adversarial attacks (Ren et al. 2020), attacks on explainers are comparatively less explored, especially those that are perturbation-based (Baniecki 2022). Slack et al. were the first to propose an adversarial attack on black box post hoc explainers (Slack et al. 2020). The motivation for the attack arose from the observation that explainer perturbations are often off of the data manifold. In the attack, the adversary aims to detect whether some $\mathbf{x}_i$ originates from $\boldsymbol{\mathcal{X}}$ or from the explainer-generated samples, $\boldsymbol{\mathcal{X}}^{(g)}$, using a scaffolding classifier. $\boldsymbol{\mathcal{X}}^{(g)}$ is generated by perturbing each $\mathbf{x}_i \in \boldsymbol{\mathcal{X}}$, creating a set of new instances $\boldsymbol{\mathcal{X}}_i^{(g)}$. The union of each $\boldsymbol{\mathcal{X}}_i^{(g)}$ yields the complete set, i.e., $\boldsymbol{\mathcal{X}}^{(g)} = \bigcup_i \boldsymbol{\mathcal{X}}_i^{(g)}$. If the adversary determines that the query originates from an explainer, then it can elect to predict using an *unbiased* model. In effect, the adversary lies to the explainer about how it uses the features. Formally, given a biased predictor f_{biased} and an unbiased predictor f_{unbiased}, the adversary can toggle between predictors using an out-of-distribution (ood) detector d according to Eq. (2).

$$f(\mathbf{x}_i) = \begin{cases} f_{\text{biased}}(\mathbf{x}_i) & \text{if } d(\mathbf{x}_i) = 1 \\ f_{\text{unbiased}}(\mathbf{x}_i) & \text{otherwise} \end{cases} \tag{2}$$

The detector d is trained on the samples $\boldsymbol{\mathcal{X}} \cup \boldsymbol{\mathcal{X}}^{(g)}$ with a corresponding label of '0' if the sample is ood (from $\boldsymbol{\mathcal{X}}^{(g)}$) and '1' otherwise (from $\boldsymbol{\mathcal{X}}$). Put simply, the detector d determines if the input sample originated from real data or is an explainer-generated perturbation. If the sample origin is predicted to be an explainer, the unbiased predictor f_{unbiased} makes predictions in an innocuous manner, e.g., independent of sensitive features such as race. The authors demonstrate that the attack obscures the biased use of features, such as making predictions based on race, from `LIME` and `SHAP` while maintaining near-perfect prediction fidelity on in-distribution samples.

In related but tangential works, attacks have been developed on the model and explainer simultaneously. Abdukhamidov et al. introduce a gradient-free black box attack on XAI systems that manipulates model predictions without significantly altering the explanations by post hoc explainers (Abdukhamidov et al. 2022). Closely related, Zhan et al. develop a joint attack with the same implications (Zhan et al. 2022). Both works consider explainers that require gradient-level access to the model and are unsuitable for auditing. Noppel et al. propose a trigger-based neural backdoor attack on XAI systems that simultaneously manipulates the prediction and explanation of gradient-based explainers (Noppel, Peter, and Wressnegger 2022). Again, the attack scenario in our work deviates from this model.

Adversarial Defense for Explainers In a similar fashion, defense against adversarial attacks is well explored in the literature (Ren et al. 2020). However, there is relatively scarce work in defending against adversarial attacks on explainers. Ghalebikesabi et al. address the problems with the locality of generated samples by perturbation-based post hoc explainers (Ghalebikesabi et al. 2021a). They propose two variants of `SHAP`, the most relevant being `Neighborhood SHAP` which considers local reference populations to improve on-manifold sampling. Their approach is able to mitigate the scaffolding attack (Slack et al. 2020). While this is a notable achievement, it is unclear how the approach compares to baseline `SHAP` with respect to quantitative and qualitative measures of explanation quality, whether it still upholds the properties of `SHAP` explanations, and other concerns (Ghalebikesabi et al. 2021b). Also related is a constraint-driven variant of `LIME`, `CLIME` (Shrotri et al.

2022). CLIME has been demonstrated to mitigate the scaffolding attack, but requires hand-crafted constraints based on domain expert knowledge and for data to be discrete.

A step forward in explainer defense, Schneider et al. propose two approaches to detect manipulated Grad-CAM explanations (Schneider, Meske, and Vlachos 2022). The first is a supervised approach that determines if there is (in)consistency between the explanations of an ensemble of models and explanations that are labeled as manipulated or not. The second is an unsupervised approach that determines if the explanations of f are as sufficient to reproduce its predictions as those from an ensemble of models. They conclude that detection without domain knowledge is difficult. Aside from the deviant attack model in their work, we do not require that deceptive explanations be labeled (an expensive and error-prone process) and we require that only a single model be learned (the authors use 35 CNNs as the ensemble in experiments).

Recently, a perturbation-based explainer coined EMaP was introduced that also helps to mitigate the scaffolding attack (Vu, Mai, and Thai 2022). Similar to Neighborhood SHAP, it improves upon its perturbation strategy to create more in-distribution samples. This is accomplished by perturbing along orthogonal directions of the input manifold, which is demonstrated to maintain the data topology more faithfully.

Conditional Anomaly Detection Vanilla anomaly detection aims to discover observations that deviate from a notion of normality, typically in an unsupervised paradigm (Ruff et al. 2021). Of interest in this work is conditional, also referred to as contextual, anomaly detection. This type of anomaly is an observation that is abnormal in a particular context, e.g., in time or space. Formally, a set of conditional anomalies $\mathcal{A}$ is given by Eq. (3)

$$\mathcal{A} = \{(\mathbf{x}_i, \mathbf{y}_i) \in (\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}) \mid \mathbb{P}(\mathbf{y}_i \mid \mathbf{x}_i) \leq \tau\},\ \tau \geq 0 \quad (3)$$

where $\mathbb{P}$ is the probability measure of some probability density function (pdf) that characterizes normality and τ is a low-probability threshold separating normal and abnormal observations. Following the terminology in (Song et al. 2007), $\boldsymbol{\mathcal{X}}$ is the set of environmental variables that the set of observed variables $\boldsymbol{\mathcal{Y}}$ is conditioned on.

Song et al. proposed the first conditional anomaly detector using Gaussian mixture models — two sets of Gaussians model the environmental and observed variables, respectively, while a learned probability mapping function determines how the Gaussians in each set map to one another (Song et al. 2007). Since, several approaches have been proposed based on classical and deep learning techniques — we point to this comprehensive survey for further reading (Ruff et al. 2021). In this work, we propose a new conditional anomaly detection method as 1) the deep learning techniques are data-hungry and 2) most techniques do not consider or fair well with categorical data, which is plentiful in real-world high-stakes data: credit scoring (FICO 2018), recidivism risk scoring (Angwin et al. 2016), etc.

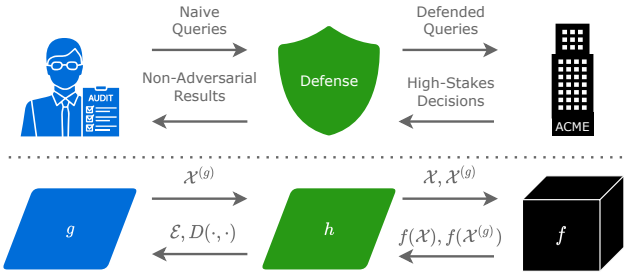

Figure 1: An overview of the adversarial attack and defense scenario. The top panel mirrors the bottom panel formalism on a higher level. Our defense approach provides defense to auditors from potential adversarial behavior of the auditee, ACME. Given the explainer g-generated samples $\boldsymbol{\mathcal{X}}^{(g)}$ and reference samples $\boldsymbol{\mathcal{X}}$, the defense h queries the black box f. With the results, h gives the auditor the defended explanations $\mathcal{E}$ and a measure of adversarial behavior $D(\cdot,\cdot)$.

The Problem and a Solution

The Attack and Defense Models In the attack scenario, an organization algorithmically makes decisions on behalf of users (customers, patients, employees, etc.) using a black box predictor f. Note that f can be a black box due to it being either uninterpretable or proprietary (or both) (Rudin 2019). We will refer to the organization as the *ACME* (*Corporation*) for the remainder of the paper. We recommend referring back to Figure 1 as a guide on how elements of the scenario relate. The attacker only provides query-level access to f, e.g., via an API, a web form, or even a social media app facial filter. In the case of f being a classifier, only the final decision value is available, not the class-wise probabilities. Moreover, the class of model that f belongs to is assumed to be unknown. Thus, to ensure that ACME meets legal or regulatory compliance, an auditor uses a local post hoc explainer, g, to determine how features are used to make individual decisions. Due to a variety of incentives (financial, political, personal, etc.), an adversary within ACME desires to conceal the behavior of f. The adversary does not know what explainer is employed but may exploit weaknesses that are shared among common black box post hoc explanation algorithms. Furthermore, the adversary is unaware of when the audit occurs and is only aware of the information contained within the queries made to f. The general attack objective is to minimize the multi-objective problem given by Eq. (4)

$$\min_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}^{(g)} \cup \boldsymbol{\mathcal{X}}} \Big(L_f\left(f(\mathbf{x}_i), y_i\right), |\{j \mid a_{ij} \in \mathcal{E}_i^{\text{sensitive}}, r(\mathcal{E}_i^-, a_{ij}) < r(\mathcal{E}_i^-, a_{ik}), \forall a_{ik} \in \mathcal{E}_i^{\text{harmless}}\}|, \max(\mathcal{E}_i^{\text{sensitive}}) \Big) \quad (4)$$

where $\mathcal{E}_i^- = \mathcal{E}_i^{\text{sensitive}} \cup \mathcal{E}_i^{\text{harmless}} = g(\mathbf{x}_i; f)$ is an ordered set of feature contributions for the i^{th} sample, $\mathcal{E}_i^{\text{sensitive}}$ and $\mathcal{E}_i^{\text{harmless}}$ are the contributions for the sensitive and harmless features, respectively, L_f is some metric that measures the error between its two scalar arguments, and r gives the index of its

second argument within its first argument. $\mathcal{E}_i^-$ is ordered by decreasing value of the contribution magnitude to the negative (adverse) outcome. Put simply, the first objective quantifies the error between the predictor and the ground truth, the second objective is the number of feature contributions for sensitive features that are greater than those of harmless features, and the third objective quantifies the magnitude of the feature contributions for sensitive features. The advantage of this formulation is that it applies to any type of attack that intends to manipulate the attribution of features during an audit. I.e., it is not tied to just the scaffolding attack.

In this work, the scaffolding attack is employed by ACME as described in the "Background" section. This attack is multi-objective and aims to minimize Eq. (4) by finding some $d \in \mathcal{D}_{\text{ood}}$ that minimizes Eq. (5)

$$\min_{(\mathbf{x}_i,t_i)\in(\boldsymbol{\mathcal{X}}^{(g)}\cup\boldsymbol{\mathcal{X}},\mathcal{T})} \Big([\![t_i{=}0]\!] L_f(f_{\text{biased}}(\mathbf{x}_i), f(\mathbf{x}_i)) \,, \\ [\![t_i{=}1]\!] L_a(g(\mathbf{x}_i; f_{\text{unbiased}}), g(\mathbf{x}_i; f)) \Big) \tag{5}$$

where $[\![\cdot]\!]$ are Iverson brackets, $\mathcal{D}_{\text{ood}}$ is the set of all out-of-distribution detectors, $\mathcal{T}$ is a set of flags indicating whether $\mathbf{x}_i \in \boldsymbol{\mathcal{X}}^{(g)}$, and L_a is some metric that measures the error between its two scalar arguments. Naturally, there is a trade-off between prediction fidelity (objective one) and unbiased explanation to an auditor (objective two). The "Experiments" section describes how f_{biased} and f_{unbiased} are selected.

We now formulate the detection and defense objectives — here, the goal of *detection* is to determine whether f is using the scaffolding attack, whereas the goal of *defense* is to ascertain whether the scaffolding attack is used for individual samples. As we will detail in the next section, the latter can be used to "unfool" the explainers. Formally, the general sample-wise defense objective is given by Eq. (6)

$$\operatorname*{argmin}_{h\in\mathcal{H}} L_h\left(h(\mathbf{x}_i; f), \mathbb{P}\left(f(\mathbf{x}_i) \mid \mathbf{x}_i\right)\right) \text{ s.t. } \mathbf{x}_i \in \boldsymbol{\mathcal{X}}^{(g)} \cup \boldsymbol{\mathcal{X}} \tag{6}$$

where $\mathcal{H}$ is the class of conditional distribution estimators, h is the detector that estimates the conditional pmf p with support $\{0, 1\}$, and $\mathbb{P}$ is a probability measure of p[1]. The pmf p is the distribution of normality for f. Here, L_h is some metric that measures the error between its two scalar arguments. The role of h is to determine the likelihood of a prediction by f given a sample. The threshold-based detection is in turn used to classify anomalies for individual samples as in Eq. (3).

To globally detect an attack, the same objective should be minimized as for defense. However, discriminating an attack instead is determined by having h estimate Eq. (7)

$$\left[\!\!\left[D\left(p(f(\mathbf{x}_i) \mid \mathbf{x}_i), p(f(\mathbf{x}_i^{(g)}) \mid \mathbf{x}_i^{(g)})\right) \geq \tau_{\text{global}} \right]\!\!\right] \tag{7}$$

where $\tau_{\text{global}} \geq 0$ is a threshold, $\mathbf{x}_i^{(g)} \in \boldsymbol{\mathcal{X}}^{(g)}$, $\mathbf{x}_i \in \boldsymbol{\mathcal{X}}$, and D is an asymmetric measure of statistical distance between the two distributions that is permitted to take on negative values and maintains the identity of indiscernibles (i.e., $D(x, y){=}0$ iff $x{=}y$). With a properly calibrated h and sufficient samples to represent the distribution of normality, $D(\cdot, \cdot) > 0$ if $d(\mathbf{x}_i){=}1$ (or if f is not adversarial) and $D(\cdot, \cdot) \leq 0$ otherwise.

[1]To conserve space, we omit random variables of arguments to the pmf p (and $\mathbb{P}$) and use realizations instead.

Detection, Defense and `KNN-CAD` In this section, we describe a non-parametric approach to detect conditional anomalies based on k-nearest neighbors: k-nearest neighbors conditional anomaly detector (`KNN-CAD`). We then describe general algorithms for the detection (`CAD-Detect`) and defense (`CAD-Defend`) of adversarial attacks on g for any given h. For simplicity, we treat f as a classifier in describing `KNN-CAD` — in the case that f is a regressor, we cast it to a classifier by binning its output. Since f does not return class-wise probabilities in the problem setting, we exploit the fact that we have a single discrete observed variable y_i. On a high level, the main idea of `KNN-CAD` is to compare the labels of the neighbors of some $\mathbf{x}_i$ to $f(\mathbf{x}_i)$ — the disagreement of the labels of its neighbors determines the degree of abnormality. Algorithms 1 (`KNN-CAD.fit`) and 2 (`KNN-CAD.score_samples`) formalize this process. With samples representing normality $\boldsymbol{\mathcal{X}}$, the standard k-nearest neighbors algorithm is fit to the data in `KNN-CAD.fit`. These samples are collected by the auditor and are very unlikely to overlap with data that f (and d, if applicable) were trained on. After the fit is made, each $\mathbf{x}_i \in \boldsymbol{\mathcal{X}}$ is scored by `KNN-CAD.score_samples`, which gives the scored samples $\mathcal{S}$ (lower is more abnormal). Subsequently, the threshold τ is set to the percentile ϵ of $\mathcal{S}$. The rounding operator used in computing the percentile is denoted as round$(\cdot)$.

In `KNN-CAD.score_samples` (Algorithm 2), the pmf $p(f(\mathbf{x}_i) \mid \mathbf{x}_i)$ is estimated. To do so, the k-nearest neighbors of, and distances from, each $\mathbf{x}_i$ are computed. For each $\mathbf{x}_i$, the labels of its neighbors are retrieved. For the neighbors belonging to each class, the corresponding distances are gathered (denoted by, e.g., $\mathbf{d}_0$ for each $y_i = 0$) and then aggregated by the function ϕ. The aggregator ϕ estimates the statistical distance between $\mathbf{x}_i$ and its neighbors by computing, e.g., the median, mean, or maximum value. If the vector argument of ϕ is empty, then the output of the function is ∞. In the final step, we are interested in measuring the dynamic range between the aggregate distances corresponding to the label of the queried sample, d_{y_j}, and to the alternative label(s), $d_{\neg y_j}$. The dynamic range is defined as the ratio between two values on the logarithmic scale as in Eq. (8).

$$\texttt{dynamic\_range}(a, b) = \log\left(\frac{a}{b}\right) \tag{8}$$

The values given by the dynamic range of such distances are treated as logits, so we apply the standard logistic function σ to map the values to probabilities as in Eq. (9)

$$\begin{aligned} \zeta(d_{\neg y_j}, d_{y_j}) &= \sigma(\texttt{dynamic\_range}(d_{\neg y_j}, d_{y_j})) \\ &= \frac{1}{1 + \exp(-\log\left(\frac{d_{\neg y_j}}{d_{y_j}}\right))} \\ &= \frac{1}{1 + \frac{d_{y_j}}{d_{\neg y_j}}} = \frac{d_{\neg y_j}}{d_{\neg y_j} + d_{y_j}}. \end{aligned} \tag{9}$$

Algorithm 1: `KNN-CAD.fit`$(f, \boldsymbol{\mathcal{X}}, k, \phi, \epsilon)$

Input: f, $\boldsymbol{\mathcal{X}}$, k, ϕ, ϵ
Output: h, the `KNN-CAD` object

`// Fit the distribution of normality`
$h \leftarrow \texttt{KNN}(\boldsymbol{\mathcal{X}})$;
$\mathcal{S} \leftarrow h.\texttt{score\_samples}(f, \boldsymbol{\mathcal{X}}, k, \phi)$;
$\mathcal{S}' \leftarrow \texttt{sort}(\mathcal{S})$; `// sort ascending`
$h.\tau \leftarrow \mathcal{S}'[\text{round}(\epsilon \cdot |\mathcal{S}'|)]$; `// set threshold`
$h.\boldsymbol{\mathcal{X}}_{\text{train}} \leftarrow \boldsymbol{\mathcal{X}}$; $h.\boldsymbol{\mathcal{Y}}_{\text{train}} \leftarrow f(\boldsymbol{\mathcal{X}})$;
return h

Algorithm 2: `KNN-CAD.score_samples`$(h, f, \boldsymbol{\mathcal{X}}, k, \phi)$

Input: h, f, $\boldsymbol{\mathcal{X}}$, k, ϕ
Output: $\mathcal{S}$, the scored samples

$\boldsymbol{\mathcal{Y}} \leftarrow f(\boldsymbol{\mathcal{X}})$;
`// Get the k-nearest neighbor distances and indices`
$\mathcal{D}, \mathcal{I} \leftarrow h.\texttt{neighbors}(\boldsymbol{\mathcal{X}}, k)$;
$\mathcal{S} \leftarrow$ `new array`;
for $(\mathbf{d}, \mathbf{i}, y_j) \in (\mathcal{D}, \mathcal{I}, \boldsymbol{\mathcal{Y}})$ **do**
  $\mathbf{y}_i \leftarrow h.\boldsymbol{\mathcal{Y}}_{\text{train}}[\mathbf{i}]$; `// Gather neighbor labels`
  $\mathbf{d}_0 \leftarrow \mathbf{d}[[\![\mathbf{y}_i = 0]\!]]$; $\mathbf{d}_1 \leftarrow \mathbf{d}[[\![\mathbf{y}_i = 1]\!]]$;
  $d_0 \leftarrow \phi(\mathbf{d}_0)$; $d_1 \leftarrow \phi(\mathbf{d}_1)$;
  if $y_j = 1$ **then**
    $d_{y_j} \leftarrow d_1$; $d_{\neg y_j} \leftarrow d_0$;
  else
    $d_{y_j} \leftarrow d_0$; $d_{\neg y_j} \leftarrow d_1$;
  $s \leftarrow \zeta(d_{\neg y_j}, d_{y_j})$;
  $\mathcal{S}.\texttt{append}(s)$;
return $\mathcal{S}$

Algorithm 3: `CAD-Detect`$(f, g, h, \Theta_h, \boldsymbol{\mathcal{X}}, n_{\text{train}}, \tau_{\text{global}})$

Input: f, g, h, Θ_h, $\boldsymbol{\mathcal{X}}$, n_{train}, τ_{global}
Output: Whether f is behaving adversarially toward g

$\boldsymbol{\mathcal{X}}_{\text{train}}, \boldsymbol{\mathcal{X}}_{\text{test}} \leftarrow \boldsymbol{\mathcal{X}}[:n_{\text{train}}], \boldsymbol{\mathcal{X}}[n_{\text{train}}:]$;
$h.\texttt{fit}(f, \boldsymbol{\mathcal{X}}_{\text{train}}, \Theta_h)$;
$\mathcal{S}_{\text{test}} \leftarrow h.\texttt{score\_samples}(f, \boldsymbol{\mathcal{X}}_{\text{test}}, \Theta_h)$;
$\boldsymbol{\mathcal{X}}^{(g)}_{\text{test}} \leftarrow \{\}$;
for $\mathbf{x}_i \in \boldsymbol{\mathcal{X}}_{\text{test}}$ **do**
  $\boldsymbol{\mathcal{X}}^{(g)}_{\text{test}} \leftarrow \boldsymbol{\mathcal{X}}^{(g)}_{\text{test}} \cup \pi^{(g)}_{\mathbf{x}_i}(\mathbf{x}_i)$;
$\mathcal{S}^{(g)}_{\text{test}} \leftarrow h.\texttt{score\_samples}(f, \boldsymbol{\mathcal{X}}^{(g)}_{\text{test}}, \Theta_h)$;
$a_{\text{test}} \leftarrow \texttt{AUC}(\texttt{ECDF}(\mathcal{S}_{\text{test}}))$;
$a^{(g)}_{\text{test}} \leftarrow \texttt{AUC}(\texttt{ECDF}(\mathcal{S}^{(g)}_{\text{test}}))$;
return $[\![(a^{(g)}_{\text{test}} - a_{\text{test}}) \geq \tau_{\text{global}}]\!]$

Algorithm 4: `CAD-Defend`$(f, g, h, \mathbf{x}_i, \Theta_h, n_p)$

Input: f, g, h, $\mathbf{x}_i$, Θ_h, n_p
Output: $\boldsymbol{\mathcal{X}}^{(g)}_i$, (more) in-distribution perturbations

$\boldsymbol{\mathcal{X}}^{(g)}_i \leftarrow \pi^{(g)}_{\mathbf{x}_i}(\mathbf{x}_i)$;
$\mathcal{S}^{(g)}_i \leftarrow h.\texttt{score\_samples}(f, \boldsymbol{\mathcal{X}}^{(g)}_i, \Theta_h)$;
`// Remove abnormal samples`
$\boldsymbol{\mathcal{X}}^{(g)}_i \leftarrow \boldsymbol{\mathcal{X}}^{(g)}_i[[\![\mathcal{S}^{(g)}_i > h.\tau]\!]]$;
$n'_p \leftarrow n_p - |\boldsymbol{\mathcal{X}}^{(g)}_i|$;
if $n'_p \neq 0$ **then**
  $\boldsymbol{\mathcal{X}}'^{(g)}_i \leftarrow$ `CAD-Defend`$(f, g, h, \mathbf{x}_i, \Theta_h, n'_p)$;
  $\boldsymbol{\mathcal{X}}^{(g)}_i \leftarrow \boldsymbol{\mathcal{X}}^{(g)}_i \cup \boldsymbol{\mathcal{X}}'^{(g)}_i$;
return $\boldsymbol{\mathcal{X}}^{(g)}_i$

We denote this probability mapping function as ζ. The algorithm `KNN-CAD.score_samples` is written for the case of f being a binary classifier, but it is easily generalizable to multi-class by taking the expected value of the comparison of d_{y_j} to every other $d_{\neg y_j}$, i.e., $\mathbb{E}_{d_{\neg y_j} \in \mathbf{d}_{\neg y_j}}[\zeta(d_{\neg y_j}, d_{y_j})]$.

In Algorithm 3, the procedure for adversarial attack detection, `CAD-Detect`, is detailed. First, the samples $\boldsymbol{\mathcal{X}}$ representing normality are partitioned into $\boldsymbol{\mathcal{X}}_{\text{train}}$ (n_{train} samples) and $\boldsymbol{\mathcal{X}}_{\text{test}}$ ($|\boldsymbol{\mathcal{X}}| - n_{\text{train}}$ samples). Thereafter, h is fit using $\boldsymbol{\mathcal{X}}_{\text{train}}$ and the (hyper-)parameters pertinent to h, Θ_h. After $\boldsymbol{\mathcal{X}}^{(g)}_{\text{test}}$ is collected, h estimates $p(f(\mathbf{x}_i) \mid \mathbf{x}_i)$ and $p(f(\mathbf{x}^{(g)}_i) \mid \mathbf{x}^{(g)}_i)$ for $\boldsymbol{\mathcal{X}}_{\text{test}}$ and $\boldsymbol{\mathcal{X}}^{(g)}_{\text{test}}$, respectively. The statistical distance D between these two conditional distributions is estimated by comparing the areas under the empirical cumulative distributions of $\mathcal{S}_{\text{test}}$ and $\mathcal{S}^{(g)}_{\text{test}}$. We denote these areas as a_{test} and $a^{(g)}_{\text{test}}$, respectively. In the algorithm, we denote the empirical cumulative distribution function as `ECDF` and the function that computes the area under a curve using the trapezoidal rule as `AUC`. Intuitively, a greater difference between $a^{(g)}_{\text{test}}$ and a_{test} indicates a concentrated region of low-probability samples (likely adversary behavior of f), whereas a smaller difference indicates that an attack was likely not carried out. In practice, the number of perturbed samples generated around each sample, n_p, is relatively large — fewer samples are necessary to accurately estimate p. We explore the sample-efficiency of h in the "Experiments" section.

The algorithm for defending against adversarial attacks, `CAD-Defend`, is detailed in Algorithm 4. The approach is a fairly straightforward modification to the neighborhood generation function $\pi^{(g)}_{\mathbf{x}_i}$. For each sample $\mathbf{x}_i$ to be explained by g, the perturbed samples $\boldsymbol{\mathcal{X}}^{(g)}_i$ generated by $\pi^{(g)}_{\mathbf{x}_i}$ are scored by h. The samples with scores below the threshold $h.\tau$ are discarded and `CAD-Defend` recursively builds the remaining samples until $|\boldsymbol{\mathcal{X}}^{(g)}_i| = n_p$. In practice, the recursive depth can be limited with either an explicit limit or by reducing $h.\tau$. However, an auditor will prioritize faithful scrutiny of ACME's algorithms over the speed of the explainer.

On a final note, neither `CAD-Detect` nor `CAD-Defend` is tied to `KNN-CAD` — rather, they are compatible with any $h \in \mathcal{H}$ (any conditional anomaly detector).

Time and Space Complexity The time and space complexity of every introduced algorithm are listed in Table 1 and derived in detail in Appendix I. We assume that `KNN-CAD` is used as h in Algorithms 3 and 4. Note that `KNN-CAD` uses the ball tree algorithm in computing nearest neighbors. We let $T_f(\cdot)$ and $S_f(\cdot)$ be the functions that give the time and space complexity of f, respectively, and R be the number of recursions in Algorithm 4. In practice, the time and space complexity of each algorithm are dominated by that of the model to audit, f. Hence, reducing the num-

Alg.	Type	Complexity	Practical Complexity
1	Time	$\mathcal{O}(N((F+k)\log N + T_f(F)))$	$\mathcal{O}(NT_f(F))$
	Space	$\mathcal{O}(N(F+k+S_f(F)))$	$\mathcal{O}(NS_f(F))$
2	Time	$\mathcal{O}(N(k\log N + T_f(F)))$	$\mathcal{O}(NT_f(F))$
	Space	$\mathcal{O}(N(k+S_f(F)))$	$\mathcal{O}(NS_f(F))$
3	Time	$\mathcal{O}(Nn_p(k\log(Nn_p) + T_f(F)))$	$\mathcal{O}(Nn_pT_f(F))$
	Space	$\mathcal{O}(Nn_p(k+S_f(F)))$	$\mathcal{O}(Nn_pS_f(F))$
4	Time	$\mathcal{O}(Rn_p(k\log n_p + T_f(F)))$	$\mathcal{O}(Rn_pT_f(F))$
	Space	$\mathcal{O}(n_p(k+S_f(F)))$	$\mathcal{O}(n_pS_f(F))$

Table 1: The time and space complexity of each algorithm introduced in this paper: Algorithms 1 (`KNN-CAD.fit`), 2 (`KNN-CAD.score_samples`), 3 (`CAD-Detect`), and 4 (`CAD-Defend`). With practical complexity, it is assumed that $T_f(F) \gg (k+F)\log N$ and $S_f(F) \gg k+F$, e.g., as is the case with DNNs and most decision trees.

ber of queries is desirable. We explore the sample-efficiency of our approach in the "Experiments" section. If the queries are pre-computed, then our algorithms using `KNN-CAD` take linearithmic time and linear space.

Experiments

We consider three real-world high-stakes data sets to evaluate our approach:

- The Correctional Offender Management Profiling for Alternative Sanctions (**COMPAS**) dataset was collected by *ProPublica* in 2016 for defendants from Broward County, Florida (Angwin et al. 2016). The attributes of individuals are used by the COMPAS algorithm to assign recidivism risk scores provided to relevant decision-makers.
- The **German Credit** data set, donated to the University of California Irvine (UCI) machine learning repository in 1994, comprises a set of attributes for German individuals and the corresponding lender risk (Dua and Graff 2017).
- The **Communities and Crime** data set combines socio-economic US census data (1990), US Law Enforcement Management and Administrative Statistics (LEMAS) survey data (1990), and US FBI Uniform Crime Reporting (UCR) data (1995) (Redmond and Baveja 2002). Covariates describing individual communities are posited to be predictive of the crime rate.

Each data set contains at least one protected attribute that should not be used to make a decision in order to meet regulatory compliance. We follow the attacks as proposed in (Slack et al. 2020) and recall them here. See Table 2 for the sensitive features that each f_{biased} uses and the harmless features that each f_{unbiased} uses to make decisions for each experiment. As in (Slack et al. 2020), the OOD detector d is selected to be a random forest. Uncorrelated features are generated from the discrete uniform distribution $\mathcal{U}\{0,1\}$, and two different f_{unbiased} are considered for each experiment when uncorrelated features are considered: one with one uncorrelated feature and another with two uncorrelated features. The two classifiers, f_{biased} and f_{unbiased}, are rule-based and return the target listed in Table 2 as a function of the selected features(s). See Appendix A for further details.

Data Set	Sensitive Feature	Harmless Feature(s)	Target
COMPAS	African-American	Uncorrelated	High Risk of Recidivism
German Credit	Gender	Income % Toward Loan	Good Customer
CC	Count of White Population	Uncorrelated	Violent Crime Rate

Table 2: Data sets on which the attack and defense approaches are evaluated. The Communities & Crime data set is abbreviated as CC. In each experiment, f_{biased} makes decisions based on sensitive features and f_{unbiased} makes decisions based on harmless features.

For each task, we consider both cases in which ACME elects to and not to employ the attack. Each data set contains a train and test partition, $\boldsymbol{\mathcal{D}}_{\text{train}}$ and $\boldsymbol{\mathcal{D}}_{\text{test}}$, respectively — both are considered to be representative of normality (as defined in "The Problem and a Solution" section). f is trained on $\boldsymbol{\mathcal{D}}_{\text{train}}$ and scrutinized by the auditor on $\boldsymbol{\mathcal{D}}_{\text{test}}$ using g. Recall that f is a function of both f_{biased} and f_{unbiased} as defined in Eq. (2). Our defense approach aided by h employs the algorithms `CAD-Detect` and `CAD-Defend`. The defense is evaluated using $\boldsymbol{\mathcal{D}}_{\text{test}}$ as the auditor does not have access to $\boldsymbol{\mathcal{D}}_{\text{train}}$. The results across all tasks are shown in Table 3. Appendix B specifies all reproducibility details. In the proceeding sections, we propose and employ several fidelity metrics to evaluate the quality of both the attack and defense.

How effective is the attack? We evaluate the efficacy of the attack with several metrics, expanding on its original evaluation (Slack et al. 2020). The classifier f should be accurate on in-distribution data (fidelity_f) as given by Eq. (10)

$$\text{fidelity}_f(\boldsymbol{\mathcal{X}}) = \frac{1}{|\boldsymbol{\mathcal{X}}|}\sum_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}} [\![f(\mathbf{x}_i) = f_{\text{actual}}(\mathbf{x}_i)]\!] \tag{10}$$

where

$$f_{\text{actual}}(\mathbf{x}_i) = \begin{cases} f_{\text{biased}}(\mathbf{x}_i) & \text{if attack is deployed} \\ f(\mathbf{x}_i) & \text{otherwise} \end{cases} \tag{11}$$

When the attack is deployed, we can evaluate the quality of d (fidelity_d) as the balanced accuracy between samples from $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{X}}^{(g)}$ as in Eq. (12)

$$\text{fidelity}_d(\boldsymbol{\mathcal{X}} \cup \boldsymbol{\mathcal{X}}^{(g)}) = \frac{1}{2}\left(\frac{1}{|\boldsymbol{\mathcal{X}}|}\sum_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}} [\![d(\mathbf{x}_i)=1]\!] + \frac{1}{|\boldsymbol{\mathcal{X}}^{(g)}|}\sum_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}^{(g)}} [\![d(\mathbf{x}_i)=0]\!]\right). \tag{12}$$

To quantify the degree that g is "fooled," we compare the feature contributions to the features used by f_{actual} from a feature-selectivity perspective. The features used by f_{actual} can be split into selected ($\mathcal{F}_{\text{selected}}$) and unselected ($\mathcal{F}_{\neg\text{selected}}$)

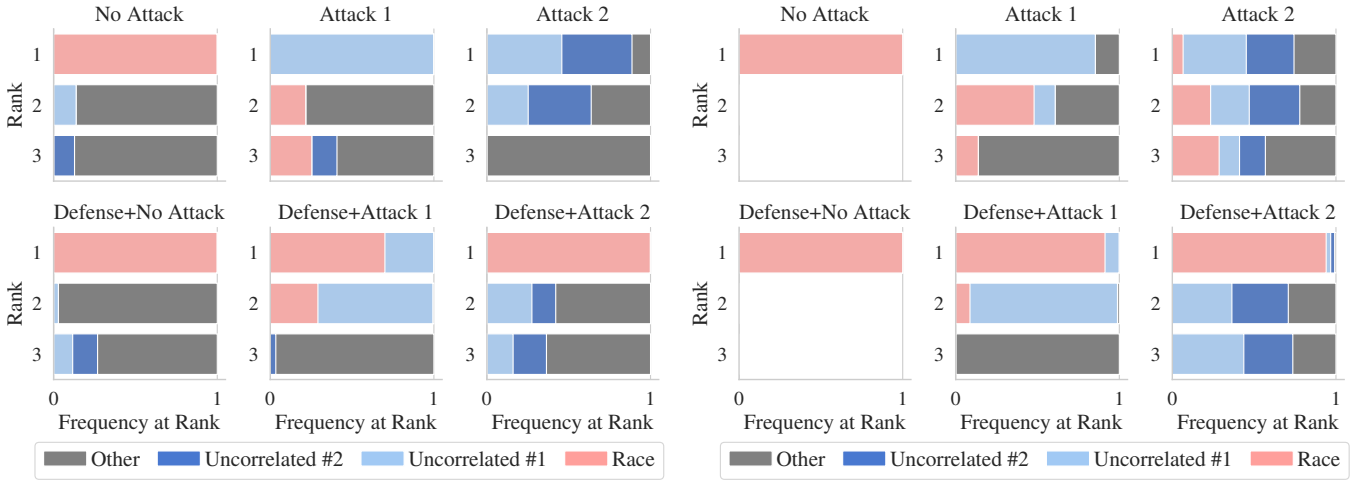

Figure 2: The top-3 most important features identified by LIME (left) and SHAP (right) with and without the defense on the COMPAS task. Our defense mitigates the attack and demonstrates that ACME attempted to conceal that race is an important factor in deciding the risk of recidivism. In the legend, "Other" represents any other COMPAS feature that is not "Race" or an "Uncorrelated" feature.

features. In turn, we can estimate the fidelity of g as Eq. (13)

$$\text{fidelity}_g(\boldsymbol{\mathcal{X}}) = \frac{1}{F \cdot |\boldsymbol{\mathcal{X}}|} \sum_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}} \rho\big(\{F-1 | j \in \mathcal{F}_{\text{selected}}\} \cup \{0 | j \in \mathcal{F}_{\neg\text{selected}}\}, \{r(\mathcal{E}_i^-, a_{ij}) | j \in \mathcal{F}\}\big) \tag{13}$$

where $\mathcal{F} = \mathcal{F}_{\text{selected}} \cup \mathcal{F}_{\neg\text{selected}}$, $|\mathcal{F}|{=}F$, and ρ gives the Spearman rank correlation coefficient between its two ordered arguments. Put simply, the correlation increases when features in $\mathcal{F}_{\text{selected}}$ are ranked higher than those in $\mathcal{F}_{\neg\text{selected}}$ by an explainer, i.e., explanations are more faithful.

As shown in Table 3, the attack is quite effective — fidelity_f remains high when the attack is active while fidelity_g decreases. This indicates that d successfully toggles between f_{biased} and f_{unbiased}, which is supported by the fidelity_d scores. Figure 2 qualitatively shows the efficacy of the attack on the COMPAS data set by visualizing the top-3 most frequent features in explanations.

Can we detect the attack? We introduce metrics to evaluate h and CAD-Detect in their ability to detect attacks. Here, d_{actual} is d when a scaffolding attack is deployed and a dummy function that returns '1' otherwise. The ability of h to model $\mathbb{P}(f(\mathbf{x}_i)|\mathbf{x}_i)$ is given by the weighted mean-square-error of h and d_{actual} (fidelity_h) as in Eq. (14)

$$\text{fidelity}_h(\boldsymbol{\mathcal{X}} \cup \boldsymbol{\mathcal{X}}^{(g)}) = 1 - \frac{1}{2}\left(\frac{1}{|\boldsymbol{\mathcal{X}}|}\sum_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}} (d_{\text{actual}}(\mathbf{x}_i) - h(\mathbf{x}_i))^2 + \frac{1}{|\boldsymbol{\mathcal{X}}^{(g)}|}\sum_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}^{(g)}} \left(d_{\text{actual}}(\mathbf{x}_i^{(g)}) - h(\mathbf{x}_i^{(g)})\right)^2\right) \tag{14}$$

where $h(\cdot)$ is shorthand for h.score_samples($\{\cdot\}$). Since h gives probabilities, the error term is bounded to the range $[0, 1]$ which we subtract from '1' in order for the score to be interpreted as fidelity. To evaluate the quality of CAD-Detect, we compare the areas between the cumulative distributions of scores derived from $\boldsymbol{\mathcal{X}}^{(g)}$ and $\boldsymbol{\mathcal{X}}$ as described in Algorithm 3: $\Delta_{\text{cdf}} := a_{\text{test}}^{(g)} - a_{\text{test}}$. This score should be higher if there is an attack deployed and lower otherwise.

Across all experiments, we use KNN-CAD as h. Our approach is able to detect every attack without false positives when $\tau_{\text{global}} \in [0.00, 0.12]$ for SHAP and $\tau_{\text{global}} \in [0.11, 0.12]$ for LIME. There is a trade-off in fidelity_h and fidelity_d — the more off-manifold the perturbations are, the harder it is to fit $\mathbb{P}(f(\mathbf{x}_i) \mid \mathbf{x}_i)$ (without an attack), whereas it is harder to fit d when the perturbations are more on-manifold. The more principled game-theoretic perturbation scheme of SHAP demonstrably improves on that of LIME, yet it still can be gamed adversarially. Furthermore, a higher Δ_{cdf} without an attack deployed is indicative of more perturbed samples being off-manifold — discriminating attacks using SHAP has a far greater margin between Δ_{cdf} scores than LIME.

Can we "unfool" explainers? In order for an explainer to be "unfooled," it should produce explanations that 1) agree with its own explanations without an attack present and 2) agree with the feature selection by the black box algorithm. We propose $\text{infidelity}_{\text{CAD-Defend},g}$ to quantify how well CAD-Defend accomplishes the former goal. This metric measures the disagreement of the explanations when using CAD-Defend to those only using g as in Eq. (15)

$$\text{infidelity}_{\text{CAD-Defend},g}(\boldsymbol{\mathcal{X}}) = \frac{1}{F \cdot |\boldsymbol{\mathcal{X}}|}\sum_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}} \sum_{(a_{ij}^{(g)}, a_{ij}^{(h)}) \in (\mathcal{E}_i^{(g)}, \mathcal{E}_i^{(h)})} (a_{ij}^{(g)} - a_{ij}^{(h)})^2 \tag{15}$$

where $\mathcal{E}_i^{(g)}$ is the explanation from g without an attack, $\mathcal{E}_i^{(h)}$ is the explanation when h is used for defense during an attack, and $a_{ij}^{(*)}$ is as defined in the "Background" section.

Data Set	Attack	N_{hl}	g = LIME							g = SHAP						
			Fidelity				CAD-Detect	CAD-Defend		Fidelity				CAD-Detect	CAD-Defend	
			f	d	g	h	Δ_{cdf}	inf_g	fid_f	f	d	g	h	Δ_{cdf}	inf_g	fid_f
COMPAS	✗	–	1.00	–	0.31	0.93	0.11	0.00	0.31	1.00	–	0.29	0.99	-0.02	0.00	0.29
	✓	1	0.99	0.99	0.23	0.81	0.30	0.05	0.30	0.93	0.82	0.24	0.77	0.20	0.02	0.28
	✓	2	0.99	0.99	0.10	0.80	0.30	0.00	0.31	0.91	0.82	0.24	0.77	0.25	0.01	0.26
German Credit	✗	–	1.00	–	0.27	0.78	0.07	0.00	0.27	1.00	–	0.24	0.86	-0.05	0.00	0.27
	✓	1	1.00	0.99	0.18	0.70	0.18	0.00	0.27	0.83	0.71	0.12	0.61	0.12	0.00	0.27
CC	✗	–	1.00	–	0.21	0.80	0.09	0.00	0.16	1.00	–	0.08	0.85	0.00	0.00	0.20
	✓	1	1.00	0.99	0.16	0.79	0.12	0.01	0.20	0.97	0.85	0.07	0.79	0.13	0.01	0.19
	✓	2	1.00	0.99	0.17	0.79	0.12	0.00	0.31	0.99	0.85	0.07	0.77	0.12	0.00	0.30

Table 3: The (in)fidelity and Δ_{cdf} scores of the attack and defense across the real-world data sets when KNN-CAD is used as h. N_{hl} is the number of harmless features used by f_{unbiased} and is unspecified when no attack is deployed. See the text for the differing definitions of fidelity for each algorithm. We abbreviate $\text{infidelity}_{\text{CAD-Defend},g}$ as inf_g and $\text{fidelity}_{\text{CAD-Defend},f}$ as fid_f. Our approach successfully detects the scaffolding attack while maintaining explanation fidelity for both explainers.

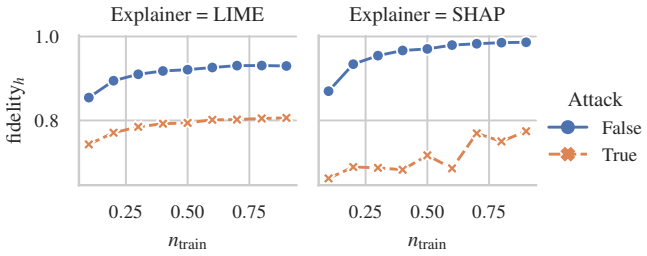

Figure 3: Sample-efficiency of KNN-CAD on the COMPAS data set when using LIME and SHAP as g — fidelity$_h$ is plotted as a function of the training proportion of the COMPAS data (n_{train}).

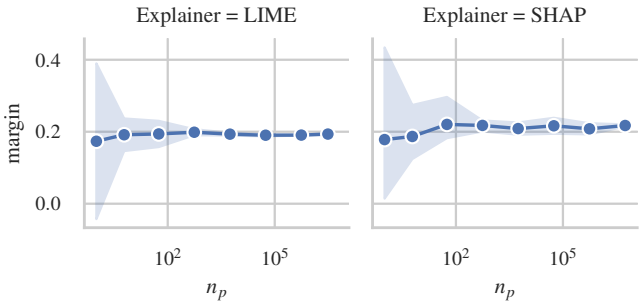

Figure 4: Sample-efficiency of CAD-Detect on the COMPAS data set when using LIME and SHAP as g — the margin between the Δ_{cdf} scores with and without an attack is plotted as a function of the number of explainer perturbations (n_p).

To evaluate against the true feature selection of f_{actual}, we use the same metric definition as fidelity$_g$ and refer to it as $\text{fidelity}_{\text{CAD-Defend},f}$.

The infidelity scores in Table 3 are near-zero across all tasks, demonstrating that our defense has very little disagreement with the explainer g when g is not under attack. In addition, the $\text{fidelity}_{\text{CAD-Defend},f}$ scores exceed those of fidelity$_g$ for all experiments when an attack is deployed and are close to fidelity$_g$ without an attack. A final piece of evidence that our defense mitigates the attack is shown in Figure 2 — comparing the top-3 most important features with and without the defense confirms that the defense is highly successful. See Appendix E for the same figure with the remaining explainers and data sets.

Sample Efficiency We evaluate the training sample-efficiency of our approach. Figure 3 plots fidelity$_h$ as a function of the training proportion of the COMPAS data when using LIME and SHAP as g. Because of the strong inductive bias and nonparametric nature of KNN-CAD, the algorithm hardly degrades in performance even when 10% of the data is in use. This in part indicates the sample-efficiency of the CAD-Detect and CAD-Defend algorithms when KNN-CAD is used. We also evaluate the sample-efficiency of CAD-Detect on the COMPAS data set when using LIME and SHAP as g. The margin between the Δ_{cdf} scores with and without an attack is plotted as a function of the number of explainer perturbations, n_p. The main benefit of increasing n_p is to increase the consistency of the detection score. When $n_p > 1,000$ the variance tapers off for both explainers, which is just a small percentage of the millions of explainer-generated perturbations when all test set samples are explained. These findings are quite notable as the complexity bottleneck is due to querying f as discussed in "The Problem and a Solution" section.

Additional Analyses We include analyses of the hyperparameters of the three core algorithms, KNN-CAD, CAD-Detect and CAD-Defend, in Appendix F.

Discussion

In this work, we introduced several novel algorithms to defend against adversarial attacks on perturbation-based post

hoc explainers: KNN-CAD for conditional anomaly detection, CAD-Detect for attack detection, and CAD-Defend to improve the fidelity of explanations when under attack. We rigorously formalized the attack and defense models, as well as introduced new quantitative metrics to evaluate the quality of the attack and defense. Our approach demonstrably mitigated the scaffolding attack across several real-world high-stakes data sets. The results indicate that it is easier to defend SHAP than LIME due to its more realistic data perturbations.

A limitation to consider is that the realistic samples used in the training set for the defense algorithm can be expensive or difficult to collect. Moreover, in realistic scenarios, an API to a black box may be costly to query with the perturbed samples generated by explainers. In practice, explainer queries should be rate-limited so as to not arise suspicion from the auditee. On this note, we do not consider the case when the adversary irregularly deploys the attack. We point to (Schneider, Meske, and Vlachos 2022) which characterizes this attack and demonstrates that explanations that are infrequently manipulated can be difficult to detect. In addition, we considered the case of an adversary masking malicious behavior. However, the motivation for such behavior could arise for privacy reasons or to protect intellectual property from model extraction attacks (Tramèr et al. 2016).

We analyze and alleviate a single shortcoming of post hoc explainers. However, the explainers we consider have also been shown to be inconsistent, unfaithful, and intractable (Krishna et al. 2022; Bordt et al. 2022; den Broeck et al. 2021; Garreau and von Luxburg 2020; Carmichael and Scheirer 2021a,b). Consequently, a potential source of negative societal impact in this work arises from practitioners overtrusting post hoc explainers (Kaur et al. 2020). Nevertheless, our study demonstrates that the explainers backed with our proposed defense not only detect adversarial behavior but also faithfully identify the most important features in decisions. Moreover, if an explainer is not to be trusted, our approach can at least exploit it to identify misbehaving algorithms. In future work, the ramifications of an adversarial organization caught red-handed should be explored in the context of existing regulatory guidelines.

Acknowledgments

We thank Derek Prijatelj[2] for helpful discussions in the early stages of the conditional anomaly detection formalism. Funding for this work comes from the University of Notre Dame.

References

Abdukhamidov, E.; Juraev, F.; Abuhamad, M.; and Abuhmed, T. 2022. Black-box and Target-specific Attack Against Interpretable Deep Learning Systems. In *Asia Conference on Computer and Communications Security*. ACM.

Akpinar, N.-J.; Leqi, L.; Hadfield-Menell, D.; and Lipton, Z. 2022. Counterfactual Metrics for Auditing Black-Box Recommender Systems for Ethical Concerns. In *Workshop on Responsible Decision Making in Dynamic Environments, International Conference on Machine Learning, ICML*, volume 162 of *Proceedings of Machine Learning Research*. PMLR.

Angwin, J.; Larson, J.; Mattu, S.; and Kirchner, L. 2016. Machine bias: there's software used across the country to predict future criminals. And it's biased against blacks. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Accessed: 2022-05-01.

Baniecki, H. 2022. Adversarial Explainable AI. https://hbaniecki.com/adversarial-explainable-ai/. Accessed: 2022-11-29.

Barredo Arrieta, A.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; Garcia, S.; Gil-Lopez, S.; Molina, D.; Benjamins, R.; Chatila, R.; and Herrera, F. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58: 82–115.

Bordt, S.; Finck, M.; Raidl, E.; and von Luxburg, U. 2022. Post-Hoc Explanations Fail to Achieve their Purpose in Adversarial Contexts. *ACM Conference on Fairness, Accountability, and Transparency*, 5.

Buolamwini, J.; and Gebru, T. 2018. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Conference on Fairness, Accountability and Transparency*, 77–91. PMLR.

Carmichael, Z.; and Scheirer, W. J. 2021a. A Framework for Evaluating Post Hoc Feature-Additive Explainers. arXiv:2106.08376.

Carmichael, Z.; and Scheirer, W. J. 2021b. On the Objective Evaluation of Post Hoc Explainers. arXiv:2106.08376.

den Broeck, G. V.; Lykov, A.; Schleich, M.; and Suciu, D. 2021. On the Tractability of SHAP Explanations. In *AAAI Conference on Innovative Applications of Artificial Intelligence, IAAI*, 6505–6513. AAAI Press.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences. Accessed: 2022-05-01.

EU, C. o. t.; and Parliament, E. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, L 119: 1–88.

European Commission. 2021. Proposal for a regulation of the European Parliament and the Council: Laying down harmonised rules on Artificial Intelligence (Artificial Intelligence Act) and amending certain Union legislative acts. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206. Accessed: 2022-05-10.

FICO, F. I. C. 2018. FICO Explainable Machine Learning Challenge: Home Equity Line of Credit (HELOC) Dataset. https://community.fico.com/s/explainable-machine-learning-challenge. Accessed: 2022-05-01.

[2]https://prijatelj.github.io

Garreau, D.; and von Luxburg, U. 2020. Explaining the Explainer: A First Theoretical Analysis of LIME. In Chiappa, S.; and Calandra, R., eds., *International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, 1287–1296. PMLR.

Ghalebikesabi, S.; Ter-Minassian, L.; DiazOrdaz, K.; and Holmes, C. C. 2021a. On locality of local explanation models. *Advances in Neural Information Processing Systems*, 34.

Ghalebikesabi, S.; Ter-Minassian, L.; DiazOrdaz, K.; and Holmes, C. C. 2021b. [OpenReview Discussion] On Locality of Local Explanation Models. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Hendrycks, D.; Zhao, K.; Basart, S.; Steinhardt, J.; and Song, D. 2021. Natural Adversarial Examples. In *Conference on Computer Vision and Pattern Recognition*, 15262–15271. IEEE/Computer Vision Foundation.

Kaur, H.; Nori, H.; Jenkins, S.; Caruana, R.; Wallach, H.; and Wortman Vaughan, J. 2020. Interpreting Interpretability: Understanding Data Scientists' Use of Interpretability Tools for Machine Learning. In Bernhaupt, R.; Mueller, F. F.; Verweij, D.; Andres, J.; McGrenere, J.; Cockburn, A.; Avellino, I.; Goguey, A.; Bjøn, P.; Zhao, S.; Samson, B. P.; and Kocielnik, R., eds., *Conference on Human Factors in Computing Systems*, 1–14. ACM.

Krishna, S.; Han, T.; Gu, A.; Pombra, J.; Jabbari, S.; Wu, S.; and Lakkaraju, H. 2022. The Disagreement Problem in Explainable Machine Learning: A Practitioner's Perspective. arXiv:2202.01602.

Lipton, Z. C. 2018. The Mythos of Model Interpretability. *ACM Queue*, 16(3): 30.

Lundberg, S. M.; and Lee, S.-I. 2017. A Unified Approach to Interpreting Model Predictions. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Conference on Neural Information Processing Systems*, 4765–4774.

McGregor, S. 2021. Preventing Repeated Real World AI Failures by Cataloging Incidents: The AI Incident Database. In *AAAI Conference on Innovative Applications of Artificial Intelligence, IAAI*, 15458–15463. AAAI Press.

Miller, D. D.; and Brown, E. W. 2018. Artificial Intelligence in Medical Practice: The Question to the Answer? *The American Journal of Medicine*, 131(2): 129–133.

Noppel, M.; Peter, L.; and Wressnegger, C. 2022. Backdooring Explainable Machine Learning. arXiv:2204.09498.

O'Neil, C. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown. ISBN 978-0-553-41883-5.

Redmond, M.; and Baveja, A. 2002. A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, 141(3): 660–678.

Ren, K.; Zheng, T.; Qin, Z.; and Liu, X. 2020. Adversarial Attacks and Defenses in Deep Learning. *Engineering*, 6(3): 346–360.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1135–1144. ACM.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215.

Ruff, L.; Kauffmann, J. R.; Vandermeulen, R. A.; Montavon, G.; Samek, W.; Kloft, M.; Dietterich, T. G.; and Müller, K. 2021. A Unifying Review of Deep and Shallow Anomaly Detection. *Proc. IEEE*, 109(5): 756–795.

Schneider, J.; Meske, C.; and Vlachos, M. 2022. Deceptive AI Explanations: Creation and Detection. In Rocha, A. P.; Steels, L.; and van den Herik, H. J., eds., *International Conference on Agents and Artificial Intelligence, ICAART*, 44–55. SciTePress.

Shrotri, A. A.; Narodytska, N.; Ignatiev, A.; Meel, K. S.; Marques-Silva, J.; and Vardi, M. Y. 2022. Constraint-Driven Explanations for Black-Box ML Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8): 8304–8314.

Slack, D.; Hilgard, S.; Jia, E.; Singh, S.; and Lakkaraju, H. 2020. Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods. In Markham, A. N.; Powles, J.; Walsh, T.; and Washington, A. L., eds., *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, 180–186. ACM.

Song, X.; Wu, M.; Jermaine, C. M.; and Ranka, S. 2007. Conditional Anomaly Detection. *IEEE Trans. Knowl. Data Eng.*, 19(5): 631–645.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In Bengio, Y.; and LeCun, Y., eds., *International Conference on Learning Representations*, 1–10.

Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M. K.; and Ristenpart, T. 2016. Stealing Machine Learning Models via Prediction APIs. In *25th USENIX Security Symposium (USENIX Security 16)*, 601–618. Austin, TX: USENIX Association. ISBN 978-1-931971-32-4.

U.S.-EU TTC. 2022. U.S.-EU Joint Statement of the Trade and Technology Council. https://www.commerce.gov/news/press-releases/2022/05/us-eu-joint-statement-trade-and-technology-council. Accessed: 2022-05-10.

Vu, M. N.; Mai, H. Q.; and Thai, M. T. 2022. EMaP: Explainable AI with Manifold-based Perturbations. arXiv:2209.08453.

Zhan, Y.; Zheng, B.; Wang, Q.; Mou, N.; Guo, B.; Li, Q.; Shen, C.; and Wang, C. 2022. Towards Black-Box Adversarial Attacks on Interpretable Deep Learning Systems. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, 1–6.

Zhang, C. A.; Cho, S.; and Vasarhelyi, M. 2022. Explainable Artificial Intelligence (XAI) in auditing. *International Journal of Accounting Information Systems*, 100572.