Future Generation Computer Systems 175 (2026) 108059

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# CANDICE: An explainable and intelligent framework for network intrusion detection

Shuhua Li [a], Ruiying Du [a,*], Jing Chen [a], Kun He [a], Cong Wu [b], Yebo Feng [b]

[a] Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, 430072, China

[b] College of Computing and Data Science, Nanyang Technological University, 639798, Singapore

ARTICLE INFO

*Keywords:*
Network intrusion detection system
Network traffic analysis
Deep learning
Explainable AI (XAI)
Counterfactual explanation
Disentangled representation learning

ABSTRACT

In recent years, Deep Learning-based Network Intrusion Detection System (DL-NIDS) have demonstrated remarkable performance in detecting cyberattacks in network traffic. However, the lack of explainability for DL-NIDSs prevents end-users from trusting and understanding the detection results, thereby limiting their applications in practice. Although several approaches have been proposed to explain DL-NIDS, they run the risk of providing unfaithful explanations. In addition, existing methods merely output a set of important features as explanation, which is insufficient for end-users to thoroughly understand the attack. In this paper, we propose CANDICE, an explainable and intelligent framework for detecting and explaining intrusions in network traffic. Differing from existing works, CANDICE is highlighted by: (i) providing faithful explanation by disentangling the traffic representations and generating counterfactual explanations, and (ii) offering end-users a comprehensive view of the attack by generating an intrusion profile based on the explanation. We conduct experiments on four representative traffic datasets to evaluate the effectiveness of CANDICE. The results demonstrate that CANDICE surpasses existing methods in terms of explanation fidelity, sparsity, stability, and efficiency, while achieving high accuracy of above 96.10% in detecting intrusions.

## 1. Introduction

Network-based Intrusion Detection Systems (NIDS) form the front-line defense in cybersecurity by monitoring and detecting intrusions in network traffic. At present, most of the popular NIDSs are learning-based, which utilize Artificial Intelligence (AI) models to distinguish between benign and malicious traffic [1–3]. Thanks to the great capacity of Deep Neural Networks (DNNs) in capturing complex patterns and subtle deviations [4], Deep Learning-based NIDSs (DL-NIDS) demonstrate remarkable performance in detecting sophisticated attacks and have become the cutting-edge pillar of AI-based security applications [5–7].

Despite the superiority of DL-NIDSs, their inability to explain the detection results has severely hampered their application in practice [8,9]. Most current DL-NIDS simply output a prediction label (e.g., *malicious* or *benign*), without providing any additional information about the detected traffic. As a result, the end-users struggle to trust and understand the decision, as well as take further actions to deal with the alerts. For instance, when an anomaly alert is triggered, the Security Operation Centers (SOCs) are expected to take immediate countermeasures against the attack [10], such as blocking traffic or isolating network segments. In the absence of explanation, the SOCs have to manually inspect and verify the alert, which is both labor-intensive and impractical in large-scale network [11]. This can lead to a situation where the security analysts are overwhelmed with tons of intrusion alarms requiring investigation, resulting in delays in responding to actual threats.

Recent studies have proposed several approaches to explain DL-based security applications [9,12–16]. However, they suffer from two main challenges. *First*, most existing methods explain model results by exploring the contribution of each feature to the decision. They rely on an over-ideal assumption that the model correctly extract knowledge about real traffic patterns [12–14]. However, since the traffic features are highly correlated, DNN models have been revealed to learn a entangled representation [17] in which different traffic patterns are mixed together, making it difficult to clarify the contribution of features. We visualized the significant correlations between features to reveal this phenomenon. (see Section 5.1.1, Fig. 2). As a result, the entangled representation fail to capture the underlying factors of variation in the observed traffic data, leading to spurious explanations. *Second*, Existing methods merely output a set of important features

* Corresponding author.
*E-mail address:* duraying@whu.edu.cn (R. Du).

https://doi.org/10.1016/j.future.2025.108059
Received 24 February 2025; Received in revised form 12 June 2025; Accepted 27 July 2025
Available online 6 August 2025

as explanation, which is insufficient to provide users with a deeper understanding of the anomalies. For instance, beyond identifying the *number of connections* as an important indicator of DDoS attack, security analysts are more concerned to know how often connection attempts within a time window will trigger an intrusion alert. Such insights allow them to appropriately adjust their defense strategy to prevent potential attacks. Unfortunately, existing methods fail to provide such an intuitive perspective for this purpose.

In this work, we address the above challenges by proposing CANDICE, an explainable and intelligent framework for detecting and explaining intrusions in network traffic. CANDICE consists of two main components: the *CANDICE-detector* and the *CANDICE-explainer*. First, the CANDICE-detector extracts disentangled representations from network traffic, in order to better clarify feature contributions and mitigate spurious explanations. To achieve this, we introduce a Variational Autoencoder (VAE)-based disentangled representation learning mechanism into the CANDICE-detector's learning process to train a calibrated detection model. Second, the CANDICE-explainer generates counterfactual explanations to reveal both important features and their specific impacts on detection results. To achieve this, we design and optimize a multi-objective loss function to generate counterfactuals that satisfy security-specific constraints. In addition, we design a human-readable *intrusion profile* as the output of CANDICE framework. The intrusion profile integrates both attack information and explanations, enabling security analysts to thoroughly understand the detected attack.

The main contributions of this work are as follows:

- **Novel explainable intrusion detection framework.** We propose CANDICE, an explainable and intelligent framework for detecting and explaining intrusions in network traffic. By integrating the detector and explainer components into a unified architecture, CANDICE bridges the semantic gap between detection results and explanations, enabling security analysts to better understand and trust the detection results.
- **New explanation method and human-readable threat intelligence.** We design an optimized counterfactual-based explanation method and user-friendly intrusion profile for learning-based NIDS. By disentangling traffic representations and generating counterfactuals, we provide faithful explanations for the detection results. By presenting intrusion profiles that integrate both attack information and explanation, we offer an intuitive and comprehensive view for anomaly understanding, allowing end-users to easily inspect the threats.
- **Comprehensive evaluation.** We conducted experiments on four representative traffic datasets to evaluate the effectiveness of CANDICE. The experimental results show that CANDICE outperforms existing methods in terms of fidelity, sparsity, stability, and efficiency of explanation, while achieving high accuracy of above 96.10% in detecting intrusions.

The rest of this paper is organized as follows: Section 2 introduces the related works on learning-based NIDS and the eXplainable AI (XAI) approaches within the security domain. Section 3 introduce the notations and two key techniques used in this work, i.e., disentangled representation learning and counterfactual explanation. Section 4 describes the details of designing CANDICE. Section 5 presents the experimental results of evaluating CANDICE in terms of intrusion detection and explanation. Section 7 provides the conclusion of this work.

## 2. Related work

In this section, we first briefly overview the AI-based network intrusion detection systems. Then we introduce state-of-the-art explanation methods for interpreting AI-based security applications.

### 2.1. AI-based network intrusion detection system

Existing AI-based NIDS can be categorized into two major groups, i.e., the Machine Learning-based NIDS (ML-NIDS) and the Deep Learning-based NIDS (DL-NIDS) [1]. The ML-based approaches rely on traditional machine learning algorithms such as Support Vector Machine (SVM)and Random Forest (RF) to identify attacks [18–20]. However, due to their limited capability in capturing complex traffic patterns with shallow ML models, they are inadequate for detecting sophisticated attacks. In contrast, DL-based approaches utilize deep neural networks to automatically extract traffic patterns and demonstrate remarkable performance in network intrusion detection [6,7,21–24]. The key to the success of DL-NIDS is the traffic representations learned by the DNN model. Among these, Auto-Encoder (AE) and its variants have been widely used to construct effective NIDS due to its strong capability in traffic representation learning [6,7,22]. Despite the benefits of DL-NIDS, they lack the ability to explain to users what they have detected and why they made the decision. Departing from existing NIDS methods, our CANDICE framework addresses this limitation by proposing a novel explainable intrusion detection framework. In addition to detecting attacks, it generates explanations for the detection results, as well as outputs a human-readable intrusion profile for end-users to better understand the model decision.

### 2.2. XAI methods for cybersecurity

XAI techniques can be categorized into different groups depending on their explanation scope and objective. For example, ante-hoc explanation methods builds self-interpretable models based on intrinsically transparent learning algorithms (e.g., Naive Bayes and Decision Tree [25]), whereas post-hoc methods design additive explainers for pre-trained models [26,27]. In addition, local explanation focus on explaining individual decisions/outputs made by the model [26,28,29], while global explanation aims to understand the overall model behavior [30–32]. In this work, we focus on the most prevalent form of explaining NIDSs [9,16], i.e., post-hoc local explanation methods, which can be divided into four categories:

**Approximation-based methods.** These approaches utilize an interpretable surrogate agent to locally approximate the decision boundary of a black-box DL-NIDS, then retrieve explanations from the agent model [26]. LEMNA [12] first proposes to use a non-linear mixture regression model to approximate RNN-based security applications [33,34] and leverages fused Lasso [35] to cope with the feature dependency in RNN. xNIDS [13] explains DL-NIDS by approximating and sampling around history inputs, and capturing feature dependency using sparse group lasso. However, since approximation-based methods cannot guarantee to accurately cross the decision boundary of the model, they run the risk of providing unfaithful explanations for security applications.

**Perturbation-based methods.** These approaches explain model decisions by perturbing the corresponding input samples and observing model sensitivity to the perturbations [29,36]. They identify the features that most influence the model's output as the explanation. CADE [14] detects and explains concept drift samples in DL-based security applications by perturbing the input and observing the distance changes between the sample and its nearest class. However, CADE focuses on explaining concept drift in the security domain rather than intrusion detection, which fails to satisfy specific constraints when explaining the detection results of DL-NIDS, such as the sparsity and stability of explanation.

**Gradient-based methods.** These approaches leverage the gradient information from DNN model to measure the sensitivity of features [37–39]. As they rely on access to model gradients, they are categorized as white-box methods, as opposed to approximation-based and perturbation-based methods. DeepAID [15] is specially designed for explaining DL-based anomaly detection systems, which is based on

**Table 1**
Overall comparison of representative explanation methods.

| Explainers | Design aspects | | | Properties | | | |
|---|---|---|---|---|---|---|---|
| | Strategies[a] | Support Black-box | FA/Example-based | Fidelity[b] | Sparsity[b] | Stability[b] | Efficiency[b] |
| LEMNA [12] | Approx. | ✓ | FA | ◐ | ○ | ○ | ○ |
| xNIDS [13] | Approx. | ✓ | FA | ◐ | ◐ | ◐ | ◐ |
| SHAP [27] | SHAP-based | ✓ | FA | ◐ | ○ | ◐ | ○ |
| DeepAID [15] | Grad. | ✗ | Example | ● | ◐ | ● | ● |
| CADE [14] | Perturb. | ✗ | Example | ◐ | ○ | ○ | ○ |
| **Ours** | Perturb. | ✓ | Example | ● | ● | ● | ● |

(✓= support,✗= not support, ●= true, ◐= partially true, ○= false).
[a] This indicates the category of explanation methods introduced in Section 2.2, i.e., the Approximation-based, Perturbation-based, Gradient-based, and SHAP-based methods.
[b] This indicates the four criteria for evaluating explanation methods, as measured through the experiments in Section 5.4.

back-propagation to retrieve explanations. Unfortunately, the access to the model's internal details required by white-box methods is not always feasible in reality.

**SHAP-based methods.** This category refers to approaches that employ SHapley Additive exPlanations (SHAP) [27] to attribute NIDS outputs to important input features. As a representative Feature Attribution (FA)-based explanation method, SHAP utilizes the Shapley value [40] from cooperative game theory to fairly quantify each feature's contribution to model decisions. Based on different ways to approximate the Shapley value, SHAP includes three main variants: DeepSHAP and TreeSHAP, which are specifically used for explaining DNN-based and tree-based security applications [41–43], and KernelSHAP, a model-agnostic approach that can be applied to diverse ML/DL models [44–47]. Despite the widespread adoption in explaining learning-based security applications, SHAP-based methods are criticized for their high computational cost, especially in high-dimensional feature space [8].

Table 1 summarizes the explanation methods mentioned above. Note that the SHAP in Table 1 and Section 5 refers to KernelSHAP. We include KernelSHAP in the comparison baselines, considering its model-agnostic property that can be used to explain different types of AI-based security applications. Our CANDICE-explainer falls into the category of perturbation-based explanation method, in which we change the input features to generate counterfactual explanations. However, CANDICE-explainer differs from existing works in two aspects: (1) we are the first work that calibrates the model's internal behavior by disentangling traffic representations to facilitate model explainability; (2) we provide black-box explanation by generating traffic counterfactuals that explicitly cross the local decision boundary of the model, thereby ensuring the faithfulness of explanation.

## 3. Preliminaries

In this section, we first outline the key notations and definitions used throughout this paper. Then we introduce two main techniques used in our CANDICE framework, namely *VAE-based Disentangled Representation Learning* and *Counterfactual Explanation*.

### 3.1. Notations

Let $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ be a traffic sample in network traffic dataset $\mathcal{D}$, where $\{x_i\}_{i=1}^n$ represents the traffic features. The DL-NIDS $f(\cdot)$ takes $x$ as input and outputs the prediction label $y_{pred} = f(x)$. We denote the latent representation extracted from $x$ by $z = (z_1, z_2, \ldots, z_m) \in \mathbb{R}^m$. The counterfactual traffic sample is defined as $x_{cf}$, which has a different prediction label $y_{cf} \neq y_{pred}$.

### 3.2. VAE-based disentangled representation learning

Variational Auto-Encoder (VAE) [48] is a type of deep generative model that combines the idea of Auto-Encoder and variational inference. VAE consists of an encoder neural network $q_\phi(z|x)$, which maps the input data $x$ into a low-dimensional latent space, and a decoder neural network $p_\theta(x|z)$, which reconstructs data $x$ from latent variable $z$. The optimization objective of VAE is to maximize the Evidence Lower Bound (ELBO):

$$\mathcal{L}(\theta, \phi; x, z) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}\left(q_\phi(z|x) \parallel p_\theta(z)\right), \tag{1}$$

where the first term $\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right]$ represents the conditional log likelihood in charge of reconstruction quality. The second term is the Kullback–Leibler (KL) divergence, which encourages the approximate posterior distribution $q_\phi(z|x)$ to be close to the prior $p_\theta(z)$. The detailed derivation of VAE can be found in the original paper [48].

In VAE, the prior distribution $p_\theta(z)$ is generally chosen as a standard Gaussian distribution $\mathcal{N}(0, 1)$, which allows the KL term to impose independent constraints on the learned representations and thereby leads to the disentanglement [49]. However, the vanilla VAE has proven to be insufficient dealing with complex dataset [50]. To achieve better disentanglement performance, researchers have proposed various regularizers combined with the original VAE loss function, resulting in the family of VAE-based disentanglement methods [49–53]. In this work, we utilize $\beta$-TCVAE to achieve the disentanglement of traffic representation. The details are described in Section 4.2.

### 3.3. Counterfactual explanation

Wachter et al. [54] first proposed the concept of Counterfactual Explanations (CE) for explaining black-box AI models. They formulate CE generation as an optimization problem, which aims to find the minimal changes of inputs that can reverse the model's decision. Eq. (2) illustrates the optimization objective for computing the counterfactual example $x_{cf}$ for an input $x$:

$$\arg\min_{x_{cf}} \max_{\lambda} \lambda(f(x_{cf}) - y_{cf})^2 + d(x, x_{cf}). \tag{2}$$

The first term ensures that the model's prediction $f(x_{cf})$ to be close to the desired label $y_{cf}$, where $y_{cf} \neq f(x)$. The second term measures the distance between $x_{cf}$ and $x$, which is minimized to ensure the generated counterfactual remains as close as possible to the original input. Later studies attempt to generate counterfactuals with desirable properties by imposing restrictions on the objective function [55–57]. In this work, our goal is to generate counterfactuals for network traffic that satisfy constraints within security domain to explain DL-NIDS, i.e., validity, feasibility, and sparsity. The design of the objective function is described in Section 4.3.

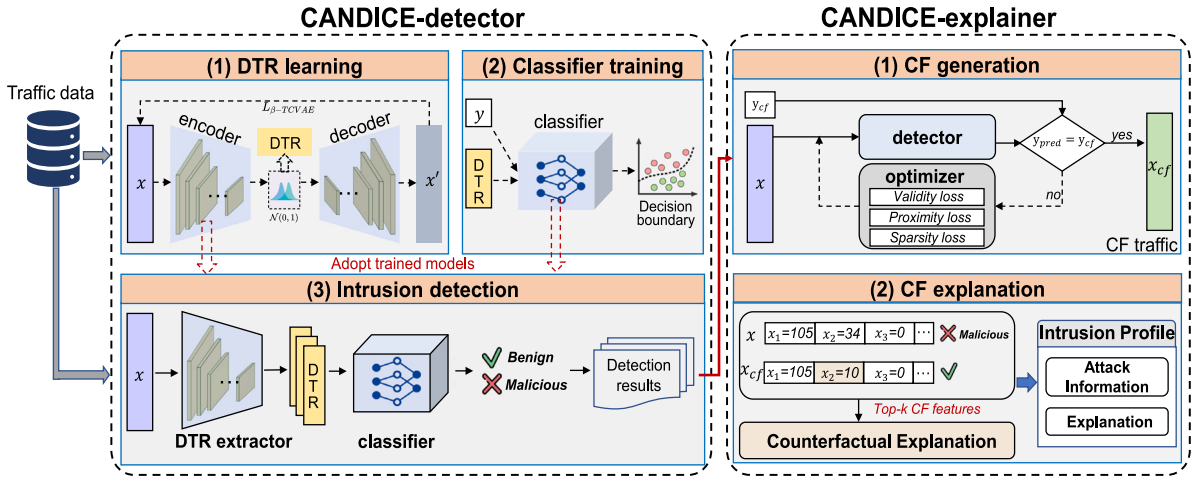



**Fig. 1.** Overview of the CANDICE framework.

## 4. The CANDICE framework

### 4.1. Overview

Fig. 1 depicts the framework of CANDICE, which consists of two key components: the *CANDICE-detector* and the *CANDICE-explainer*. The CANDICE-detector consists of a calibrated feature extractor and a classifier, which extracts disentangled representations from input traffic and detects intrusions. The detection results are further sent to the CANDICE-explainer for explanation. The CANDICE-explainer generates counterfactual traffic by optimizing a multi-objective loss function especially designed for explaining security applications. The explanations are derived by comparing the differences between original traffic and the counterfactuals. At the end, CANDICE outputs an *Intrusion Profile* containing both the attack information and explanation, enabling end-users to gain a deeper understanding of the attack.

### 4.2. CANDICE-detector

As illustrated in Fig. 1, the CANDICE-detector operates in three phases: the Disentangled Traffic Representation (DTR) Learning phase, the Classifier Training phase, and the Intrusion Detection phase.

#### 4.2.1. DTR learning phase

As discussed in Section 1, the entangled traffic representation learned by the NIDS model hinders the clarification of feature contributions and leads to spurious explanation. To address this problem, we integrate a Disentangled Representation Learning (DRL) mechanism into CANDICE-detector's learning process. The DRL mechanism aims to identify and isolate the underlying factors of variation in network traffic data, mapping them into a structured latent space where each dimension corresponds to a distinct aspect of traffic behavior (e.g., temporal or spatial traffic patterns). By disentangling traffic features into independent factors, CANDICE-detector can better identify the root causes of intrusions, thereby facilitating more accurate detection and faithful explanation.

We utilize $\beta$-TCVAE [49] to achieve traffic representation disentanglement. $\beta$-TCVAE follows the same encoder–decoder architecture as VAE [48], while additionally introduces a hyperparameter $\beta$ to control the degree of disentanglement. As illustrated in Fig. 1, the encoder $q_\phi(z|x)$ first encodes the input traffic data $x$ into a low-dimensional disentangled latent space, where each dimension of the latent variable $z$ captures a distinct factor of variation of $x$. Then, the decoder $p_\theta(x|z)$ reconstructs $x'$ from $z$. $\beta$-TCVAE achieves disentanglement by decomposing the KL divergence into three terms and separately penalizing them, i.e., the Mutual Information (MI) term, the Total Correlation (TC) term, and the dimension-wise KL divergence term. Eq. (3) shows the objective function of $\beta$-TCVAE:

$$\mathcal{L}(\theta,\phi;\alpha,\beta,\gamma;x,z) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \alpha I_q(z;x) - \beta D_{KL}\left(q_\phi(z) \parallel \prod_j q_\phi(z_j)\right) - \gamma \sum_j D_{KL}\left(q_\phi(z_j) \parallel p_\theta(z_j)\right). \quad (3)$$

As illustrated in Eq. (3), the MI term measures the mutual information between latent representation $z$ and data $x$, which controls how much information of $x$ is captured by $z$. By penalizing the MI term with $\alpha$, the model is encouraged to learn more compact and disentangled representations. The TC term measures the dependence between the dimensions of latent variable $z$, which penalizing $\beta$ encourages the model to find statistically independent factors in the data distribution, thus leading to a more disentangled representation. Finally, the third term weighted by $\gamma$ is used to ensure that each latent dimension individually adheres to the prior distribution.

#### 4.2.2. Classifier training phase

In this phase, the disentangled traffic representations are utilized to train the traffic classifier. We develop three types of classifiers for the CANDICE-detector, respectively based on Support Vector Machine (SVM), Random Forest (RF), and Multi-Layer Perceptron (MLP). We use the RBF kernel for CANDICE-SVM and set the number of trees to 100 for CANDICE-RF. The MLP classifier is implemented as a two-layer architecture, with each layer containing 8 nodes.

#### 4.2.3. Intrusion detection phase

After the pre-training is complete, the encoder of $\beta$-TCVAE is adopted as the feature extractor of CANDICE-detector, followed by the trained classifier. During the intrusion detection phase, CANDICE-detector continuously monitors the incoming network traffic, extracts disentangled traffic representations, and identifies attacks within the traffic. By calibrating the representation learning process, CANDICE-detector is able to detect intrusions in a more interpretable latent space, thereby improving the explainability of the entire system.

### 4.3. CANDICE-explainer

Once the CANDICE-detector detects an attack, the detection result and corresponding traffic sample are sent to the CANDICE-explainer to generate Counterfactual Explanations (CFs). The CANDICE-explainer operates in two phase: the CF Generation phase and the CF Explanation phase.

#### 4.3.1. CF generation phase

Given the detection result and traffic sample to-be-explained, we first define the *Counterfactual Traffic* as follows:

**Definition 1.** *(Counterfactual Traffic). Let $x \in \mathbb{R}^n$ be a traffic sample in dataset $\mathcal{D}$ and $y_{pred} = f(x)$ be the label predicted by DL-NIDS model $f(\cdot)$. The counterfactual traffic $x_{cf}$ is a modified version of $x$ with target label $y_{cf}$, where $y_{cf} \neq f(x)$.*

For example, if the original traffic $x$ is identified as *malicious* by the CANDICE-detector, the goal of CANDICE-explainer is to generate CF traffic $x_{cf}$ with a prediction label of *benign*. We design a multi-objective loss function to generate CF traffic that satisfy three constraints within the security domain, i.e., *validity, feasibility, and sparsity*.

**Validity.** This property ensures that $x_{cf}$ has a prediction label $y_{cf}$, which is opposed to the label $y_{pred}$ of input $x$. It enforces the CF traffic explicitly moves across the local decision boundary of underlying NIDS model from the *malicious* side to the *benign* side, thus providing faithful explanation. This is particularly crucial for security applications, as unfaithful explanations would result in misunderstanding to the detected intrusion and inappropriate response. To address this problem, we use the Binary Cross-Entropy loss to enforce $x_{cf}$ obtains the target prediction $y_{cf}$. The *validity loss*, denoted as $\mathcal{L}_{\text{Val}}$, is defined as follows:

$$\mathcal{L}_{\text{Val}} = y_{cf} \cdot \log(y_{pred}) + (1 - y_{cf}) \cdot \log(1 - y_{pred}). \tag{4}$$

**Feasibility.** This property requires that $x_{cf}$ should be a plausible instance that adheres to the characteristics of real-world traffic and could be observed in actual network environments. It helps to maintain the semantic meaning of CF traffic, thus facilitating realistic and actionable explanation. This is particularly important in security domain, where explanations might be utilized to design countermeasures against the attack, and impractical explanations could have a negative impact on developing effective and efficient defense. We take two steps to ensure the feasibility of CF traffic. First, we incorporate domain knowledge of real-world traffic into the CF generation process by leveraging the historical traffic data. Specifically, we divide the traffic features into mutable features (e.g., flow bytes) and immutable features (e.g., protocol type), and change only the mutable features when generating counterfactual traffic. In addition, we define the permitted min–max ranges for each features to limit their variation within a reasonable range. This prevents generating implausible counterfactuals, such as the feature of packet size lower than zero. Second, we introduce a *proximity loss* denoted as $\mathcal{L}_{\text{Prox}}$ to encourage the CF traffic $x_{cf}$ to be close to the original traffic $x$. We employ the Mahalanobis Distance (MD) to measure the proximity, given its ability to handle the correlations among traffic features and in-distribution deviation. Here, $C \in \mathbb{R}^{D \times D}$ represents the inverse of the covariance matrix of the traffic dataset:

$$\mathcal{L}_{\text{Prox}} = \sqrt{(x - x_{cf})^T C^{-1} (x - x_{cf})}. \tag{5}$$

**Sparsity.** This property requires that the original traffic $x$ should modify as few features as possible to yield the CF traffic $x_{cf}$. It helps to pinpoint the most important features that lead to the decision, thus contributes to concise and non-trivial explanation. Sparse explanations are especially critical for security applications, as dense changes in traffic features would obscure the underlying rationale behind the detection, leading to confusion in understanding decisions and overwhelming the SOCs in inspecting explanations. To achieve sparsity, we introduce a *sparsity loss* in CF generation process based on the $\ell_1$ Norm, which is denoted as $\mathcal{L}_{\text{Spars}}$:

$$\mathcal{L}_{\text{Spars}} = \|x - x_{cf}\|_1. \tag{6}$$

Overall, the multi-objective loss function for generating counterfactual traffic is illustrated in Eq. (7), where $\lambda_1$ and $\lambda_2$ are the hyperparameters controlling the proximity and sparsity properties:

$$\mathcal{L} = \mathcal{L}_{\text{Val}} + \lambda_1 \cdot \mathcal{L}_{\text{Prox}} + \lambda_2 \cdot \mathcal{L}_{\text{Spars}}. \tag{7}$$

#### 4.3.2. CF explanation phase

After generating CF traffic, CANDICE-explainer derives CF explanations by: (1) comparing the differences between original traffic and its counterfactuals; (2) ranking the $top-k$ traffic features as the explanation. The $top-k$ features are ranked based on their importance to model decision-making. For example, the approximation-based approaches LEMNA [12] and xNIDS [13] assigned importance scores to input traffic features by examining the coefficients of the surrogate model. However, since counterfactuals are primarily focused on modifying features to change the predictions, they are unable to directly calculate the importance scores of features. Inspired by [58], we solve this problem by generating multiple counterfactuals for a single traffic example and calculating the change frequency of each counterfactual feature as its importance score. The intuition is that a feature that changes more frequently when generating counterfactual traffic examples is likely to be more important. Specifically, we generate $d$ counterfactuals for a $n$-dimensional traffic example using our CF traffic generation method. Then, we construct a change matrix $M$, where $M_{i,j}$ indicates whether the $j$th traffic feature $x_j$ in the $i$th counterfactual example $x_{cf,i}$ has changed:

$$M_{ij} = \begin{cases} 1 & \text{if } x_j \neq x_{cf,i,j} \\ 0 & \text{if } x_j = x_{cf,i,j}. \end{cases} \tag{8}$$

The feature importance of $x_j$ is then calculated by:

$$FI_j = \frac{1}{d} \sum_{i=1}^{d} M_{ij}. \tag{9}$$

A higher value of change frequency indicates that the corresponding feature frequently occurs in different counterfactuals, suggesting that the model strongly relies on that feature for prediction. In our experiments in Section 5, we set $d = 5$ and rank the $top-10$ counterfactual features to obtain the explanation.

CANDICE-explainer finally outputs an *Intrusion Profile* that provides detailed description of the detected attack. It consists of two main parts: *Attack Information* and *Explanation*. The attack information field is a tuple consisting of five elements:

- *Entity*: The identifier of the attacker and the victim, including IP address, MAC address, ports, protocols, etc.
- *Attack Type*: The category of attack identified by the CANDICE-detector, such as DoS and Probing.
- *Attack Description*: The coarse-grained description of the attack methods and targets.
- *Detection Time*: The timestamp when the attack is detected by CANDICE-detector.
- *Detection Confidence*: The CANDICE-detector's confidence score in identifying the attack.

In addition, the explanation field contains two elements:

- *Key Features*: The $top-k$ traffic features and their difference derived from counterfactual explanation, which indicate the key evidence of detection results.
- *Anomaly Understanding*: The fine-grained description of attack behavior based on counterfactual analysis, integrating information provided by the $top-k$ counterfactual features.

## 5. Experiments

In this section, we first outline the experimental setup and present two usage examples to demonstrate how CANDICE detects and explains intrusions in network traffic. Then we illustrate the experimental results for evaluating: (1) the intrusion detection performance of CANDICE-detector, and (2) the explanation performance of CANDICE-explainer.

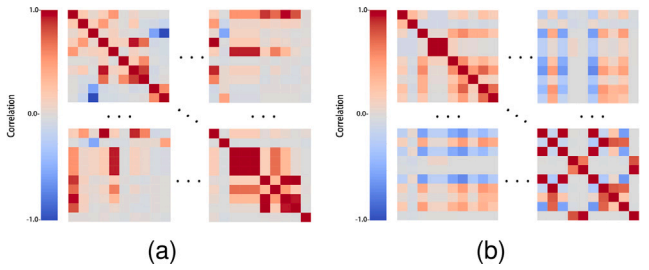



**Fig. 2.** Correlation heatmaps of traffic features of two attacks. (a) DoS attack from CIC-IDS2017-improved dataset. (b) Mirai attack from CIC-IoT2023 dataset.

### 5.1. Experimental setup

#### 5.1.1. Datasets

We conduct the experiments on four representative traffic datasets used for network intrusion detection. As shown in Table 2, these datasets involve diverse network environments and attack types, allowing for a comprehensive evaluation of CANDICE. The details of the datasets are as follows:

- CIC-IoT2023 [59]: This dataset is collected from real IoT devices and networks, with traffic features extracted using the CICFlowMeter tool developed by the Canadian Institute for Cybersecurity. We select four attacks in our experiments: DDoS-SlowLoris, Mirai-udpplain, Recon-Host Discovery, and DNS-Spoofing, with each type comprising 15000 samples represented by 46 traffic features.
- CIC-IDS2017-improved [60]: This dataset is a refined version of the CIC-IDS2017 [61] dataset, which is collected over 5 days of real-world network traffic. We select 1% of benign traffic and 10% of malicious traffic from Portscan and DoS-Hulk attacks, with each sample consisting of 72 traffic features.
- CIC-DoHBrw-2020 [62]: This dataset focuses on the emerging DoH tunneling threats introduced by the DNS over HTTPS protocol, such as Command-and-Control (C2) communication and data exfiltration. Each traffic sample is labeled as benign or malicious, consisting of 28 traffic features extracted by the DoHlyzer tool.
- NSL-KDD [63]: This is a typical dataset that has been widely used for network intrusion detection. We use the DoS and Probing attacks for the experiments, with each traffic sample containing 41 traffic features.

To obtain deeper insights into the data used in our experiments, we perform Exploratory Data Analysis (EDA) on the above datasets. Fig. 2 shows the correlation heatmaps of traffic features for two attacks, i.e., DoS attack from CIC-IDS2017-improved dataset and Mirai attack from CIC-IoT2023 dataset. We find that the traffic features of the two attacks are highly correlated, indicating the existence of traffic pattern entanglement as introduced in Section 1. This observation further supports our motivation, as well as the rationality of the chosen dataset for experiments. Each dataset is split into training and testing sets at a ratio of 80%:20%. The training set is further split into different training and validation subsets via 5-fold cross-validation, as illustrated in Section 5.1.2. The testing set is held-out during training, in order to ensure unbiased evaluation of model performance. Note that the traffic features in the datasets need to be further normalized and encoded. To avoid data leakage, we only apply One-Hot Encoding to categorical features and StandardScaler to continuous features in the training data to build the data preprocessor.

#### 5.1.2. CANDICE implementation

For the CANDICE-detector, we implement a symmetric encoder–decoder using MLP neural network with hidden layers of dimensions 128-64-32-16-8. The dimensions of input and output layers are adjusted based on the numbers of features and classes of each dataset. Each hidden layer uses the ReLU activation function, and the Adam optimizer is employed with a learning rate of 0.0001 during training. The model is trained for a maximum of 100 epochs with a batch size of 256. As mentioned in Section 4.2, three types of CANDICE-detectors are deployed in the experiments, referred to respectively as CANDICE-SVM, CANDICE-RF, and CANDICE-MLP. We use the RBF kernel for CANDICE-SVM and set the number of trees to 100 for CANDICE-RF. The MLP classifier is implemented as a two-layer architecture, with each layer containing 8 nodes. To ensure robust model training, we performed 5-fold cross-validation with an early-stopping mechanism to prevent overfitting. For the CANDICE-explainer, we utilize the Adam optimizer to optimize counterfactual generation over 500 iterations. Five counterfactuals are generated for each to-be-explained traffic instance to assess the importance scores of features. As soon as a valid counterfactual is generated, an early stopping mechanism is applied to finish the optimization process.

For hyperparameter tuning, we utilize Bayesian Optimization [64] to find the optimal hyperparameters of CANDICE. There are two groups of hyperparameters that are configurable in CANDICE framework. For the CANDICE-detector, we set the dimension of disentangled traffic representation to 8 and $\beta = 2$ in Eq. (3). For the CANDICE-explainer, we set $\lambda_1 = \lambda_2 = 0.5$ for the proximity loss and sparsity loss in Eq. (7).

#### 5.1.3. Baselines and metrics

For evaluating the detection performance of CANDICE-detectors, we construct a standard Vanilla AutoEncoder (Vanilla-AE) [65] as the baseline. The difference between Vanilla-AE and CANDICE-detector is that Vanilla-AE does not perform disentangled representation learning, which is used to evaluate the effectiveness of our method. The Vanilla-AE was configured to have the same architecture as CANDICE-MLP and trained on the same datasets listed in Table 2. Four commonly used metrics in intrusion detection are evaluated in this experiment, including Accuracy, Recall, F1-score, and False Positive Rate (FPR).

For evaluating the explanation performance of CANDICE-explainer, we use the explanation methods listed in Table 1 as the baselines for comparison. The details of these methods are described in Section 2. We implement them following the configurations in original works [12–15]. The fidelity, sparsity, stability, and efficiency of explanation are evaluated under the three CANDICE-detectors and Vanilla-AE. The definitions and calculation methods for these metrics are described in Section 5.4.

### 5.2. Examples: detecting and explaining network intrusions with CANDICE

We present two examples to illustrate the usage of CANDICE in detecting and explaining network attacks. The first example involves a DoH tunneling attack from the CIC-DoHBrw-2020 dataset [62], while the second one is a DoS-TCP-flood attack in IoT environment [59].

The traffic are first fed into the CANDICE-detector, which outputs the predicted attack type and the confidence score of the detection. We use CANDICE-MLP as the underlying detector for both examples. Then, the CANDICE-explainer is activated to explain the detection results. It outputs traffic features that contribute most to the decision as explanation, as well as the difference in feature values between the original traffic and counterfactual traffic. The description of the features are illustrated in Table 3. Here, we showcase the $top-3$ explanatory features for simplicity, where ↑ indicates that the feature value of the original traffic is larger than the counterfactual traffic, and ↓ the opposite. Finally, CANDICE integrates attack information and explanation into a human-readable intrusion profile. Note that all host information (e.g., IP addresses) is virtualized and not related to any actual network or device.

**Table 2**
Datasets used in the experiments.

| Datasets | Attack types | Traces | Features |
|---|---|---|---|
| CIC-IoT2023 [59] | DDoS-SlowLoris, Mirai-udpplain, Recon-Host Discovery, DNS-Spoofing, Benign | 75,000 | 46 |
| CIC-IDS2017-improved [60] | DoS-Hulk, PortScan, Benign | 47,622 | 72 |
| CIC-DoHBrw-2020 [62] | Benign-DoH, Malicious-DoH | 39,000 | 28 |
| NSL-KDD [63] | DoS, Probing, Benign | 42,762 | 41 |

**Table 3**
Feature description of the two examples.

| Attack | Feature | Description |
|---|---|---|
| DoH Tunneling | Duration | Time between first and last packet received in the flow |
| | FlowBytesSent | Number of flow bytes sent |
| | PacketLength Variance | Variance of Packet Length |
| DoS-TCP-flood | Flow_duration | Time between first and last packet received in the flow |
| | Header_length | Length of packet header in bits |
| | AVG | Average packet length in the flow |

#### 5.2.1. Example 1

Fig. 3(a) shows the intrusion profile of a DoH tunneling attack example. The CANDICE-detector successfully identifies the attack with a confidence score of 0.87. The $top-3$ key explanatory features pinpointed by CANDICE-explainer are *Duration*, *FlowBytesSent* and *PacketLengthVariance*. The sharp increase in Feature 1 indicates that the detected traffic lasts abnormally longer compared to normal DNS lookups, suggesting that the connection is possibly being exploited as a covert channel. Feature 2 further confirms the presence of anomalous data transmission, whereby bytes sent by the host is significantly larger than standard DNS queries. Feature 3 implies a large variance of packet lengths within a traffic flow, which is also anomalous because legitimate domains generally have similar lengths and result in smaller variance.

#### 5.2.2. Example 2

Fig. 3(b) shows the intrusion profile of a DoS-TCP-flood attack example in IoT environment, which is successfully detected by the CANDICE-detector with a confidence score of 0.90. The $top-3$ key explanatory features identified by CANDICE-explainer are *Flow_duration*, *Header_length*, and *AVG*. As opposite to Example 1, the decrease in Feature 1 of the DoS-TCP-flood sample indicates a significantly shorter flow duration compared to normal connections. This traffic pattern aligns with typical DoS attack behavior, where the attacker sends short-lived spoofed requests at a high frequency instead of maintaining a relatively long-standing connection. Furthermore, we examined the original dataset and observed that DoS-TCP-flood samples exhibit smaller packet headers and highly consistent within-flow packet sizes compared to benign traffic, which supports our explanations of Features 2 and Feature 3. This pattern can be attributed to DoS-TCP-flood attack uses minimal-sized spoofed packets to establish TCP connections, in contrast to legitimate traffic that carries actual payload data and thus exhibits significant variations in packet lengths.

These two examples demonstrate that the proposed CANDICE framework can accurately detect intrusions and provide faithful explanations. By analyzing the fine-grained explanation of the detected attack in intrusion profile, the SOCs can more effectively understand the attack behavior and develop countermeasures, e.g., configuring filtering rules based on the counterfactual features.

### 5.3. Evaluation of CANDICE-detector

#### 5.3.1. Intrusion detection performance

The comparison results of the three CANDICE-detectors and Vanilla-AE are shown in Fig. 4. Compared to Vanilla-AE, the detectors demonstrate superior intrusion detection performance across all four datasets. Specifically, CANDICE-SVM achieves the highest detection accuracy of 98.95% on the CIC-IoT2023 dataset, showing an increase of 3.29% over Vanilla-AE. The results on CIC-IDS2017-improved dataset further demonstrate the superiority of CANDICE, which outperforms Vanilla-AE with up to 5.68%, 3.60%, and 5.64% improvements in accuracy, recall, and F1-score, respectively. As for the CIC-DoHBrw-2020 dataset, although Vanilla-AE is slightly superior with the lowest FPR of 2.31%, it underperforms our CANDICE-detectors on the other three metrics. For the NSL-KDD dataset, all four detectors demonstrate comparable detection performance, while CANDICE-RF stands out with the lowest FPR of 1.13%. The experimental results indicate that our CANDICE-detectors can effectively identify network intrusions with a high accuracy and low false positive rate. To verify that the observed improvements are statistically meaningful, we further performed paired t-test [66] on the 5-fold cross-validation results of each dataset. All tests comparing the CANDICE-detectors against the baseline Vanilla-AE yielded statistically significant results ($p < 0.05$), confirming the superiority of our method. We attribute this superiority to the gains of traffic representation disentanglement, which extracts intrinsic and interpretable behavioral patterns from complex traffic inputs, enabling more accurate intrusion detection.

**Table 4**
Comparison results of generalization performance on CIC-IDS2017-improved dataset.

| Detectors | Accuracy | Recall | F1-score | FPR |
|---|---|---|---|---|
| CANDICE-SVM | 94.55 ± 0.53 | 93.73 ± 0.81 | 94.27 ± 0.73 | 3.64 ± 0.71 |
| CANDICE-RF | 92.27 ± 1.02 | 92.13 ± 0.92 | 92.21 ± 0.87 | 3.47 ± 0.51 |
| CANDICE-MLP | 94.67 ± 0.47 | 93.87 ± 0.56 | 94.56 ± 0.54 | 3.16 ± 0.40 |
| Vanilla-AE | 91.64 ± 1.08 | 90.57 ± 1.43 | 91.43 ± 1.12 | 3.83 ± 0.81 |

#### 5.3.2. Generalization performance

Beyond detecting known attacks, we evaluate CANDICE-detector's ability to generalize to unseen attacks. We conduct experiments on the CIC-IDS2017-improved dataset using a leave-one-class-out strategy to simulate a real-world zero-day attack scenario. Specifically, we iteratively pick one attack type as the *previously unseen* attack by discarding the corresponding samples in the training set and identifying them in the testing set. Then, we train the detectors on the remaining training data and observe whether they can identify the unknown traffic as malicious. We repeat this process for each attack type and report the mean and standard deviation in Table 4. The results demonstrate that the CANDICE-detectors consistently outperform Vanilla-AE across all metrics. Specifically, CANDICE-MLP achieves the best generalization performance among all detectors. It shows the highest accuracy (94.67% ± 0.47) and recall (93.87% ± 0.56), with improvements of up to 3.03% and 3.30% compared to Vanilla-AE, respectively. CANDICE-MLP also achieves the best F1-score (94.56% ± 0.54), indicating its superiority in effectively detecting unknown attacks while maintaining low false positives. The results further validate the effectiveness of the

**Intrusion Profile 1: Encrypted DNS Tunneling Attack**

**Attack Information**

**Entity**: <src_ip = 203.0.113.12, dst_ip = 192.0.2.13, src_port = 1910, dst_port = 443, protocol = tcp >

**Attack Type**: DoH Tunneling

**Attack Description**: The *DNS Tunneling* attack attempts to covertly exfiltrate data or establish C2 communications by encapsulating data into encrypted DNS queries and responses.

**Detection Time**: 2025-01-28 15:32:10

**Detection Confidence**: 0.87

**Explanation**

**Key Features**:
$\Delta_{f1} = (Duration, \uparrow 112.86)$,
$\Delta_{f2} = (FlowBytesSent, \uparrow 37169)$,
$\Delta_{f3} = (PacketLengthVariance, \uparrow 160028.71)$

**Anomaly Understanding**: ❶ Prolonged connection; ❷ Large volume of transmitted data; ❸ Varied greatly in packet lengths.

**Intrusion Profile 2: DoS Attack**

**Attack Information**

**Entity**: <src_ip = 203.0.113.14, dst_ip = 192.0.2.18, protocol = tcp >

**Attack Type**: DoS-TCP-flood

**Attack Description**: The *DoS-TCP-flood* attack attempts to overwhelm the target systems and disrupt services to legitimate users by sending large volumes of TCP connection requests.

**Detection Time**: 2025-01-28 09:14:21

**Detection Confidence**: 0.90

**Explanation**

**Key Features**:
$\Delta_{f1} = (Flow\_duration, \downarrow 15.66)$,
$\Delta_{f2} = (Header\_length, \downarrow 75946)$,
$\Delta_{f3} = (AVG, \downarrow 70)$

**Anomaly Understanding**: ❶ Large volume of short-lived TCP connections; ❷ Abnormally small and consistent packet sizes within the flow.

**Fig. 3.** Examples of CANDICE usage. (a) Intrusion profile of DoH Tunneling attack. (b) Intrusion profile of DoS-TCP-flood attack.

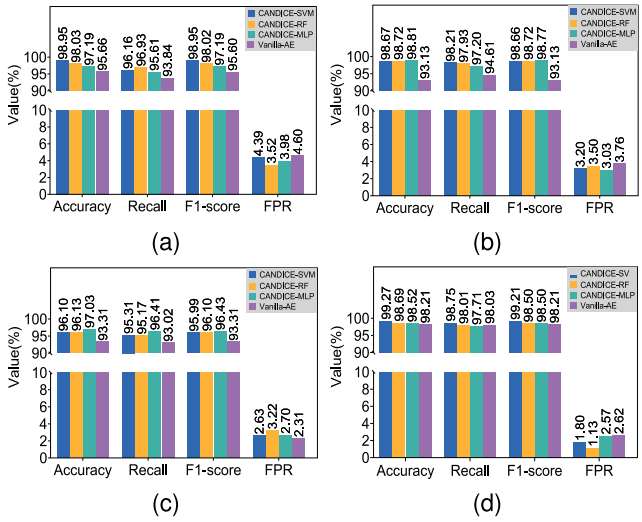



**Fig. 4.** Comparison results of intrusion detection performance on four traffic datasets. (a) CIC-IoT2023 dataset. (b) CIC-IDS2017-improved dataset. (c) CIC-DoHBrw-2020 dataset. (d) NSL-KDD dataset.

DTR learning in CANDICE design, as it excels at capturing the essential behavioral difference between benign and malicious traffic. This core ability allows the CANDICE-detector to recognize novel deviations from normal traffic patterns when an unseen attack emerges, thereby enabling its superior generalization capability.

### 5.4. Evaluation of CANDICE-explainer

#### 5.4.1. Explanation fidelity

This criterion evaluates whether the explainer accurately captures the relevant features of a prediction. We adopt the Descriptive Accuracy (DA) [8] to assess the fidelity of explanations. Given a traffic sample $x$ and its prediction label $y_{pred}$ generated by DL-NIDS $f(\cdot)$, we first remove the $top-k$ explanatory features pinpointed by the explanation methods. We then compute the score of prediction $y_{pred}$ without the $top-k$ features:

$$\mathrm{DA}_k(x, f) = f\left(x \mid remove(x_1, \ldots, x_k)\right). \tag{10}$$

If the selected features are actually relevant to the prediction $y_{pred}$, the accuracy should decrease drastically after performing $remove(x_1, \ldots, x_k)$, since the model loses critical information for making a correct prediction. We set the value of modified features to zero and calculate the Average Descriptive Accuracy (ADA) [8] for all samples in the dataset.

As illustrated in Fig. 5, our CANDICE-explainer has the steepest drop in ADA among all security systems compared to other explanation methods. Specifically, the ADA of CANDICE-MLP decreases to 56% with only two features modified. The results demonstrate that CANDICE-explainer can effectively identifies features that are highly relevant to the predictions of DL-NIDS. We attribute the high-fidelity of CANDICE-explainer to two reasons. First, the validity constraint applied to Eq. (7) requires that the generated counterfactual traffic $x_{cf}$ explicitly has a different prediction label to $x$, i.e., $y_{cf} \neq y_{pred}$. This forces CANDICE-explainer to find the features that truly affect the decision boundary to change the prediction. Second, the sparsity constraint requires CANDICE-explainer modifies as few features as possible, making the selected features both important and concise to satisfy the constraints. As a result, CANDICE-explainer finds the smallest subset of features that are critical to the prediction, thereby ensuring the fidelity of explanation.

#### 5.4.2. Explanation sparsity

This criterion evaluates whether the explainer generates sparse explanations with a limited number of features being changed, in order to reduce the workload of manual inspection and to provide concise explanations. We use the Mass Around Zero (MAZ) [8] to evaluate the sparsity of explanation. Specifically, we first scale the importance

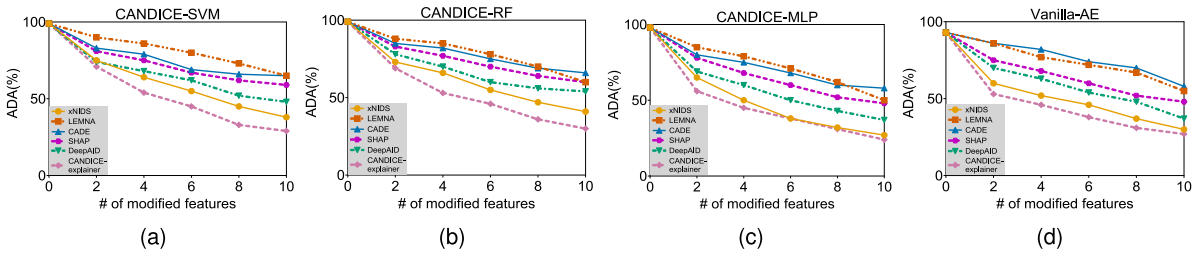



**Fig. 5.** Fidelity evaluation of explanation methods. (a) CANDICE-SVM. (b) CANDICE-RF. (c) CANDICE-MLP. (d) Vanilla-AE.

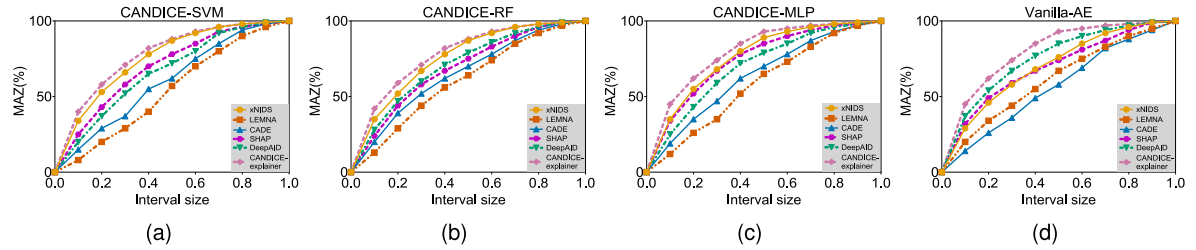



**Fig. 6.** Sparsity evaluation of explanation methods. (a) CANDICE-SVM. (b) CANDICE-RF. (c) CANDICE-MLP. (d) Vanilla-AE.

scores of counterfactual features to the range $[0, 1]$ and compute their normalized histogram $h$. The MAZ is then calculated as follows:

$$\mathrm{MAZ}(r) = \int_0^1 h(x)\mathrm{d}x \text{ for } r \in [0, 1]. \tag{11}$$

The sparse explanation is supposed to have a steep slope near zero in MAZ, as most of the features are not marked as relevant. The comparison results in Fig. 6 show that CANDICE-explainer outperforms baseline methods with the steepest slopes across all settings, demonstrating its superior performance in generating sparse explanations. Among all the explainers, LEMNA [12] shows the poorest ability in providing sparse explanation. This limitation stems from its underlying assumption that all features contribute similarly to the decision, leading to most of them being marked as relevant. In contrast, CANDICE-explainer prevents dense explanation by adding sparsity constraint to the counterfactual traffic generation process. In addition, the constraint of only modifying mutable features to generate feasible counterfactual traffic also promotes the sparsity.

### 5.4.3. Explanation stability

This criterion evaluates whether the explainer provides consistent explanations for the same detection result over multiple runs. We assess the stability of explanation by calculating the average Intersection Size (IS) [8] of explanatory features:

$$\mathrm{IS}(i, j) = \frac{|T_i \cap T_j|}{k}, \tag{12}$$

where $T_i$ and $T_j$ are the $top-k$ explanatory feature sets for the $i$th and $j$th run. For a stable explanation method, the intersection size is supposed to be close to 1, which indicates that the $top-k$ features are identical over multiple runs. In the experiment, we set $k = 10$ and run the explanation methods for five times.

As shown in Table 5, DeepAID [15] achieves an intersection size of 1.0 in all settings, indicating perfect stability of explanation. DeepAID benefits from its white-box property, i.e., using backpropagation to explain the model's decision, which ensures the consistency over different runs. On the other hand, the approximation-based methods (LEMNA [12] and xNIDS [13]), perturbation-based methods (CADE [14] and CANDICE-explainer), and SHAP [27] show sub-optimal performance due to the randomness of their sampling/perturbation process. However, our CANDICE-explainer still outperforms all other explainers, while overcoming the limitation of DeepAID which requires full access to model details. It provides stable explanations on all DL-NIDS systems, especially on CANDICE-MLP with a stability score of 0.729. We attribute the superior stability to the fact that, when generating counterfactual traffic under the three constraints, CANDICE-explainer is forced to constantly seek the most critical changes that lead to the reverse prediction. This objective-guided selection of explanatory features reduces the variation introduced by the randomness.

**Table 5**
Stability evaluation of explanation methods.

| Explanation methods | CANDICE-SVM | CANDICE-RF | CANDICE-MLP | Vanilla-AE |
|---|---|---|---|---|
| LEMNA [12] | 0.509 | 0.481 | 0.437 | 0.488 |
| CADE [14] | 0.537 | 0.540 | 0.547 | 0.393 |
| DeepAID [15] | 1 | 1 | 1 | 1 |
| xNIDS [13] | 0.680 | 0.677 | 0.685 | 0.630 |
| SHAP [27] | 0.625 | 0.597 | 0.572 | 0.603 |
| **CANDICE-explainer** | 0.701 | 0.683 | 0.729 | 0.635 |

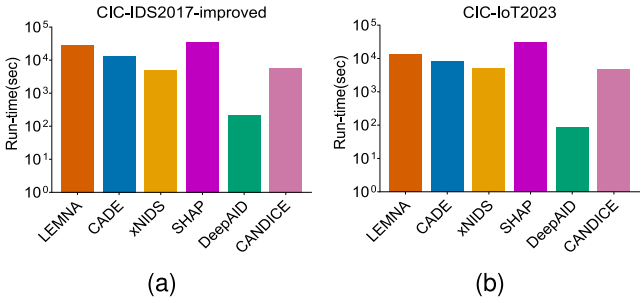



**Fig. 7.** Efficiency evaluation of explanation methods on two datasets. (a) CIC-IDS2017-improved dataset. (b) CIC-IoT2023 dataset.

#### 5.4.4. Explanation efficiency

This criterion evaluates whether the explainer is efficient in generating explanations with limited run-time overhead. When intrusion alerts arise, the security experts need to quickly understand the alerts to take immediate countermeasures to respond. Hence, the explanations should be available within a reasonable time, without obstructing the workflow of inspection. We assess the explanation efficiency by measuring the run-time required for each explainer to explain 2500 traffic samples. For LEMNA [12], xNIDS [13], and CADE [14], the time required for necessary procedures to generate explanations (e.g., training the surrogate model) is included. We use CANDICE-MLP as the underlying detector, repeating the experiment for five times with random sample selection and recording the average run-time overhead. The results on CIC-IDS2017-improved and CIC-IoT2023 datasets are shown in Fig. 7. LEMNA and CADE show sub-optimal efficiency due to the additional time overhead introduced by building additional learning models. The white-box method DeepAID [15] shows the best efficiency, as it utilizes gradient information and back-propagation to derive explanations, which is not easily available in practice. Regarding the black-box methods, CANDICE-explainer is 10x faster than SHAP [27] and LEMNA [12], two of the most popular explanation methods in the security domain. Of all the baselines, SHAP [27] shows the worst performance in efficiency because it requires extensive sampling of feature coalitions to approximate the Shapley values, leading to high computational costs. In contrast, CANDICE benefits from its goal-oriented optimization to guide the generation of explanations, combined with an early-stopping mechanism and limited search space to further speed up the process.

## 6. Discussion

The core objective of this work is to enhance the interpretability of AI-based NIDS. The experimental results in Section 5 demonstrate the effectiveness of the proposed CANDICE framework, in terms of explanation fidelity, sparsity, stability, and efficiency. In this section, we discuss the trade-offs between different explanation types (feature attribution vs. counterfactuals) and the reasons behind our choice of counterfactual explanations to explain NIDS, as well as the limitations and future work.

*Why counterfactual explanations for security applications?* As outlined in Table 1, local post-hoc explanation methods can be broadly categorized into Feature Attribution-based and example-based approaches. FA-based methods (LEMNA, xNIDS and SHAP) focus on identifying the contribution of individual input feature to model prediction, while example-based methods (DeepAID, CADE and CANDICE) explain model decisions by finding reference examples and comparing the differences between the references and the query instance. Counterfactual explanation, as we explored in this work, are the most typical example-based explanations in the field of Explainable AI. We discuss the trade-offs between these two paradigms in terms of fidelity, computational overhead, and human understanding. (1) *fidelity*: As the most important criterion for evaluating the explanation quality, we show in Section 5.4.1 that CANDICE outperform FA-based methods (LEMNA, xNIDS and SHAP) regarding fidelity. CANDICE provides high-fidelity explanations via two mechanisms in its design. First, it mitigates spurious explanation by learning disentangled traffic patterns that are more interpretable. Second, it generates counterfactuals that explicitly cross the model's decision boundary to ensure the explanation is necessary [58], mitigating the risk of unfaithfulness introduced by approximation-based FA methods such as LEMNA. (2) *computational overhead*: While CANDICE requires to solve a optimization problem to generate counterfactuals, as shown in Section 5.4.4, it still outperforms FA-based methods like SHAP with an reasonable run-time. In addition, CANDICE can further improve efficiency through amortized generation of counterfactuals [67]. (3) *human understanding*: FA-based methods provide explanations by outputting a list of importance scores of features. However, regarding security domain, the analysts are more concerned with the degree of deviation in the features that trigger an anomaly. Counterfactuals naturally provide such an intuitive understanding by demonstrating the minimal feature changes that reverse the model decision. Furthermore, we provide an intrusion profile along with the explanation in CANDICE, which helps the end-users to thoroughly understanding the attack.

*Limitations and future work.* Our work demonstrates the potential of counterfactual explanations in improving the interpretability of AI-based NIDS, yet several limitations remain and warrant future investigation. First, CANDICE currently focuses on generating tailored counterfactuals for individual detection results and instances requiring explanation. Although providing high-quality explanations, the per-instance optimization may hinder its scalability in high-throughput NIDS deployments. For future work, we plan to explore amortized generation of counterfactual explanations [67], e.g., training a generative model to produce counterfactuals for multiple input datapoints at once [68]. Second, while CANDICE's explanations offer intuitive guidance for designing defense strategies (e.g., by suggesting which features to modify and how), the development of low-cost, practical countermeasures remains an area for further exploration. One possible direction is to incorporate causal intervention [69] to identify the true cause-and-effect relationships in network traffic, which helps derive more practical defense rules. Third, CANDICE is now designed to detect and explain anomalies in network traffic. In the future work, we plan to extend our explainable intrusion detection framework to broader cybersecurity domains, such as binary code analysis [34] and API-level anomaly detection [70].

## 7. Conclusion

In this paper, we propose CANDICE, an explainable and intelligent framework for detecting and explaining intrusions in network traffic. CANDICE provides high-quality explanations for the detection results of NIDS by disentangling the traffic representations and generating counterfactual explanations. The evaluation results show that CANDICE outperforms existing explanation methods in terms of fidelity, sparsity, stability, and efficiency, while achieving high accuracy of above 96.10% in detecting intrusions. In addition, we showcase how CANDICE helps end-users better understand the detected attacks through a human-readable intrusion profile, which can further be utilized for developing practical and actionable defense mechanisms.

## CRediT authorship contribution statement

**Shuhua Li:** Writing – original draft, Validation, Methodology, Investigation, Conceptualization. **Ruiying Du:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Jing Chen:** Writing – review & editing, Project administration, Funding acquisition. **Kun He:** Writing – review & editing, Investigation, Funding acquisition. **Cong Wu:** Writing – review & editing, Validation. **Yebo Feng:** Writing – review & editing, Validation.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

The authors thank the editor and anonymous reviewers for their valuable comments. This research was supported in part by the National Key R&D Program of China under grant No. 2022YFB3103300.

## Data availability

Data will be made available on request.

## References


[1] Z. Ahmad, A.S. Khan, C.W. Shiang, J. Abdullah, F. Ahmad, Network intrusion detection system: A systematic study of machine learning and deep learning approaches, Trans. Emerg. Telecommun. Technol. 32 (1) (2021).

[2] G. Pang, C. Shen, L. Cao, A. van den Hengel, Deep learning for anomaly detection: A review, ACM Comput. Surv. 54 (2) (2022) 38:1–38:38.

[3] I.H. Sarker, A.S.M. Kayes, S. Badsha, H. AlQahtani, P.A. Watters, A. Ng, Cybersecurity data science: an overview from machine learning perspective, J. Big Data 7 (1) (2020) 41.

[4] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[5] C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks, IEEE Access 5 (2017) 21954–21961.

[6] N. Shone, N.N. Tran, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE Trans. Emerg. Top. Comput. Intell. 2 (1) (2018) 41–50.

[7] Y. Mirsky, T. Doitshman, Y. Elovici, A. Shabtai, Kitsune: An ensemble of autoencoders for online network intrusion detection, in: Proc. Netw. Distrib. Syst. Secur. Symp., NDSS, 2018.

[8] A. Warnecke, D. Arp, C. Wressnegger, K. Rieck, Evaluating explanation methods for deep learning in security, in: Proc. IEEE Eur. Symp. Secur. Privacy, EuroS&P, 2020, pp. 158–174.

[9] N. Capuano, G. Fenza, V. Loia, C. Stanzione, Explainable artificial intelligence in CyberSecurity: A survey, IEEE Access 10 (2022) 93575–93600.

[10] B.S. Rawal, G. Manogaran, A. Peter, Cybersecurity and Identity Access Management, Springer, 2023.

[11] S. Axelsson, The base-rate fallacy and the difficulty of intrusion detection, ACM Trans. Inf. Syst. Secur. 3 (3) (2000) 186–205.

[12] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, X. Xing, LEMNA: Explaining deep learning based security applications, in: Proc. ACM SIGSAC Conf. Comput. Commun. Secur., CCS, 2018, pp. 364–379.

[13] F. Wei, H. Li, Z. Zhao, H. Hu, XNIDS: Explaining deep learning-based network intrusion detection systems for active intrusion responses, in: Proc. USENIX Secur. Symp., USENIX, 2023, pp. 4337–4354.

[14] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, G. Wang, CADE: Detecting and explaining concept drift samples for security applications, in: Proc. USENIX Secur. Symp., USENIX, 2021, pp. 2327–2344.

[15] D. Han, Z. Wang, W. Chen, Y. Zhong, S. Wang, H. Zhang, J. Yang, X. Shi, X. Yin, DeepAID: Interpreting and improving deep learning-based anomaly detection in security applications, in: Proc. ACM SIGSAC Conf. Comput. Commun. Secur., CCS, 2021, pp. 3197–3217.

[16] G. Srivastava, R.H. Jhaveri, S. Bhattacharya, S. Pandya, Rajeswari, P.K.R. Maddikunta, G. Yenduri, J.G. Hall, M. Alazab, T.R. Gadekallu, XAI for cybersecurity: State of the art, challenges, open issues and future directions, 2022, arXiv preprint arXiv:2206.03585.

[17] X. Wang, H. Chen, Z. Wu, W. Zhu, et al., Disentangled representation learning, IEEE Trans. Pattern Anal. Mach. Intell. (2024).

[18] I.F. Kilincer, F. Ertam, A. Sengür, Machine learning methods for cyber security intrusion detection: Datasets and comparative study, Comput. Netw. 188 (2021) 107840.

[19] M. Fatima, O. Rehman, S. Ali, M.F. Niazi, ELIDS: Ensemble feature selection for lightweight IDS against DDoS attacks in resource-constrained IoT environment, Future Gener. Comput. Syst. 159 (2024) 172–187.

[20] D.D. Bikila, J. Čapek, Machine learning-based attack detection for the internet of things, Future Gener. Comput. Syst. 166 (2025) 107630.

[21] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, Malware traffic classification using convolutional neural network for representation learning, in: Proc. Int. Conf. Inf. Netw., ICOIN, 2017, pp. 712–717.

[22] X. Li, W. Chen, Q. Zhang, L. Wu, Building auto-encoder intrusion detection system based on random forest feature selection, Comput. Secur. 95 (2020) 101851.

[23] R.D. Corin, S. Millar, S. Scott-Hayward, J.M. del Rincón, D. Siracusa, Lucid: A practical, lightweight deep learning solution for DDoS attack detection, IEEE Trans. Netw. Serv. Manag. 17 (2) (2020) 876–889.

[24] Z. Jin, J. Zhou, B. Li, X. Wu, C. Duan, FL-IIDS: A novel federated learning-based incremental intrusion detection system, Future Gener. Comput. Syst. 151 (2024) 57–70.

[25] D.L. Aguilar, M.A. Medina-Pérez, O. Loyola-González, K.R. Choo, E. Bucheli-Susarrey, Towards an interpretable autoencoder: A decision-tree-based autoencoder and its application in anomaly detection, IEEE Trans. Dependable Secur. Comput. 20 (2) (2023) 1048–1059.

[26] M.T. Ribeiro, S. Singh, C. Guestrin, Why should I trust you?: Explaining the predictions of any classifier, in: Proc. ACM Int. Conf. Knowl. Discovery Data Mining, SIGKDD, 2016, pp. 1135–1144.

[27] S.M. Lundberg, S. Lee, A unified approach to interpreting model predictions, in: Proc. Adv. Neural Inf. Process. Syst., NeurIPS, 2017, pp. 4765–4774.

[28] G. Plumb, D. Molitor, A. Talwalkar, Model agnostic supervised local explanations, in: Proc. Adv. Neural Inf. Process. Syst., NeurIPS, 2018, pp. 2520–2529.

[29] R.C. Fong, A. Vedaldi, Interpretable explanations of black boxes by meaningful perturbation, in: Proc. IEEE Int. Conf. Comput. Vis., ICCV, 2017, pp. 3449–3457.

[30] M. Wu, M.C. Hughes, S. Parbhoo, M. Zazzi, V. Roth, F. Doshi-Velez, Beyond sparsity: Tree regularization of deep models for interpretability, in: Proc. AAAI Conf. Artif. Intell., 2018, pp. 1670–1678.

[31] M. Ibrahim, M. Louie, C. Modarres, J.W. Paisley, Global explanations of neural networks: Mapping the landscape of predictions, in: Proc. AAAI/ACM Conf. AI, Ethics, and Soc., AIES, 2019, pp. 279–287.

[32] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, A. Pescapè, XAI meets mobile traffic classification: Understanding and improving multimodal deep learning architectures, IEEE Trans. Netw. Serv. Manag. 18 (4) (2021) 4225–4246.

[33] C. Smutz, A. Stavrou, Malicious PDF detection using metadata and structural features, in: Proc. Annu. Comput. Secur. Appl. Conf., ACSAC, 2012, pp. 239–248.

[34] E.C.R. Shin, D. Song, R. Moazzezi, Recognizing functions in binaries with neural networks, in: Proc. USENIX Secur. Symp., USENIX, 2015, pp. 611–626.

[35] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, K. Knight, Sparsity and smoothness via the fused lasso, J. R. Stat. Soc. B: Stat. Methodol. 67 (1) (2005) 91–108.

[36] R. Fong, M. Patrick, A. Vedaldi, Understanding deep networks via extremal perturbations and smooth masks, in: Proc. IEEE Int. Conf. Comput. Vis., ICCV, 2019, pp. 2950–2958.

[37] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, in: Proc. IEEE Int. Conf. Comput. Vis., ICCV, 2017, pp. 618–626.

[38] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, in: Proc. Int. Conf. Mach. Learn., ICML, 2017, pp. 3145–3153.

[39] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, in: Proc. Int. Conf. Learn. Represent., ICLR, 2014.

[40] L. Shapley, A value for N-person games, 1953.

[41] A. Oseni, N. Moustafa, G. Creech, N. Sohrabi, A. Strelzoff, Z. Tari, I. Linkov, An explainable deep learning framework for resilient intrusion detection in IoT-enabled transportation networks, IEEE Trans. Intell. Transp. Syst. 24 (1) (2023) 1000–1014.

[42] R. Kalakoti, H. Bahsi, S. Nõmm, Improving IoT security with explainable AI: quantitative evaluation of explainability for IoT botnet detection, IEEE Internet Things J. 11 (10) (2024) 18237–18254.

[43] T. Zebin, S. Rezvy, Y. Luo, An explainable AI-based intrusion detection system for DNS over HTTPS (DoH) attacks, IEEE Trans. Inf. Forensics Secur. 17 (2022) 2339–2349.

[44] M. Wang, K. Zheng, Y. Yang, X. Wang, An explainable machine learning framework for intrusion detection systems, IEEE Access 8 (2020) 73127–73141.

[45] M. Keshk, N. Koroniotis, N. Pham, N. Moustafa, B.P. Turnbull, A.Y. Zomaya, An explainable deep learning-enabled intrusion detection framework in IoT networks, Inf. Sci. 639 (2023) 119000.

[46] L. Antwarg, R.M. Miller, B. Shapira, L. Rokach, Explaining anomalies detected by autoencoders using Shapley additive explanations, Expert Syst. Appl. 186 (2021) 115736.

[47] R. Kumar, A. Aljuhani, D. Javeed, P. Kumar, S. Islam, A.K.M.N. Islam, Digital twins-enabled zero touch network: A smart contract and explainable AI integrated cybersecurity framework, Future Gener. Comput. Syst. 156 (2024) 191–205.

[48] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: Proc. Int. Conf. Learn. Represent., ICLR, 2014.

[49] T.Q. Chen, X. Li, R.B. Grosse, D. Duvenaud, Isolating sources of disentanglement in variational autoencoders, in: Proc. Int. Conf. Learn. Represent., ICLR, 2018.

[50] I. Higgins, L. Matthey, A. Pal, C.P. Burgess, X. Glorot, M.M. Botvinick, S. Mohamed, A. Lerchner, Beta-VAE: Learning basic visual concepts with a constrained variational framework, in: Proc. Int. Conf. Learn. Represent., ICLR, 2017.

[51] A. Kumar, P. Sattigeri, A. Balakrishnan, Variational inference of disentangled latent concepts from unlabeled observations, in: Proc. Int. Conf. Learn. Represent., ICLR, 2018.

[52] H. Kim, A. Mnih, Disentangling by factorising, in: Proc. Int. Conf. Mach. Learn., ICML, 2018, pp. 2654–2663.

[53] E. Dupont, Learning disentangled joint continuous and discrete representations, in: Proc. Adv. Neural Inf. Process. Syst., NeurIPS, 2018, pp. 708–718.

[54] S. Wachter, B.D. Mittelstadt, C. Russell, Counterfactual explanations without opening the black box: Automated decisions and the GDPR, Harv. JL Tech. 31 (2017) 841.
[55] A. Dhurandhar, P. Chen, R. Luss, C. Tu, P. Ting, K. Shanmugam, P. Das, Explanations based on the missing: Towards contrastive explanations with pertinent negatives, in: Proc. Adv. Neural Inf. Process. Syst., NeurIPS, 2018, pp. 590–601.
[56] R.K. Mothilal, A. Sharma, C. Tan, Explaining machine learning classifiers through diverse counterfactual explanations, in: Proc. Conf. Fairness, Accountability, and Transparency, FAT, 2020, pp. 607–617.
[57] R. Guidotti, Counterfactual explanations and how to find them: literature review and benchmarking, Data Min. Knowl. Discov. 38 (5) (2024) 2770–2824.
[58] R.K. Mothilal, D. Mahajan, C. Tan, A. Sharma, Towards unifying feature attribution and counterfactual explanations: Different means to the same end, in: Proc. AAAI/ACM Conf. AI, Ethics, and Soc., AIES, 2021, pp. 652–663.
[59] E.C.P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, A.A. Ghorbani, CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment, Sensors 23 (13) (2023) 5941.
[60] G. Engelen, V. Rimmer, W. Joosen, Troubleshooting an intrusion detection dataset: the CICIDS2017 case study, in: Proc. IEEE Secur. Privacy Workshops, SPW, 2021, pp. 7–12.
[61] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, in: Proc. Int. Conf. Inf. Syst. Secur. Privacy, ICISSP, 2018, pp. 108–116.
[62] M. MontazeriShatoori, L. Davidson, G. Kaur, A.H. Lashkari, Detection of DoH tunnels using time-series classification of encrypted traffic, in: Proc. IEEE Int. Conf. Dependable, Autonomic Secure Comput., Int. Conf. Pervasive Intell. Comput., Int. Conf. Cloud Big Data Comput., Int. Conf. Cyber Sci. Technol. Congr., DASC/PiCom/CBDCom/CyberSciTech, 2020, pp. 63–70.
[63] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl., CISDA, 2009, pp. 1–6.
[64] J. Snoek, H. Larochelle, R.P. Adams, Practical Bayesian optimization of machine learning algorithms, in: Proc. Adv. Neural Inf. Process. Syst., NeurIPS, 2012, pp. 2960–2968.
[65] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[66] Student, The probable error of a mean, Biometrika (1908) 1–25.
[67] S. Verma, V. Boonsanong, M. Hoang, K. Hines, J. Dickerson, C. Shah, Counterfactual explanations and algorithmic recourses for machine learning: A review, ACM Comput. Surv. 56 (12) (2024) 312:1–312:42.
[68] F. Yang, S.S. Alva, J. Chen, X. Hu, Model-based counterfactual synthesizer for interpretation, in: Proc. ACM Int. Conf. Knowl. Discovery Data Mining, SIGKDD, 2021, pp. 1964–1974.
[69] A. Karimi, B. Schölkopf, I. Valera, Algorithmic recourse: from counterfactual explanations to interventions, in: Proc. ACM Fairness, Accountability, and Transparency, FAccT, 2021, pp. 353–362.
[70] R. Guntur, API security: Access behavior anomaly dataset, 2022, https://www.kaggle.com/datasets/tangodelta/api-access-behaviour-anomaly-dataset.

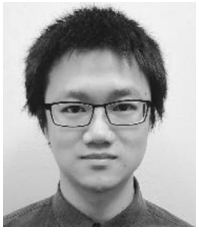
**Shuhua Li** received the M.S. degree in cyberspace security from Wuhan University, Wuhan, China, in 2019. She is currently working toward the Ph.D. degree with the School of Cyber Science and Engineering, Wuhan University, China. Her research interests include network security and traffic analysis.

**Ruiying Du** received the B.S., M.S., PH. D. degrees in computer science in 1987, 1994 and 2008, from Wuhan University, Wuhan, China. She is a professor at School of Cyber Science and Engineering, Wuhan University. Her research interests include network security, wireless network, cloud computing and mobile computing. She has published more than 80 research papers in many international journals and conferences, such as TPDS, USENIX Security, CCS, INFOCOM, SECON, TrustCom, et al.

**Jing Chen** received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan. He is a professor at School of Cyber Science and Engineering, Wuhan University. His research interests in computer science are in the areas of network security, cloud security. He has published more than 100 research papers in many international journals and conferences, such as TDSC, TIFS, USENIX Security, CCS, INFOCOM, TC, TPDS, et al. He acts as a reviewer for many journals and conferences, such as TIFS, TC, TON.

**Kun He** received a Ph.D. in computer science from the computer school, Wuhan University. He is an associate professor at School of Cyber Science and Engineering, Wuhan University. His research interests include cryptography, network security, mobile computing, and cloud computing. He has published more than 20 research papers in many international journals and conferences, such as TC, TIFS, TDSC, TMC, USENIX Security, CCS, INFOCOM, et al.

**Cong Wu** received the Ph.D. degree from the School of Cyber Science and Engineering, Wuhan University, in 2022. He is currently a research fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include AI system security and Web3 security. His research outcomes have appeared in USENIX Security, CCS, TDSC, TIFS, et al.



**Yebo Feng** is a research fellow in the School of Computer Science and Engineering (SCSE) at Nanyang Technological University (NTU). He received his Ph.D. degree in Computer Science from the University of Oregon (UO) in 2023. His research interests include network security, blockchain security, and anomaly detection. He is the recipient of the Best Paper Award of 2019 IEEE CNS, Gurdeep Pall Graduate Student Fellowship of UO, and Ripple Research Fellowship. He has served as the reviewer of IEEE TDSC, IEEE TIFS, ACM TKDD, IEEE JSAC, IEEE COMST, etc. Furthermore, he has been a member of the program committees for international conferences including SDM, CIKM, and CYBER, and has also served on the Artifact Evaluation (AE) committees for USENIX OSDI and USENIX ATC.