TRUST XAI: A Novel Model for Explainable AI with An Example Using IIoT Security

Maede Zolanvari*, Zebo Yang*, Khaled Khan†, Raj Jain*, and Nader Meskin†
*Department of Computer Science and Engineering, Washington University, St. Louis, MO USA
†Department of Computer Science and Engineering, Qatar University, Doha, Qatar

***Abstract*—Despite AI's significant growth, its "black box" nature creates challenges in generating adequate trust. Thus, it is seldom utilized as a standalone unit in high-risk applications. Explainable AI (XAI) has emerged to help with this problem. Designing effectively fast and accurate XAI is still challenging, especially in numerical applications. We propose a novel XAI model named Transparency Relying Upon Statistical Theory (TRUST) for XAI. TRUST XAI models the statistical behavior of the underlying AI's outputs. Factor analysis is used to transform the input features into a new set of latent variables. We use mutual information to rank these parameters and pick only the most influential ones on the AI's outputs and call them "representatives" of the classes. Then we use multi-model Gaussian distributions to determine the likelihood of any new sample belonging to each class. The proposed technique is a surrogate model that is not dependent on the type of the underlying AI. TRUST is suitable for any numerical application. Here, we use cybersecurity of the industrial internet of things (IIoT) as an example application. We analyze the performance of the model using three different cybersecurity datasets, including "WUSTL-IIoT", "NSL-KDD", and "UNSW". We also show how TRUST is explained to the user. The TRUST XAI provides explanations for new random samples with an average success rate of 98%. Also, the advantages of our model over another popular XAI model, LIME, including performance, speed, and the method of explainability are evaluated.**

***Index Terms*—Explainable AI (XAI), Artificial Intelligence, Transparency, Statistical Modeling, Gaussian Distributions**

I. Introduction

THE impact of artificial intelligence (AI) on today's technological advancements is undeniable. Despite the popularity of AI, it is limited by its current inability to build trust. Researchers and industrial leaders have a hard time explaining the decisions that sophisticated AI algorithms come up with because they cannot fully understand why and how these "black boxes" make their decisions.

Trusting AI blindly impacts its applicability and legitimacy. Based on Gartner's predictions, 85% of the AI projects until 2020 will produce inaccurate outcomes resulting from the organization's limited knowledge of the deployed AI and its behavior [1]. Also, the increasing growth in implementing AI has raised concerns of 91% of cybersecurity professionals about cyberattacks using AI [2]. In the medical field, dedicated AI algorithms are able to precisely (sometimes even more accurately than a medical doctor) detect rare diseases. However, due to the lack of AI's decision transparency and explainability, the results are not considered trustworthy enough to be utilized in practice [3]. A 2018 global survey shows that more than 67% of the business leaders do not trust AI, and they believe AI's ambiguity will have negative impacts on their business in the next five years [4].

Designing self-explanatory models has gained much attention recently. Explainable AI (XAI) deals with psychology and cognitive science to provide transparent reasonings for users [5]. Based on how the underlying AI is designed and what the input types (image, text, voice, numerical data) are, the methods of explainability differ.

XAI can be applied either by using interpretable models as the underlying AI model or by developing a separate explainer working in parallel with the underlying model to explain its behaviors. However, learning performance and explainability of the model usually have an inverse relationship. Better learning performance comes with more sophistication in finding high-level relationships in the data features, as such it becomes more challenging for the individuals to grasp its rationality. Therefore, choosing interpretable models as the primary AI yields a significantly impeded learning performance. Hence, surrogate explainers are better suited to successful XAI implementation. They are independent and do not affect the performance of the underlying AI model.

It is important to note that, in the XAI domain, it is not expected that the explanation would be in layman's terms. Most of the available research work uses another learning model (that is more easily understood than the underlying AI) as the surrogate explainer. Therefore, they have assumed that the user is an expert in the AI model's internal working and knows the mathematics behind it. Here, we argue that statistics are much more easily verifiable and explainable than AI. Therefore, we propose using statistical models to simplify the explaining process.

In this paper, we propose an XAI model that provides transparency, relying upon statistical theory (TRUST). The TRUST XAI is a surrogate explainer that provides interpretability without sacrificing the performance of the underlying model or putting any restrictions on it. The explainer does not depend on the type of underlying AI in any way; thus, it is completely model-agnostic[1]. Our proposed explainer, alongside the underlying AI, will be able to contribute in future deployments of AI as a standalone unit that not only demonstrates high-level performance, but is also transparent, independent, and reliable.

[1]This simply means the XAI does not care about the underlying AI algorithm. All it requires is being able to probe the model with any arbitrary input and get the output from the "black box".

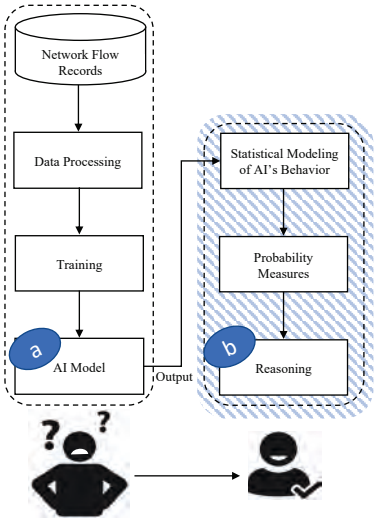


Fig. 1: Integrated TRUST explainer in an AI-based system. While (a) can be very complex and non-interpretable, (b) should be as easily interpretable as possible for humans.

Most of the research so far in the XAI domain has been done for image-based applications. However, there are critical applications such as IoT and network security that primarily deal with numerical data. One of the distinctions of the TRUST explainer is that it applies to numerical data.

The TRUST technique works solely with the feature sets and the outputs from the underlying AI. If the underlying AI is modeled as shown in Figure 1 on the left side, our proposed novel explaining technique is represented on the right using a background with diagonal stripes. The first step builds the core of the explainer, and it has to be calculated just once, and only the last two steps (marked with a dashed line square) are required for explaining new data instances.

To provide a practical instance, we use our dataset, "WUSTL-IIoT," generated from our industrial IoT (IIoT) testbed. We have also tested the performance of the model on two other datasets,"NSL-KDD" and "UNSW". We empirically prove that our proposed XAI model is successful at explaining the labels predicted by the underlying AI in the matter of few milliseconds. Moreover, for the sake of comparison, we have implemented and applied local interpretable model-agnostic explanations (LIME) on the three mentioned datasets. Our results show LIME is much slower than our model and is not practical for real-time applications or dealing with a large number of samples.

The contributions of this work are as follows:

- We propose a novel, highly accurate technique that requires no compromise in the selection or performance of the underlying AI, while it provides transparency in the model's results.
- Our TRUST technique is very fast at reasoning the AI model's behavior, and therefore, it can be counted as one of the first XAI models suitable for real-time numerical applications.
- We plan to release our IIoT security dataset for use by other researchers in the emerging field of XAI. We have confirmed the integration of the underlying AI with our TRUST XAI on the dataset and will release both the codes and the dataset to support the research community.

II. Related Work

Feature importance is the most commonly utilized method of explainability [6]. It is one of the basic techniques to find the key features impacting the AI's outcome. Feature selection is useful for data preprocessing [7]. However, when used as an XAI technique, there is no way of evaluating the performance of the explainer.

In [8], the SHAP (SHapley Additive exPlanations) values, originally developed by [9], are used to explain the predictions made by the underlying model in network security. The results of the XAI model are presented as which features are more indicative of the class of data. In [10], a feature importance technique, layer-wise relevance propagation (LRP), is used for explainability. The method applies only to deep neural network (DNN) models, hence, it is not model agnostic. In [11], a feature recalibration technique, AdaCare, is developed. As a part of AdaCare, important features are utilized to include interpretability for decompensation and mortality predictions.

In [12], a survey on available approached utilized for XAI is provided. Related challenges and future directions are also discussed throughly. One of the main focuses of this paper is dedicated to XAI in the medical domain and how the black-box nature of learning models impacts the legitimacy of their outputs. It shows the urgency of having clarity on how the utilized AI makes it decisions to prevent exploitions with malicious intent.

In [13], an adversarial technique is utilized as a means of interpretability. The method calculates the minimal adjustments to the important features' values of a misclassified sample until the underlying AI changes its prediction. However, the proposed method is semi-agnostic, where it only works for the AI models that have a defined gradient cross-entropy loss function. Additionally, no performance assessment is provided.

Finally, LIME, which is used as a *benchmark XAI* in this paper, is introduced in [14]. It generates local surrogate models for each instance to explain its label. For each instance, a new dataset is synthesized which consists of samples with feature values drawn from a normal distribution with a mean and standard deviation corresponding to that instance. The underlying AI then labels the synthesized samples. Also, they are assigned weights based on their distance from the instance of interest. Next, a simple interpretable learning model (usually a linear regression model) is trained with the new synthetic dataset to explain the outcome of the underlying AI for that specific instance. Later in this paper, we compare the performance of LIME with our TRUST model and show how our model is superior in terms of both speed and performance.

TABLE I: Symbol Table

Symbol	Description
N	Number of observations in the training dataset
C	Total number of classes
c	Index indicating class number, $c \in \{1, ..., C\}$
N_c	Number of observations classified by the underlying AI as being in class c, $\sum_{c=1}^{C} N_c = N$
K	Total number of features used in the AI model
k	Number of representatives in the XAI model, $k \ll K$
i	Index indicating the feature number, $i \in \{1, ..., K\}$ for the AI model and the representative number $i \in \{1, ..., k\}$ for the XAI model
f_i	i^{th} feature
F_i^c	Random variable of i^{th} factor for class c, $F_i^c \in \mathbb{R}^{N_c \times 1}$
F_i	Vertical concatenation of $\{F_i^1, \cdots, F_i^C\}$
R_i^c	Random variable of i^{th} representative for class c, $R_i^c \in \mathbb{R}^{N_c \times 1}$
R_i	Vertical concatenation of $\{R_i^1, \cdots, R_i^C\}$
$r_{j,i}^c$	R_i^c value in the jth observation, $j \in \{1, ..., N_c\}$
$\boldsymbol{r_i^c}$	Vector of observations belonging to random variable R_i^c
$\boldsymbol{R^c}$	$N_c \times k$ observation matrix consisting of $r_{j,i}^c$ values arranged so that the column i consists of $\boldsymbol{r_i^c}$'s values
M_i^c	Number of modes of R_i^c in its MMG distribution
w_i	Importance coefficient of R_i
λ_i^c	Eigenvalue of F_i^c

III. Proposed TRUST Explainer

Our goal is to explain the rationale behind the data labeling of the main AI model. Statistical techniques are applied to the AI's outputs. Table I presents the symbols used in this paper.

A. Problem Formalization

Our XAI model produces a set of vectors called "representatives" from the training set. We make three assumptions about the dataset and the representatives as follows.

Assumption 1: The statistical inferences made by the XAI model must represent the data's characteristics. Therefore, the data used to train the underlying AI is also used to build the TRUST model. If the AI model has to be retrained due to a significant change in the data's characteristics, the core of the explainer must be rebuilt as well.

Assumption 2: The representatives must be mutually independent. This allows their joint probability function to be the multiplication of individual probability functions. This assumption simplifies our XAI technique and makes it easier for human users to understand.

Assumption 3: Suppose for the i^{th} representative R_i, we have a sample $\boldsymbol{r_i^c}$ with N_c observations belonging to the class c. The distribution of these observations can be approximated with good accuracy by a one-dimensional multi-modal Gaussian (MMG) distribution with M_i^c sub-populations, which are called *modes*, with $\{\mu_{i,1}^c, ..., \mu_{i,M_i^c}^c\}$ means and $\{\sigma_{i,1}^c, ..., \sigma_{i,M_i^c}^c\}$ standard deviations. In case of a unimodal distribution, M_i^c is equal to one.

B. Determining Representatives

We utilize factor analysis [15] to determine the representatives. Factor analysis satisfies the requirements mentioned in Assumption 2 and more. It provides a linear combination of the dataset's features. It also reduces the redundancy in the feature space, and hence, our model is less likely to overfit. Further, since these new independent latent variables are combinations of the features, we still preserve the most valuable parts of all the features. Factor analysis projects the feature values onto directions that maximize the percentage of explained variance in the data. Another advantage of this technique is that the resulting factors are orthogonal. Therefore, they are independent with zero covariations.

Suppose we have a feature set $\mathcal{F} = \{f_1, ..., f_K\}$, where K is the total number of the features. By running the factor analysis for observations belonging to each class separately, these factors are determined per class. For example, for class c, factors $\{F_1^c, ..., F_K^c\}$ are produced. The pseudo-code of the factor analysis function, $\mathcal{FA}$, is presented as Algorithm 1. In this algorithm, each data instance belongs to one of the C classes.

In Algorithm 1, ρ_{f_i,f_j} is the Pearson correlation coefficient, $\chi^2_{f_i,f_j}$ is the chi-square, and $\eta^2_{f_i,f_j}$ is one-way ANOVA (one-way analysis of variance) score between feature i and feature j from the standardized matrix of X^c [16] and [17]. Lastly, singular value decomposition (SVD) is applied to the relation matrix RM^c for factorization, removing the interdependencies among the features, and projecting them in orthogonal dimensions.

Algorithm 1 Factor Analysis

Function: $\mathcal{FA}$
Input: Training Set: $X \neq \emptyset \in \mathbb{R}^{N \times K}$;
AI Model: $\mathcal{AI}$; Features Set: $\mathcal{F}$
$y \leftarrow \mathcal{AI}(X)$
Set labels of each row in $X \leftarrow y$
Divide X into $X^c \in \mathbb{R}^{N_c \times K}$ sets per class
for each X^c **do**
  $X^{c\prime} \leftarrow$ standardized (X^c) ▷ So it has zero mean and unit variance
  Build the relationship matrix (RM^c) such that:
  for each pair (i, j) in $\mathcal{F}(X^c)$ **do**
    if f_i & f_j are both quantitative **then**
      $RM_{i,j}^c = \rho^2_{f_i,f_j}$
    else if f_i & f_j are both qualitative **then**
      $RM_{i,j}^c = \chi^2_{f_i,f_j}$
    else
      $RM_{i,j}^c = \eta^2_{f_i,f_j}$
    end if
  end for
  $\{F_1^{c\prime}, ..., F_K^{c\prime}\}$ & $\{\lambda_1^c, ..., \lambda_K^c\} \leftarrow$ SVD(RM^c)
  $\{F_1^c, ..., F_K^c\} \leftarrow$ unstandardized $(\{F_1^{c\prime}, ..., F_K^{c\prime}\})$
  return $\{F_1^c, ..., F_K^c\}$
end for

We use the factors resulting from Algorithm 1 to pick the "representatives" later in Algorithm 2. There is no need to represent each class by its all factors. Based on the applications and constraints, we can utilize only a few of them.

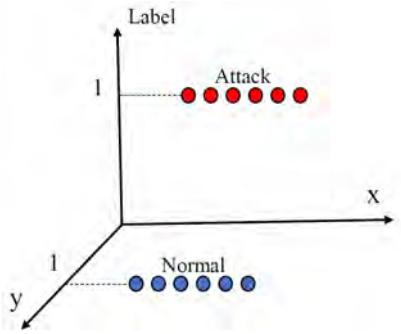


Fig. 2: Two-dimensional data with features highly correlated with the labels

The effectiveness of a factor in distinguishing the class label is not related to the percentage of explained variation of the data λ_i^c by that factor. Suppose we have two-dimensional data with their class labels on the third axis. An example is demonstrated in Fig. 2. The attack class is shown by the red points and class normal by the blue points. As seen in this figure, since the variation in the y-axis values per class is not large, the y-axis feature would not be considered important. Therefore, in case of a dimension reduction or factor analysis, separately for each class, the x-axis values would be the primary axis representing each class's samples. This is because x-axis explains almost all the variation in the data points for each class. Meanwhile, it is trivial to see that the class labels could easily be distinguished by their y-axis values. Hence, removing them or treating them as not important will cause a great loss in the accuracy of the proposed model.

Therefore, we devised a method using correlation between each factor and the class label to rank the factors and pick the ones with the highest correlation. We use mutual information (MI), which quantifies the amount of information one can get about one variable by knowing the other. MI between the factors and the class labels is calculated as

$$\text{MI}(y, F_i) = \text{H}(y) - \text{H}(y|F_i) \tag{1}$$

where, $F_i = [{F_i^1}^T \dots {F_i^C}^T]^T$ is the vertical concatenation of the ith factors of all the classes. The T superscript denotes a transpose of the vector/matrix. $\text{MI}(y, F_i)$ is the mutual information between the class label vectors y and F_i. H(.) is the entropy function, which is defined as follows.

$$\text{H}(y) = -\sum_c P(y=c)\log(P(y=c)) \tag{2}$$

For the sake of simplicity and not having to use differential entropy (since F_i is not discrete), we bin the values in F_i and ζ_i's are the possible outcomes of the binned random variable F_i. Suppose $P(\zeta_i)$ is the probability of $F_i = \zeta_i$. The conditional entropy, $\text{H}(y|F_i)$, is calculated as follows.

$$\begin{aligned}
&\text{H}(y|F_i) = \\
&\sum_{\zeta_i} P(F_i=\zeta_i)\text{H}(y|F_i=\zeta_i) = \\
&-\sum_{\zeta_i} P(F_i=\zeta_i)\sum_c P(y=c|F_i=\zeta_i)\log P(y=c|F_i=\zeta_i) \\
&= -\sum_{\zeta_i}\sum_c P(c,\zeta_i)\log\frac{P(c,\zeta_i)}{P(\zeta_i)}
\end{aligned} \tag{3}$$

We use the MI values as the importance coefficients for the factors. Therefore, the importance coefficient for F_i, w_i, is equal to $\text{MI}(y, F_i)$.

Then, we pick the first k factors with the highest w_i values as the"representatives". Thus, they represent each class's feature values and behavior. The choice of k provides a tradeoff between the speed and the performance of the explainer. Large k may result in a better performance and a lower speed and vice-versa. The algorithm to select the representatives is summarized in Algorithm 2.

Algorithm 2 Picking Representatives

Function: $\mathcal{PR}$
Input: Training Set: $X \neq \emptyset \in \mathbb{R}^{N\times K}$;
AI Model: $\mathcal{AI}$; Features Set: $\mathcal{F}$
$\{F_1^c, ..., F_K^c\} \leftarrow \mathcal{FA}(X, \mathcal{AI}, \mathcal{F})$
$y \leftarrow \mathcal{AI}(X)$
for each i **do**
  $F_i = [{F_i^1}^T, \cdots, {F_i^C}^T]^T$
  $MI_i \leftarrow \text{MI}(y, F_i)$
  $MI_i{}' \leftarrow$ sort MI_i in descending order
  $\{R_1^c, ..., R_k^c\} \leftarrow$ Pick top k factors with highest $MI_i{}'$
  return $\{R_1^c, ..., R_k^c\}$
end for

C. Density Estimation

To study the statistical attributes of the representatives' values, we approximate their density functions with one-dimensional MMG distributions.

Suppose for all the data instances belonging to the training set labeled as class c, X^c, by the black-box AI model, representative $R_i^c, i = 1, .., k$, has M_i^c modes. For now, assume we know the value of M_i^c; we will discuss how to calculate it later in the next section. Its one-dimensional multi-modal probability density functions (PDF) can be modeled as follows.

$$p_i(R_i^c) = \sum_{m=1}^{M_i^c} \gamma_{i,m}^c \mathcal{N}(R_i^c | \mu_{i,m}^c, \sigma_{i,m}^c) \tag{4}$$

$\gamma_{i,m}^c$ is the component weight of sub-population m for R_i^c observations in class c. The $\sum_m \gamma_{i,m}^c = 1, \forall c \vee \forall i$ should be satisfied, so the total area under the PDF normalizes to one for every representative per class. Here we use the expectation-maximization (EM) algorithm [18] to fit the representatives' values to a proper distribution in form of Eq. 4.

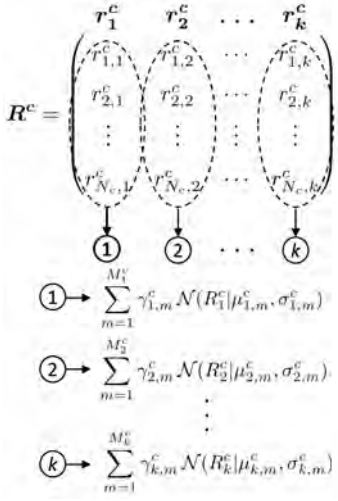


Fig. 3: Approximating representatives to unique MMG distributions for class c

Afterward, we end up with $k \times C$ distributions. These distributions build the backbone of the explainer. Figure 3 shows the approximation process for the values of class c representatives as an example. The $\boldsymbol{R^c} = [\boldsymbol{r_1^c}, \boldsymbol{r_2^c}, ..., \boldsymbol{r_k^c}] \in \mathbb{R}^{N_c \times k}$ equates to the matrix of observations of the representatives of the class c, where R_i^c is the random variable for the set of observations $\boldsymbol{r_i^c}$.

Now, it is time to show how our XAI technique explains the data instances labeled as a specific class by the AI model. First, the likelihood of the new instance's representatives belonging to each class's representatives is calculated. This likelihood can be estimated by the density function value. The total likelihood is the product of the likelihoods of all representatives. We apply a logarithm transformation to decrease the computational complexity and compute the log-likelihoods. By turning the multiplication to addition, we simplify the computation and avoid any underflow or overflow. Suppose the new instance is $\boldsymbol{z} = [z_1, ..., z_K] \in \mathbb{R}^{1 \times K}$. The inputs to the proposed XAI are only the representatives' values. Therefore, $\boldsymbol{z}$ is projected on the factors' space and cropped to its corresponding representatives' values $\boldsymbol{z'} = [z'_1, ..., z'_k] \in \mathbb{R}^{1 \times k}$. The conditional likelihood of z'_i, $i = 1, .., k$ belonging to class c can be written as:

$$p_i(z'_i|c) = \sum_{m=1}^{M_i^c} \exp\Big(\log(\gamma_{i,m}^c) + \log(\mathcal{N}(z'_i|\mu_{i,m}^c, \sigma_{i,m}^c))\Big) \tag{5}$$

After taking a logarithm from Eq. 5, the conditional log-likelihood of z'_i belonging to R_i^c is calculated by Eq. 6.

$$\begin{aligned} &\log(p(z'_i|c)) = \\ &\log\Big(\sum_{m=1}^{M_i^c} \exp\Big(\log(\gamma_{i,m}^c) + \log(\mathcal{N}(z'_i|\mu_{i,m}^c, \sigma_{i,m}^c))\Big)\Big) = \\ &\log\Big(\sum_{m=1}^{M_i^c} \exp\Big(\alpha_{i,m}^c - \frac{1}{2}(\frac{z'_i - \mu_{i,m}^c}{\sigma_{i,m}^c})^2\Big)\Big) \end{aligned} \tag{6}$$

where,

$$\alpha_{i,m}^c = \log(\gamma_{i,m}^c) - \log(\sigma_{i,m}^c) - \frac{1}{2}\log(2\pi) \tag{7}$$

$\alpha_{i,m}^c, \forall c \vee \forall i \vee \forall m$ is a fixed constant.

Subsequently, we calculate the weighted sum of these log-likelihoods using the representatives' importance coefficients. The w_i's are normalized first. For the importance coefficient of the representative R_i, we have:

$$\hat{w}_i = \frac{w_i}{\sum_{i=1}^k w_i} \tag{8}$$

Therefore, $\sum_{i=1}^k \hat{w}_i = 1$. The log-likelihood of classifying $\boldsymbol{z'}$ as class c is computed as:

$$\log(p(\boldsymbol{z'}|c)) = \sum_{i=1}^{k} \hat{w}_i \times \log(p_i(z'_i|c)) \tag{9}$$

Then, the predicted class for z, represented as ℓ_z, is explained as the class with the maximum total log-likelihoods:

$$\ell_{\boldsymbol{z}} = \max_c \log(p(\boldsymbol{z'}|c)) \tag{10}$$

It is essential to notice that we are **"not"** modeling the whole training set as a multi-dimensional (with features as the dimensions) multi-cluster distribution, and then calculating the likelihood of the new data belonging to each cluster (i.e., class), which is the basis of every clustering technique. Here, we take a completely different approach. We approximate the one-dimensional distribution of each representative and then calculate the likelihood of the new instance's representative values belonging to the corresponding representative distributions for each class. Afterward, by the weighted sum of these likelihoods for each class, we will have C likelihoods, each resulting from k distributions. Then, the class with the highest likelihood is picked as the explained class.

D. Selecting the Number of Modes

One important parameter in the TRUST explainer is the number of modes for MMG distribution for each representative. In general, this number is different for each representative, and it can be optimized to increase speed and performance. We use the grid search technique, which is a standard method in determining hyperparameters in machine learning techniques. Through grid search, for each representative, we define the number of modes that maximizes the probability of correctly distinguishing different classes' samples. The choice of M_i^c for each representative is important since it has a critical

Algorithm 3 Grid Mode Selection

Function: $\mathcal{GMS}$
Input: Representatives of all the classes:
$\{R_1^1, ..., R_k^1\} \in \mathbb{R}^{N_1 \times k}, \cdots, \{R_1^C, ..., R_k^C\} \in \mathbb{R}^{N_C \times k}$;
Zone $Z \in \mathbb{R}^{C \times C}$

$y \leftarrow [\text{ones}(N_1)^T, \ldots, C \times \text{ones}(N_C)^T]^T$
for each set of $(R_i^1 \cdots R_i^C)$ **do**
  $Max \leftarrow 0$
  for m^1 in Z **do**
    $\text{MMG}_i^1 \leftarrow$ fit R_i^1 to a Gaussian with m^1 modes
    $\vdots$
    for m^C in Z **do**
      $\text{MMG}_i^C \leftarrow$ fit R_i^C to a Gaussian with m^C modes
      $P_{m^1} \leftarrow p_1(R_i|\text{MMG}_i^1)$
      $\vdots$ ▷ $R_i = [R_i^{1T}, \ldots, R_i^{CT}]^T$
      $P_{m^C} \leftarrow p_C(R_i|\text{MMG}_i^C)$
      $P \leftarrow [P_{m^1} \cdots P_{m^C}]$
      $y_{\text{MMG}} \leftarrow \text{index}(P.\max(1))$
      $\text{score}_{m^1,\cdots,m^C} \leftarrow \text{MCC}(y_{\text{MMG}}, y)$
      if $\text{score}_{m^1,\cdots,m^C} > Max$ **then**
        $Max \leftarrow \text{score}_{m^1,\cdots,m^C}$
        $M_i^1 \leftarrow m^1$
        $\vdots$
        $M_i^C \leftarrow m^C$
      end if
    end for
    $\vdots$
  end for
end for
return $\{M_1^1, ..., M_k^1\} \cdots \{M_1^C, ..., M_k^C\}$

Algorithm 4 Fast Grid Search

Function: $\mathcal{FGS}$
Input: Representatives of all the classes:
$\{R_1^1, ..., R_k^1\} \in \mathbb{R}^{N_1 \times k}, \cdots, \{R_1^C, ..., R_k^C\} \in \mathbb{R}^{N_C \times k}$;
Zone $Z \in \mathbb{R}^{C \times C}$

$y \leftarrow [\text{ones}(N_1)^T, \ldots, C \times \text{ones}(N_C)^T]^T$
for each set of $(R_i^1 \cdots R_i^C)$ **do**
  Divide zone Z into $z_1, ..., z_d$
  $Max \leftarrow 0$
  for each z_j **do**
    $(m^1, \cdots, m^C) \leftarrow z_j$'s center
    $\text{MMG}_i^1 \leftarrow$ fit R_i^1 to a Gaussian with m^1 modes
    $\vdots$
    $\text{MMG}_i^C \leftarrow$ fit R_i^C to a Gaussian with m^C modes
    $P_{m^1} \leftarrow p_1(R_i|\text{MMG}_i^1)$
    $\vdots$ ▷ $R_i = [R_i^{1T}, \ldots, R_i^{CT}]^T$
    $P_{m^C} \leftarrow p_C(R_i|\text{MMG}_i^C)$
    $P \leftarrow [P_{m^1} \cdots P_{m^C}]$
    $y_{\text{MMG}} \leftarrow \text{index}(P.\max(1))$
    $\text{score}_{m^1,\cdots,m^C} \leftarrow \text{MCC}(y_{\text{MMG}}, y)$
    if $\text{score}_{m^1,\cdots,m^C} > Max$ **then**
      $Max \leftarrow \text{score}_{m^1,\cdots,m^C}$
      $Z_i \leftarrow z_j$
    end if
  end for
  $\mathcal{GMS}(\{R_i^1\}, \cdots, \{R_i^C\}, Z_i)$
end for
return $\{M_1^1, ..., M_k^1\} \cdots \{M_1^C, ..., M_k^C\}$

role in how well the fitted MMG distribution works as a density estimator. The pseudo-code of the method is shown in Algorithm 3. This algorithm searches over a zone called Z, which is simply a grid of integers starting from 1 to any potential number of modes.

As seen in this algorithm, dimension of the search space grows exponentially with the number of classes. Also, if the number of modes or classes is large, the search zone would be too large. Hence, the search could be very slow and time-consuming. We have developed a faster algorithm that divides the search zone into smaller sub-zones, Algorithm 4. At first, the probabilities of the centers of each sub-zone as the number of modes are calculated. Then the sub-zone with the highest score is chosen to search thoroughly. Even though this algorithm might not give us the exact number of modes, it provides a good accuracy (in the order of 99% in our experiments). With decreasing the size of sub-zones, the accuracy goes higher. Matthew's correlation coefficient (MCC), Eq. 12, is preferable to accuracy, Eq. 11, since MCC has been shown to be a better metric in learning using imbalanced datasets, which are common in the cybersecurity and other numerical applications [19].

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{11}$$

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{12}$$

where, TN is the number of normal data labeled as normal, TP is the number of attack data classified as attack, FP is the number of normal labeled as attack, and FN is the number of attacks labeled as normal by the model.

IV. An Example Using IIoT Security

Sophisticated AI models have been vastly applied in intrusion detection systems (IDS). XAI would add interpretability and trust to these systems. We have built a lab-scaled industrial Internet of Things (IIoT) system to collect realistic and up-to-date datasets. We have chosen a popular IIoT system that supervises the water level and turbidity of a water storage tank. This type of system is employed in industrial reservoirs and water distribution systems as a part of the water treatment and distribution. For more information regarding our testbed, we refer the readers to our previous papers [20] and [21].

A. Utilized Datasets

The first tested dataset is the one collected from our testbed, which we refer to is as "WUSTL-IIoT". To collect the proper dataset, we (as a white-hat attacker) attacked our testbed using manipulated commands, such as backdoor, command injection, denial of service, and reconnaissance [20].

Further, we have tested our TRUST XAI on two other cybersecurity datasets to add an empirical proof of concept,

TABLE II: Specifics of the three datasets

Dataset	WUSTL-IIoT	NSL-KDD	UNSW
# of observations	1,194,464	125,973	65,535
# of features	41	40	41
# of attacks	87,016	58,630	45,015
# of normals	1,107,448	67,343	20,520

"NSL-KDD" [22] and "UNSW" [23]. Specifications of these datasets are shown in Table II.

B. Underlying AI Model

Since TRUST is model-agnostic, any complex AI can be used as the underlying classifier. Here, we use an artificial neural network (ANN) with two hidden layers with 20 and 10 neurons respectively, and ten epochs as a binary classifier. The scikit-learn library [24] was used to implement the ANN model. ANN is generally considered complex and unexplainable to human users. The inputs to the IDS are the flow instances as mentioned in the previous subsection. The output of the IDS can be a multi-class or binary classification. However, for the sake of simplicity, we have used the outputs of the IDS as binary classes with normal as 0 and attack as 1 to develop the proposed explainer. Even though the performance of the underlying AI is not of our interest, Tables III, IV, and V show how it performed on the three datasets. The performance on the training set is presented in the (a) tables, while the (b) tables show the testing results.

TABLE III: Performance of the underlying AI on WUSTL-IIoT

	Normal	Attack
Normal	885980	12
Attack	131	69448

(a) On the training set

	Normal	Attack
Normal	221452	4
Attack	40	17397

(b) On the testing set

TABLE IV: Performance of the underlying AI on NSL-KDD

	Normal	Attack
Normal	53664	173
Attack	529	46412

(a) On the training set

	Normal	Attack
Normal	13452	54
Attack	138	11551

(b) On the testing set

TABLE V: Performance of the underlying AI on UNSW

	Normal	Attack
Normal	15791	657
Attack	473	35507

(a) On the training set

	Normal	Attack
Normal	3895	177
Attack	115	8920

(b) On the testing set

These tables have been brought up to emphasize once again that to develop the explainer, all XAI models use the labels outputted by the AI algorithm while ignoring the true labels. This is because, in the XAI domain, we care about explaining the model's behavior, not whether or not it has behaved very accurately.

The results of these tables based on the metrics including accuracy, MCC, and undetected rate (UR) using Eq. 11, Eq. 12, and Eq. 13 are summarized in Table VI.

$$\text{UR} = \frac{FN}{FN+TP} \tag{13}$$

TABLE VI: Summary of the underlying AI's performance

	Accuracy	**MCC**	**UR**
WUSTL-IIoT, Training	99.98%	99.88%	0.19%
WUSTL-IIoT, Testing	99.98%	99.86%	0.23%
NSL-KDD, Training	99.30%	98.60%	1.13%
NSL-KDD, Testing	99.24%	98.47%	1.18%
UNSW, Training	97.84%	94.98%	1.3%
UNSW, Testing	97.77%	94.78%	1.27%

C. Picking Representatives and Mode Selection

After running the $\mathcal{PR}$ function from Algorithm 2, we will end up with k representatives. This algorithm and the utilized factor analysis produces the eigenvalue of each factor that is equivalent to the amount of variation in the data that each factor explains.

However, as mentioned before, the percentage of explained variation cannot be used as the importance coefficient. The effectiveness of a factor in distinguishing the class label can be completely independent of the eigenvalues. Therefore, the MI scores between each representative and the class labels have been calculated to determine the top k representatives.

As previously discussed, the next step is to estimate the number of modes per representative. The general case with the C number of classes has been demonstrated in Algorithm 3. However, in our case, we have only two classes in our data, $C = 2$, therefore we have only two sets of representatives, $\{R_1^1, ..., R_k^1\}$ and $\{R_1^2, ..., R_k^2\}$. This modified algorithm only for two classes is shown in Algorithm 5. Also, the faster grid search algorithm for two classes is shown in Algorithm 6.

V. Explaining the TRUST Explainer

In this section, we discuss how the TRUST explainer can explain the labels to a human user simply and interpretably on the NSL-KDD dataset as an example. As mentioned before, it is safe to assume that the user of the AI-based security system has a basic knowledge of AI, mathematics, and statistics. Each traffic packet is a 40-dimensional vector. However, the explainer calculates the representatives and reduces each sample to a four-dimensional vector. This process of dimension reduction can be simply explained as follows.

TRUST modifies the feature values to remove redundancy in the data for more transparent statistical analysis. By factorizing and removing their interdependencies and projecting them onto orthogonal dimensions, TRUST removes any correlation in the data. The transformed feature values are called factors. Then, utilizing a correlation tool (here, mutual information),

Algorithm 5 Grid Mode Selection with $C = 2$

Function: $\mathcal{GMS}\_2$
Input: Representatives of class "1": $\{R_1^1, ..., R_k^1\} \in \mathbb{R}^{N_1 \times k}$;
Representatives of class "2": $\{R_1^2, ..., R_k^2\} \in \mathbb{R}^{N_2 \times k}$;
Zone $Z \in \mathbb{R}^{2\times 2}$

for each (R_i^1, R_i^2) **do**
  $Max \leftarrow 0$
  for m^1 in Z **do**
    $\text{MMG}_i^1 \leftarrow$ fit R_i^1 to a guassian with m^1 modes
    for m^2 in Z **do**
      $\text{MMG}_i^2 \leftarrow$ fit R_i^2 to a guassian with m^2 modes
      $P_{m^1} \leftarrow p_1(R_i|\text{MMG}_i^1)$ ▷ $R_i = \begin{bmatrix} R_i^1 \\ R_i^2 \end{bmatrix}$
      $P_{m^2} \leftarrow p_2(R_i|\text{MMG}_i^2)$
      $y_{\text{MMG}} \leftarrow P_{m^2} > P_{m^1}$
      $\text{score}_{m^1,m^2} \leftarrow \text{MCC}(y_{\text{MMG}}, \begin{bmatrix} \text{zeros}(N) \\ \text{ones}(N) \end{bmatrix})$
      if $\text{score}_{m^1,m^2} > Max$ **then**
        $Max \leftarrow \text{score}_{m^1,m^2}$
        $M_i^1 \leftarrow m^1$
        $M_i^2 \leftarrow m^2$
      end if
    end for
  end for
end for
return $\{M_1^1, ..., M_k^1\}$ & $\{M_1^2, ..., M_k^2\}$

Algorithm 6 Fast Grid Search with $C = 2$

Function: $\mathcal{FGS}\_2$
Input: Representatives of class "1": $\{R_1^1, ..., R_k^1\} \in \mathbb{R}^{N_1 \times k}$;
Representatives of class "2": $\{R_1^2, ..., R_k^2\} \in \mathbb{R}^{N_2 \times k}$;
Zone $Z \in \mathbb{R}^{2\times 2}$

for each (R_i^1, R_i^2) **do**
  Divide zone Z into $z_1, ..., z_d$
  $Max \leftarrow 0$
  for each z_j **do**
    $(m^1, m^2) \leftarrow z_j$'s center
    $\text{MMG}_i^1 \leftarrow$ fit R_i^1 to a guassian with m^1 modes
    $\text{MMG}_i^2 \leftarrow$ fit R_i^2 to a guassian with m^2 modes
    $P_{m^1} \leftarrow p_1(R_i|\text{MMG}_i^1)$ ▷ $R_i = \begin{bmatrix} R_i^1 \\ R_i^2 \end{bmatrix}$
    $P_{m^2} \leftarrow p_2(R_i|\text{MMG}_i^2)$
    $y_{\text{MMG}} \leftarrow P_{m^2} > P_{m^1}$
    $\text{score}_{m^1,m^2} \leftarrow \text{MCC}(y_{\text{MMG}}, \begin{bmatrix} \text{zeros}(N) \\ \text{ones}(N) \end{bmatrix})$
    if $\text{score}_{m^1,m^2} > Max$ **then**
      $Max \leftarrow \text{score}_{m^1,m^2}$
      $Z_i \leftarrow z_j$
    end if
  end for
  $\mathcal{GMS}\_2(\{R_i^1\}, \{R_i^2\}, Z_i)$
end for
return $\{M_1^1, ..., M_k^1\}$ & $\{M_1^2, ..., M_k^2\}$

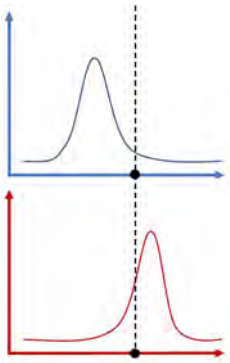


Fig. 4: Probability density functions

factors that are the most revealing in determining the class labels are chosen and called Representatives. Afterward, TRUST models the statistical behavior of the Representatives and estimates their density functions with unique multi-modal distributions.

As mentioned before, most XAI models are content with using a separate AI model for explainability, which means it is always assumed that the user knows how these AI models work internally. We will show later how models like LIME also take advantage of this assumption when it comes to explainability. However, we argue that statistics are more easily understood than AI. As an elaboration, shown Figure 4, comparing the likelihood of a new sample belonging to each class's probability density functions is easily understandable for the user. For instance, in this example, the likelihood of belonging to class attack is higher.

To show the explanation process of TRUST, we randomly chose six samples (three attacks, three normals) from the test set. The log-likelihoods of belonging to each class using individual representatives based on Eq. 6 are computed. The results of each instance belonging to class Attack, Table VIIa, and then belonging to class Normal, are shown in Table VIIb. After an element-wise comparison of the values in these two matrices, the class that has a higher likelihood is marked in Table VIIc. Notice that not all the representatives give us the right class all the time. For example, the 1st representative makes a mistake about the 4th sample. Similarly, the 3rd representative makes a mistake about the 2nd sample. However, when all four representatives are used to calculate the overall log-likelihood, Table VIId, the labeled class is correctly explained for all these instances.

A. Results

The number of representatives to use is based on the complexity constraints. Here, we start at 8. As shown later, we get high MCC scores even with 4 or 5.

Each dataset is split into training and testing sets with an 80:20 ratio. The TRUST XAI is built using the training set. Afterward, the testing samples' representatives are projected in the factor space computed using the training set. Then like-

TABLE VII: Log Likelihood

(a) Log probability of belonging to class Attack

	1st Rep.	2nd Rep.	3rd Rep.	4th Rep.
Attack samples	5.51	5.46	4.40	4.27
	1.56	-2.46	-1.19	1.54
	-1.79	-2.06	0.68	-3.06
Normal samples	-3.61	-2.09	-2.30	-5.21
	-6.99	-2.16	-3.12	-2.99
	-3.56	-2.19	-10.09	-4.72

(b) Log probability of belonging to class Normal

	1st Rep.	2nd Rep.	3rd Rep.	4th Rep.
Attack samples	-0.30	1.87	-0.04	-0.86
	0.08	-4.92	-1.11	-0.89
	-2.52	-2.16	-0.31	-6.18
Normal samples	-3.68	1.86	-2.11	-5.16
	-3.25	1.83	-2.76	-0.94
	-3.02	1.75	-2.41	-0.90

(c) Comparison of log probabilities

1st Rep.	2nd Rep.	3rd Rep.	4th Rep.
5.51 > -0.30	5.46 > 1.87	4.40 > -0.04	4.27 > -0.86
1.56 > 0.08	-2.46 > -4.92	-1.19 < -1.11	1.54 > -0.89
-1.79 > -2.52	-2.06 > -2.16	0.68 > -0.31	-3.06 > -6.18
-3.61 > -3.68	-2.09 < 1.86	-2.30 < -2.11	-5.21 < -5.16
-6.99 < -3.25	-2.16 < 1.83	-3.12 < -2.76	-2.99 < -0.94
-3.56 < -3.02	-2.19 < 1.75	-10.09 < -2.41	-4.72 < -0.90

(d) Total log probabilities

Total
19.64> 0.67
-0.55>-6.84
-6.23>-11.17
-13.21<-9.09
-15.26<-5.12
-20.56<-4.58

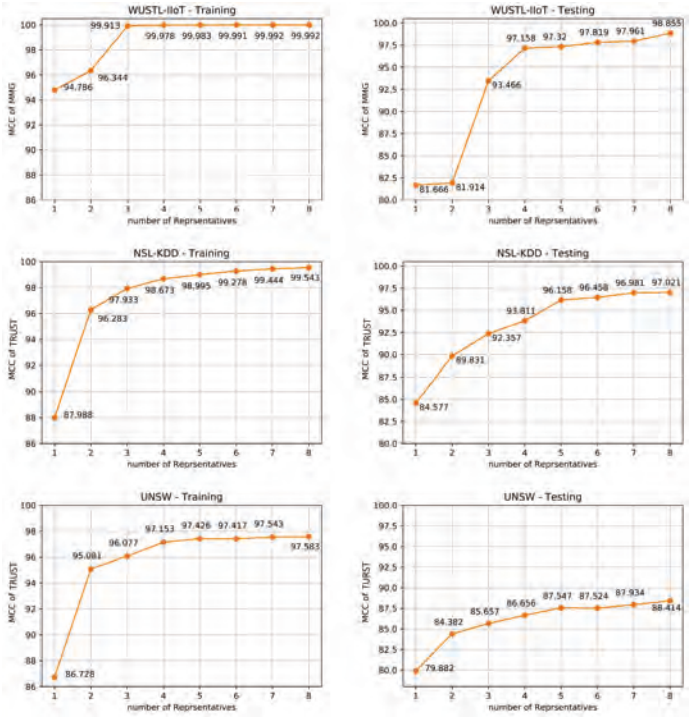


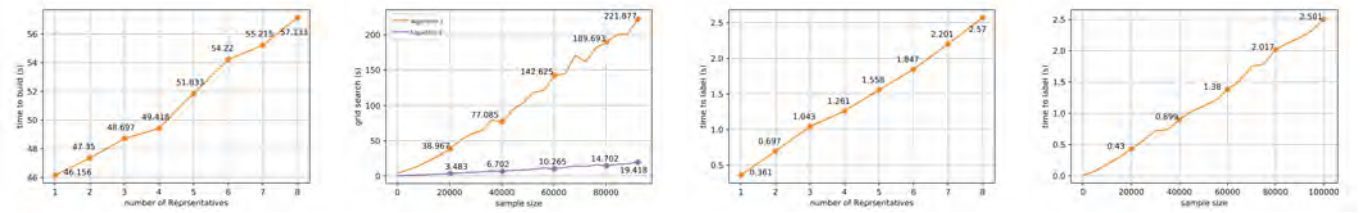
Fig. 5: MCC scores of TRUST XAI on the training sets and the testing sets

lihoods are calculated using the same representatives chosen from the training set.

Figure 5 shows the MCC results vs. the number of representatives for the training and testing sets. Note that MCC values are generally less than accuracy values. The results show TRUST is highly successful in explaining the AI's output on unseen data with an average of 90.89% MCC or 98% accuracy.

We have analyzed the performance in terms of time consumption, using the NSL-KDD dataset as an example. These tests were run on a standard laptop with an Intel Core i7 CPU, 16 GB RAM, and Windows OS (no GPU). The steps to build the TRUST model have to be done only once, and after that, the model can be easily used to explain the labels of future observations. Figure 6a shows the compute time to build the MMG model, including the factor analysis, picking the representatives, and estimating their density functions. The time to search for the number of modes is not included in this graph. Algorithm 3 and Algorithm 4 to determine the number of modes per representative is compared in Figure 7b. In this figure, the time to search for the number of modes of one representative is plotted. The search zone is a 20 by 20 grid, and when using our faster algorithm, it was divided into sixteen 5×5 sub-zones. As seen in this figure, Algorithm 4 speeds up the mode search by more than ten times. Next, the time for labeling test samples as a function of the number of representatives is plotted in Figure 6c. As seen in this figure, the compute time is pretty insignificant. Figure 6d shows the time as a function of the number of samples.

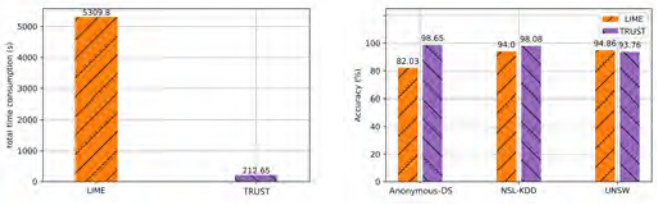
As mentioned before, our model has a significant advantage over other XAI models, such as LIME, that require repeating the whole process for every single sample. This makes their model very time-consuming and not efficient when dealing with a large number of samples. Our technique consists of a core model that, once it is built, it is done and ready to explain the labels of new samples quickly. Figure 7a demonstrates this claim. Note that the TRUST's time of 212.65 s (3.5 minutes) includes the time to build the model from the 100,778 training samples. The time includes factor analysis, picking eight representatives, running the mode grid search Algorithm 4 for them, estimating their density functions, and labeling the 8000 testing samples. Our model is almost 25 times faster than LIME, which makes a significant difference in real-time applications. Next, we have compared the performance of our model with LIME using the accuracy metric, Eq. 11. Here, we used only five representatives in our TRUST model and calculated the accuracy of the testing labels. As seen from Figure 7, our model has performed better even with only five representatives, except for UNSW, where LIME has a slightly higher accuracy (94.86% vs. 93.76%). If we had increased the



(a) (b) (c) (d)

Fig. 6: TRUST's Time consumption of (a) building the core model based on the number of representatives, (b) comparing Algorithm 3 and Algorithm 4, (c) labeling 100,778 samples vs. the number of representatives (d) labeling time vs. the number of samples. In the last two figures, the training set was used because it has a larger number of samples.

number of representatives, our model would be significantly superior to LIME in all cases. To compare the method of explainability in TRUST with LIME, we argue that our model is simpler to understand. As LIME is content with modeling the AI model with another model as a way of explainability, we model the statistical behavior of the AI's output. As we deal with probability theory frequently, statistics are much easier to understand compared to grasping the ideas behind AI models.



(a) (b)

Fig. 7: Explaining 8000 test samples; (a) time consumption of our TRUST model vs. LIME, (the TRUST's built time is also included), (b) performance of TRUST vs. LIME.

VI. Conclusion

There is a significant lack of research work in the domain of XAI, specifically for numerical applications. Meanwhile, several critical applications such as cybersecurity and IoT are highly dependent on numerical data. The available work in the literature falls short from several aspects, as discussed in this paper. Our proposed TRUST technique is a breakthrough in this domain to integrate transparency, independency, and comparability into AI-based systems. Here, we used statistical principles to build an XAI model that can accurately estimate the distributions of the AI's outputs and calculate the likelihood of new samples belonging to each class. Moreover, we have proven the superiority of our model to a currently popular XAI model, LIME. Our results show that our model is 25 times faster and more accurate than LIME. This helps our model be a great candidate for real-time and critical applications.

Acknowledgement

This work has been supported under the grant ID NPRP10-0206-170360 funded by the Qatar National Research Fund (QNRF) and NSF grant CNS-1718929. The statements made herein are solely the responsibility of the authors.

References

[1] Gartner. (2018) Gartner says nearly half of cios are planning to deploy artificial intelligence. Available at: https://www.gartner.com/en/newsroom/press-releases/2018-02-13-gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence. Date accessed: March 16, 2021.

[2] Webroot. (2017) Game changers: Ai and machine learning in cybersecurity. Available at: https://www-cdn.webroot.com/8115/1302/6957/Webroot_QTT_Survey_Executive. Date accessed: March 16, 2021.

[3] A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell, "What do we need to build explainable ai systems for the medical domain?" *arXiv preprint arXiv: 1712.09923*, 2017.

[4] C. Oxborough and E. Cameron. (2018) Explainable ai: driving business value through greater understanding. Available at: https://www.pwc.co.uk/xai. Date accessed: March 16, 2021.

[5] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[6] C. Molnar, *Interpretable machine learning. A guide for making black box models explainable*, 2018, available at: https://christophm.github.io/interpretable-ml-book/. Date accessed: March 16, 2021.

[7] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys*, vol. 15, pp. 94:1–94:45, 2017.

[8] M. Wang, K. Zheng, Y. Yang, and X. Wang, "An explainable machine learning framework for intrusion detection systems," *IEEE Access*, vol. 8, pp. 73 127–73 141, 2020.

[9] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774.

[10] K. Amarasinghe, K. Kenney, and M. Manic, "Toward explainable deep neural network based anomaly detection," in *11th IEEE International Conference Human System Interaction (HSI)*, Gdansk, 2018, pp. 311–317.

[11] L. Ma, J. Gao, Y. Wang, C. Zhang, J. Wang, W. Ruan, W. Tang, X. Gao, and X. Ma, "Adacare: Explainable clinical health status representation learning via scale-adaptive feature extraction and recalibration," in *AAAI Conference on Artificial Intelligence*, ser. 01, vol. 34, 2020, pp. 825–832.

[12] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Toward medical xai," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020.

[13] D. Marino, C. Wikramasinghe, and M. Manic, "An adversarial approach for explainable ai in intrusion detection systems," in *44th Annual Conference of the IEEE Industrial Electronics Society, IECON 2018*, Washington DC, 2018.

[14] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?? explaining the predictions of any classifier," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, 2016, p. 1135–1144.

[15] J. Pages, *Multiple Factor Analysis by Example Using R*. New York: Chapman and Hall/CRC, 2015.

[16] R. Jain, *The Art of Computer Systems Performance Analysis*. Wiley Interscience, 1991.
[17] S. Maxwell, H. Delaney, and K. Kelley, *Designing Experiments and Analyzing Data*, 3rd ed. Routledge, 2018.
[18] A. Ng. (2018) The em algorithm. Available at: http://cs229.stanford.edu/notes/cs229-notes8.pdf. Date accessed: March 16, 2021.
[19] M. Zolanvari, M. A. Teixeira, and R. Jain, "Effect of imbalanced datasets on security of industrial iot using machine learning," in *IEEE International Conference on Intelligence and Security Informatics (ISI)*, Miami, FL, 2018.
[20] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial internet of things," *IEEE Internet of Things Journal*, vol. 6, pp. 6822–6834, 2019.
[21] M. A. Teixeira, T. Salman, M. Zolanvari, R. Jain, N. Meskin, and M. Samaka, "Scada system testbed for cybersecurity research using machine learning approach," *Future Internet*, vol. 10, pp. 1–15, 2018.
[22] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, 2009, pp. 1–6.
[23] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, ACT, 2015, pp. 1–6.
[24] Pedregosa et al., "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.