DeepCAD: A Stand-alone Deep Neural Network-based Framework for Classification and Anomaly Detection in Smart Healthcare Systems

Nur Imtiazul Haque*, Mohammad Ashiqur Rahman*, and Sheikh Iqbal Ahamed†
*Analytics for Cyber Defense (ACyD) Lab, Florida International University, FL, USA
†Department of Computer Science, Marquette University, WI, USA
*{nhaqu004, marahman}@fiu.edu, †sheikh.ahamed@marquette.edu

***Abstract*—Contemporary smart healthcare systems (SHSs) frequently use wireless body sensor devices (WBSDs) for vital sign monitoring and the internet of medical things (IoMT) network for rapid communication with a cloud-based controller. The SHS controllers generate required control decisions based on the patient status to enable real-time patient medication/treatment. Hence, the correct medical delivery primarily depends on accurately identifying the patient's status. Accordingly, SHSs mostly leverage deep neural network (DNN)-based machine learning (ML) models for patient status classification due to their prediction accuracy and complex relation capturing capability. Nevertheless, the open IoMT network is prone to several cyberattacks, including adversarial ML-based attacks, which can exploit DNN models and create a life-threatening event in a safety-critical SHS. Existing solutions usually propose outlier detection or transfer learning-based ML models on top of the patient status classification model to deal with SHS security issues. However, incorporating a separate anomaly detection model increases the model complexity and raises feasibility issues for real-time deployment. This work presents a novel framework, DeepCAD, that considers training a stand-alone DNN model integrated with anomaly detection rules for classification and anomaly detection in SHS. The proposed framework is verified on the Pima Indians Diabetes and Parkinson datasets.**

***Index Terms*—Healthcare security, machine learning, internet of medical things, patient status classification, anomaly detection**

I. Introduction

A smart healthcare system (SHS) is a part of a broad multidisciplinary concept, the digital healthcare system (DHS) [1]. The SHS concept is reshaping the modern healthcare system by making it more personalized, automated, and effective by incorporating the internet of medical things (IoMT)-enabled network and machine learning-based control decision system. Patients are automatically treated with implantable medical devices (IMDs) or another automated medical delivery system in a typical SHS. The medical actuators (i.e., IMDs or so) are triggered by a control signal, which is primarily provided by a cloud-based controller. The controllers use an ML-based algorithm to determine the patient's status through the sensor measurements collected from the wireless body sensor devices (WBSDs) connected to the patients' body. The measurements from WBSDs are sent to the controller through the IoMT network. The importance of SHS was more clear during the COVID-19 pandemic period. A large number of people got delayed or denied healthcare services during this pandemic period since there was a sharp difference between hospital accommodation and affected patients [2]. The avoidance or delay in medical care significantly contributed to the surge of patients' death due to COVID-19 [3]. Almost 48% of COVID-affected Americans were either delayed or denied medical treatment during the pandemic period [4]. 11% of the patients experienced deteriorated health conditions because of treatment latency. Global acceptance of automated SHS would have reduced such unexpected events by enabling remote patient monitoring and treatment using ML-based controllers. The prevalence of sufficient healthcare data supported by the statistics stating the availability of more than 2,000 exabytes of data related to healthcare is making the patient status classification ML models more reliable [5]. The deep neural network (DNN) models are particularly getting popular due to their intricate pattern identification, nonlinear boundary acquisition, and inter-feature relationship capturing capability [6], [7].

However, the DNN model, being trained on only benign samples, classifies each and every possible sample as the learned classes. Hence, sensor measurement manipulation through knowledgeable adversarial attempts can misinform the controller with wrong information, leading to inaccurate patient status identification and thus wrong medication/ treatment. Therefore, the incorporation of ML models, despite capturing critical relationships among sensor measurements, is susceptible to several threats to be exploited by various stealthy cyberattacks [8], [9]. The open IoMT network communication is growingly increasing the possible cyberattacks in a safety-critical SHS [10]. The feasibility of exploiting healthcare sensor devices is revealed by a recent study, dictating that more than two-thirds of IoMT devices are vulnerable to several cyberattacks [11]. The SHS is exploitable with many attacks, as found in recent literature. The attacks include hardware Trojan [12], man-in-the-middle (MITM) attacks [13], malware (e.g., Medjack [14]), Sybil attacks (using either hijacked IoMT [15], denial of service attack [16] and so on. More than half of the healthcare organizations have been affected by adversarial attacks from October 2018 to October 2019 [17]. A report states that the University of Vermont Medical Center got

isolated from network connection due to a cyberattack, which is estimated to incur $64 million loss [18]. The vulnerability of IoMT-enabled SHS has also been identified by the statistics determining 6.2 (out of 10) cybersecurity vulnerabilities in 15-20 connected IoMT devices [19].

Adversarial ML-based attacks have become a severe concern for safety-critical systems. A knowledgeable adversary having information about the SHS ML model can launch Whitebox or BlackBox adversarial ML-based attacks. Adversarial ML-based attacks find the vulnerability of the ML models and can reveal attack paths to launch targeted or untargeted attacks with minimal alteration [20]–[24]. However, several solutions have been proposed to detect adversarial alteration in ML-based systems. The proposed solutions mostly adopt transfer learning, ensembled ML, and outlier detection techniques [25]–[27]. However, these techniques increase the model complexity and thus face control decision generation latency. In a safety-critical system like SHS, a minor delay can cost patients life. Hence, in this work, we attempted to develop a single DNN-based framework naming **D**eep neural network-based **C**lassification and **A**nomaly **D**etection (DeepCAD). The proposed framework performs both classification and anomaly detection by adding anomaly detection rules into a DNN model. The rule addition in a complex DNN model is not straightforward. We have developed a novel DNN model training approach with a modified loss function to determine out-of-date distribution samples accurately. The work is a proof of concept and is implemented to add circular boundaries to the two-feature classification model. We have verified the proposed framework on a state-of-the-art dataset - Pima Indians Diabetes dataset [28]. The summarized contributions are presented followingly:

- We developed a DNN-based framework that can perform both classification tasks with anomaly detection without being trained with the anomalous sample. The proposed framework does not require a separate model for anomaly detection.
- The robustness of the DNN model of the DeepCAD framework is analyzed against adversarial ML models. Moreover, We have verified our framework on a real dataset.

The rest of the paper is organized as follows: we present the overview of an SHS and other necessary background information for the reader's comprehension in Section III. In Section II, we present a comprehensive overview and differences from the existing literature. The DeepCAD framework overview can be found in Section IV, and a detailed technical aspect of the framework is explained in Section V. Moreover, we evaluate our proposed framework by running experiments on a synthetic dataset and a real dataset and present the results in Section VI. Section II provides a literature review to show differences with the existing state-of-the-art techniques. Finally, Section VII provides conclusion and future extension.

II. Related Works

Anomaly and patient status classification models of safety-critical systems like SHS are drawing significant research focus nowadays. The classification models for SHS are mainly using supervised ML models. The DNN-based classification models are particularly being used for their complex nonlinear boundary extraction capability [29]–[31]. However, these DNN models are mostly trained with benign samples. A few works consider anomalous samples to train the DNN model. However, it is almost impossible to capture the anomalous sample distribution from a set of anomalous samples due to the increasingly growing attacks. Hence outlier and transfer learning-based anomaly detection techniques are increasing in popularity [25]–[27]. One of the mention-able works is presented by Haque et al. [25], where they have proposed ensembled unsupervised ML models with an autoencoder and one-class support vector machine to perform abnormality detection. In another work, they have proposed two separate ML models for classification and anomaly detection [26]. They proposed a novel fitness function calculation for anomaly detection using bio-inspired computing-based hyper-parameter optimization in that work. However, all these models demonstrate significant complexity and in a critical safety system like SHS, minor latency can cost patients lives. Similarly, the transfer learning-based trained model also comes up with a complex model. Hence, a single model embedding the rules of classification and anomaly detection is direly needed to reduce the decision-making latency in the SHS controller end.

Several types of research have been conducted for developing anomaly detection models for the healthcare system. Alrashdi et al. developed an ensembled online sequential extreme learning machine (EOS-ELM) for identifying abnormalities and malicious activities in fog-based internet of things (IoT)-enabled healthcare system [32], [33]. Newaz et al. presented a novel ML security framework for detecting adversarial attacks in an SHS that attempts to craft the observed vital signs collected from the connected IoMT devices and thus endanger the patient's health condition [34]. Al Shorman et al. employed one-class SVM (OCSVM) to provide a new mechanism for identifying IoT botnet attacks, using grey wolf optimization (GWO) to optimize the underlying hyperparameters. [35], [36]. In transfer learning, the knowledge of an already trained machine learning model is applied to a different but related problem. Zhao et al. performed transfer learning-based anomaly detection in networks motivated by the fact that "most network attacks belong to variants of known network attack families and share common features, which suggests a good fit for applying transfer learning" [27], [37].

The embedding of classification and anomaly detection rules in a single model is needed to reduce treatment latency. There exist several research attempts to embed rules in the ML model [38]–[40]. The attempts made by Ganchev et al. [39] and Hu et al. [40] mainly use a regularization process to embed rules into ML models. The regularization process can control the DNN provided boundary. However, they cannot

be applied to impose any rules on the DNN model. Moreover, Seo et al. came up with a state-of-the-art DNN rule embedding framework, DeepCTRL, in which the robustness of rules can be controlled in the inference phase [38]. The imposed rules in the DeepCTRL model follow a specific format. Hence, to the best of our knowledge, no existing work can directly add anomaly detection rules to the DNN-based classification model.

III. Preliminaries

In this section, we discuss SHS, DNN, minimum enclosing circle, adversarial machine learning, and threat model to ease following the rest of the paper.

A. IoMT-enabled Smart Healthcare System (SHS)

The connected smart medical device network of SHS is popularly identified as the IoMT network. Due to the advancement of the medical domain through cost reduction and consultation accuracy, IoMT-enabled SHS is regarded as a game-changer for the medical field. The research community is getting benefited from carrying out statistical analyses of diseases and medication patterns due to the prevalence of an enormous amount of medical data. The engagement of IoMT in the healthcare domain enabled increasing interest in developing universal data acquisition solutions to obtain and analyze data from decentralized sources [41], [42]. In a typical IoMT network, sensor measurements are sent to a cloud server for control decision-making since the sensor devices and IMDs are incapable of processing a large amount of data. Moreover, local decision control would raise several security and implementation issues. In this work, we consider an IoMT-enabled SHS that uses WBSN, ML-based patient status classification model, and IMD-based actuators. Figure 1 shows a sample IoMT-enabled SHS, where a patient diagnosed with both COVID-19 and diabetes is being monitored through WBSDs (e.g., blood glucose, cholesterol, pulse oximeter) is being provided with automatic medical delivery through ventilator and insulin pump.

The WBSDs attached to the patient body uninterruptedly monitor the patient vital signs. These observed measurement/s from the sensor devices are delivered to the ML-based cloud controller using various wireless communication protocols (e.g., WiFi, Bluetooth, Zigbee, and so on). The controller takes decisions based on the reported/received sensor measurements and sends control commands to the IMDs to deliver the necessary treatment to the patients. For instance, in the provided example (Figure 1), if the controller figures out that the patient needs emergency insulin delivery or immediate ventilation, it notifies the responsible insulin pump implanted inside the patient's body or ventilator to inject the proper amount of insulin or oxygen.

B. Cyber Attacks in SHSs

The SHSs are exploited with various malware and man-in-the-middle (MITM) attacks. Medical Device Hijacking (MEDJACK) is a recent malware threat that targets healthcare

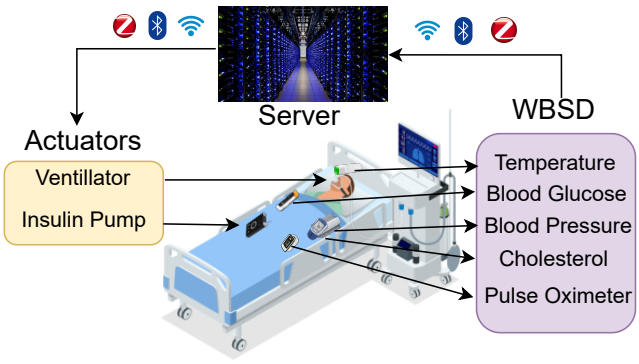

Fig. 1. An IoMT-based SHS for COVID and diabetes patient treatment.

systems by injecting malware into IoMT networks. [14]. It is a stealthy cyber-attack that employs the concept of polymorphic malware to continuously increase its capacity, making it extremely difficult to detect. MEDJACK achieves network access without being noticed by the system administrator by building a backdoor behind the firewall. On the other hand, MITM is a cyber-attack in which an adversary gains unauthorized access to a communication between two authorized parties and eavesdrops on or corrupts the data being transmitted. In sensor networks, Bluetooth-enabled medical devices have potential vulnerabilities, which is affirmed by Pournaghshband et al. [13] since they demonstrate the feasibility of launching a MITM attack in a Bluetooth-enabled pulse oximeter. They reverse-engineered a Nonin Onyx II 9550 fingertip pulse oximeter, which can measure blood oxygen saturation and pulse rate. MITM attacks on wireless networks can be carried out in various ways, such as by disrupting Bluetooth pairing with devices or access points (APs).

C. Deep Neural Network (DNN)

DNN model is the most popular supervised ML model. The difference between the DNN model and the regular neural model is that in a DNN model, multiple hidden layers exist between input and output layers. The DNN model is developed inspired by the working principle of the human brain, which is capable of doing numerous tasks in a parallel fashion without degrading system performance [43]. The key feature of the DNN model is its intricate pattern identification ability from high-dimensional data representation. The best use of DNN can be observed in a multiclass classification problem. However, in this work, for simplicity, we consider a binary classification problem to be solved by DNN. The DNN model training demands a lot of tuning, such as learning rate, batch size, number of hidden layers, etc. Our proposed DeepCAD framework leverages a feed-forward DNN that optimizes the considered loss function using an adam optimizer [44].

In a DNN, a set of nodes are arranged in a network in a particular manner depending on the application domain. The network comprises several layers, from the input to the output layer, and quite a few hidden layers in between. The DNN

model performance mainly depends on the number of hidden layers and associated nodes. For every node of different layers of the DNN model, the input is produced by the sum of products of weight and output of the previous nodes added with the bias work. The output of each node is passed through an activation function (i.e., ReLU, tanh, etc.) to obtain a non-linear boundary from the DNN model. These activation functions play a crucial role in non-linearly mapping between the input features and target. Initially, the weights and biases are assigned randomly. Later, these model parameters are tuned to reduce the error between the model prediction and target through the backpropagation process. Learning rate is another DNN model hyperparameter that controls the model parameters' magnitude in response to the estimated error. Moreover, the DNN model utilizes a regularization process to deal with the over-fitting issues.

In this section, we provide the formal definition of the DNN model for better understanding. The input of each node at any layer except the input layer is calculated using Equation 1.

$$\forall_{m\in(1,N]}(\mathcal{N}_{m,n}^{input}) = \sum_{o=1}^{|\mathcal{N}_{m-1}|} ((\mathcal{N}_{(m-1),o}^{output})\times \mathcal{W}_{m,o,n}) + \mathcal{B}_m \quad (1)$$

Here,
N denotes the number of layers of the DNN model.
$\mathcal{N}_{m,n}^{input}$ signifies the input to n-th node of m-th layer from a set of all node, N,
$\mathcal{W}_{m,o,n}$ indicates the weight of the interconnecting node n (from m-th layer) and o (from (m-1)-th layer),
$\mathcal{B}_m$ is the bias at mth layer.

Since there in no node prior to input layer, the input and output of the input layer is same as input sensor measurements as shown in Equation 2.

$$\mathcal{N}_1^{input} = \mathcal{N}_1^{output} = \mathcal{S} \quad (2)$$

Here,
$\mathcal{S}$ is the set of input sensor measurements.

D. Minimum Enclosing Circle (MEC)

We consider the smallest possible circle, which covers the minimum possible area, as our anomaly detection boundary. The idea is to enclose the data distribution with a minimal circular boundary so that the samples coming out of the boundary are labeled as an anomaly. The boundary is known as the minimum enclosing circle (MEC). An English mathematician James Joseph Sylvester was the first to propose the problem. The problem can be considered a generic problem in n-dimensional space to enclose high-dimensional features using n-sphere. Welzi's algorithm is one of the efficient recursive approaches to solve the MEC problem, using which the MEC is attainable in O(N). The aforementioned observations are the base for Welzi's algorithm. The key idea is to stochastically choose a point to be removed from the input dataset to form a circle equation. After forming the equation, it is checked whether the removed point is bounded by the equation or not. On failure, it is depicted that the point in interest must lie in the MEC boundary. Hence, this point is regarded as a boundary point, and the function is recursively called repeatedly.

E. Adversarial Machine Learning

Adversarial machine learning is an ML technique that tries to exploit models by utilizing publicly available model information to create malicious attacks using deceptive inputs to the ML model. This is considered to be the most typical reason for causing an ML model to malfunction. Adversarial ML attacks can be of two kinds - BlackBox and white-box attacks.

Adversarial Example The principal objective of the adversarial ML attack is to generate adversarial examples. Therefore, an adversarial example is an input to an ML model that is deliberately devised to cause an ML model, particularly DNN, to provide wrong predictions. The alteration is that adversarial samples are imperceptible to a human.

Adversarial Goals In an adversarial ML attack, an adversary's primary goal is to create adversarial sample creation. The adversarial goals can be categorized into three categories based on the impact on the classifier output integrity - targeted attack (i.e., adversarial sample prediction should be a specific label), untargeted attack (i.e., adversarial sample prediction can be any label other than the actual label), targeted device attack (i.e., the adversary tries to devise the minimum number of devices to compromise for attaining attack goal). Since the considered dataset has only two classes in this work, we consider the attack goal to launch an untargeted attack.

Adversarial Capabilities In order to perform the adversarial attack on the ML model, the following capabilities are considered for the adversary:

- Data Distribution: Since different smart WBSDs produce different types/ranges of data, an attacker may only have a partial understanding of the devices' data distribution. The attacker may adjust a data value within a specific threshold based on this to alter a disease-affected patient's status to a normal one, which will interrupt medication/ treatment.
- SHS Architecture: To carry out an attack, an adversary may have total or partial knowledge about the SHS architecture, including the number of devices, device correlation, and so on.
- Output Label: To launch an attack, an adversary may have knowledge of the DNN model's output labels (e.g., disease statuses and normal status).
- ML Model: The ML model (in our case, the DNN model) and corresponding architecture is known to the attacker.

The adversarial ML-based attack can be broadly classified into two groups - evasion and poisoning attack. We consider a state-of-the-art adversarial ML technique, naming the fast gradient sign method (FGSM) attack for this work.

Fast Gradient Method (FGM) Attack: To find adversarial examples, the fast gradient approach employs the gradient of the underlying model [45]. With the goal of changing the

behavior of the learning model, the original input is altered by adding or deleting a minor error in the gradient direction. In our model, an attacker's capability (threshold) was added as a minor error in the gradient direction to temper the model's classification.

An intuitive demonstration of the FGSM attack is provided in Figure **??** where with minor alteration, a Macaw is misclassified as a Bookcase.

IV. Framework

In this section, we present an overview of the proposed framework. The workflow of the framework is shown in Fig. 2. A synthetic dataset is prepared for an intuitive understanding of the workflow. The dataset contains two classes of samples- positive and negative. The samples from the positive class are colored green, and the negative class samples are colored red. The principal task is to classify the positive and negative samples correctly. It is clear from the figure that the samples are separable with a simple boundary, and a regular DNN model can accurately perform the classification task. However, since the model will be trained on two classes of samples, the trained DNN model will classify any data points into those classes, irrespective of their belonging to the data distribution. Our proposed framework solves the problem by classifying out-of-distribution samples as anomalies. We consider the data is distributed in a minimal circular boundary, and our framework attempts to impose that boundary constraint in the trained DNN model. The framework workflow is divided into two steps described as follows.

Data Processing The framework, at first, takes the prepossessed data as input. All data are then passed to Welzi's algorithm to obtain a minimal circle that can enclose all the data points (i.e., a circle with the least possible area that can bound all data points). Initially, the framework encodes the label of the positive sample as (1, 0) and negative samples as (0, 1). However, the labels are scaled and normalized after the boundary acquisition according to the circle center/ boundary distance. From the Fig. 2, we can see that after scaling and normalization, the positive samples close to the boundary are labeled close to (0.5, 0), whereas the negative samples are labeled close to (0, 0.5). Similarly, the positive samples near the circle center will be labeled closed to (1, 0), and the negative samples will be labeled close to (0, 1). The points in between will be labeled accordingly as ([0.5-1], 0) and (0, [0.5-1]) for positive and negative samples, respectively. The scaling and normalization process is further explained in Section V.

Model Training As discussed in the data processing step, instead of labeling the positive and negative samples with a single label, the labels are encoded into two labels. Although the problem is a classification problem, the DNN model of our framework views the problem as a regression problem. The DNN model takes the preprocessed data as its input. There are 2 input features and 2 output labels for the DNN model. The number of layers and the number of hidden layer nodes are later tuned for optimal model acquisition. From the figure, we can see that after initial passing (without training), the model misclassified almost each and every sample. The DNN model is trained for several epochs, and the weights of the DNN model are tuned in each epoch based on loss calculation. We consider the mean square error loss function in our case to determine the mean difference between the actual and predicted samples. The loss calculation is also supplemented by calculating errors between out-of-boundary verification samples with their model prediction (i.e., verification loss function). The later loss function is passed through a ReLU activation function. The framework, in the end, outcomes the model that has converged (i.e., the training samples are accurately predicted) or reached a pre-specified threshold. The ultimate goal of the model is to label them out of circle samples as ([0 - 0.5), [0 - 0.5)) so that after rounding, the labels are converted into (0, 0). The verification loss function helps embed the rule into the DNN model to predict labels for the out-of-boundary samples to be fewer than 0.5.

V. Technical Details

In this section, we provide a detailed description of the DeepCAD framework. The DNN model of the framework can perform the classification and anomaly detection together. The algorithm of the overall process is demonstrated in Algorithm 1. We discuss line by line the explanation of the algorithm. The overall process can be divided into two steps which are described as follows.

A. Data Preprocessing

The data is comprised of features and labels. In this work, we consider only binary classification. Hence, we consider two classes - positive and negative for the model prediction. The data preprocessing consists of 3 steps- feature preprocessing, boundary acquisition, and label preprocessing.

Feature Preprocessing As discussed before, this is a proof of concept representation, and we consider the input features are 2-dimensional. Hence, for high dimensional feature space, we reduce the feature space into 2-dimension using principal component analysis or other dimension reduction techniques. Currently, the framework works for the datasets, where the features can be reduced into 2-dimension with minimal loss of information. The framework takes the preprocessed features as input.

Boundary acquisition We consider that the anomaly detection rules are encoded by enclosing the data distribution within a boundary. Our goal is to label a data point residing out of the boundary as an anomaly. Line 16 of the Algorithm 1 uses $Welzi$ function to enclose the features in a minimal circle. The purpose of the $Welzi$ function is to obtain a MEC. The returned circle is described by the circle's center and radius, where $(circle^X, circle^Y)$ is the coordinate for the circle center and $circle^{radius}$ is the circle radius.

Label Prerocessing The label preprocessing is the principal data preprocessing step. The success of accurate anomaly detection of DeepCAD depends on label preprocessing. The label processing is demonstrated in Line 17-19. The initial labels are encoded as (l_1, l_2). For the positive sample, $l_1 = 1$

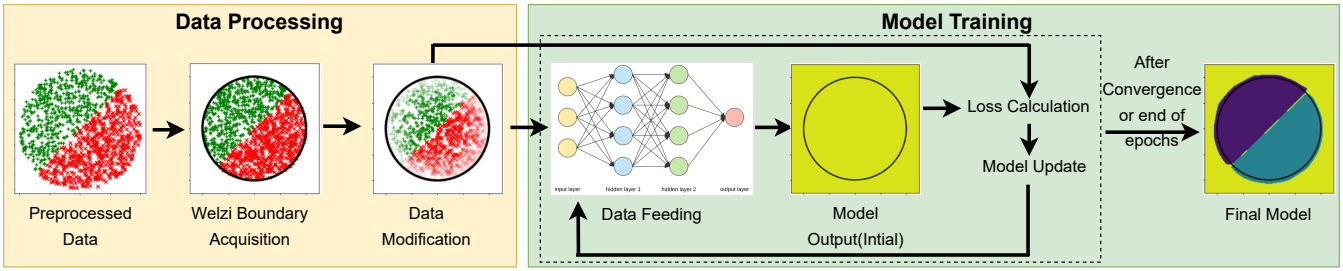

Fig. 2. Framework

Algorithm 1: DNN Model Training with DeepCAD.

1 **Function** `VerificationSampleGeneration`(*circle, numVerSamples, threshold*) **:**
2 $thetas \leftarrow GenTheta(0, 2*\pi, numVerSamples)$;
3 $verificationSamples \leftarrow Null$;
4 **for** *i in Range(numVerSamples)* **do**
5 $vSample^X \leftarrow circle^X + circle^{radius} + threshold * cos(theta_i)$;
6 $vSample^Y \leftarrow circle^Y + circle^{radius} + threshold * sin(theta_i)$;
7 $verificationSamples$.append($vSample^X$, $vSample^Y$)
8 **end**
9 **return** $verificationSamples$**;**
10
11 **Function** `LossFunction`(*labels, predictions, verificationPredictions, alpha, verThreshold*) **:**
12 $loss \leftarrow alpha$ * MSE($labels$, $predictions$) + (1 - $alpha$) * mean(ReLU($verificationPredictions$ - $verThreshold$));
13 **return** $loss$**;**
14
15 **Function** `ModelTraining`(*features, labels, epochs, numVerSamples, threshold, alpha, verThreshold, optimizer*) **:**
16 $circle \leftarrow Welzi(features)$;
17 $diff \leftarrow Distance(features, circle^{radius})$;
18 $scaler \leftarrow (0.5 + circle^{radius} - diff)$;
19 $labels \leftarrow labels * scaler$;
20 $verificationSamples \leftarrow VerificationSampleGeneration(circle, numVerSamples, threshold)$
21 **for** *i in Range(epochs)* **do**
22 $predictions \leftarrow Pred(model, features)$;
23 $verificationPredictions \leftarrow Pred(model, verificationSamples)$;
24 $loss \leftarrow LossFunction(labels, predictions, verificationPredictions, alpha, verThreshold)$;
25 Update model parameters using optimizer to minimize the $loss$
26 **end**
27 **return** $model$**;**

and $l_2 = 0$, while for the negative sample, $l_1 = 0$ and $l_2 = 1$. At first, the euclidean distance from each and every feature is calculated from the circle obtained in the boundary acquisition step and put in the $diff$ vector. All values in the $diff$ vector is in the range in the range$[0 - circle^{radius}]$. Line 18 shows the $scaler$ vector calculation process, which is in the range in the range$[0.5 - 1]$. The scaler values for the points that are closer to the center are close to 1, whereas the $scaler$ values for the points that are closer to the boundary are close to 0.5. In line 19, the labels are multiplied by the scaler vector. Hence, the labels of the positive points at the center turn in (1, 0), while the negative ones are scaled to (0, 1). The labels of the positive points on top of the circle circumference turn in (0.5, 0), while the negative ones are scaled to (0, 0.5). Let's look into a case study for the calculation. Say the MEC's center is located at (2, 2), and the radius of the circle is 2. A point located at (1, 0.8) will have a $diff$ value of 1.56205 for that particular point, and the $scaler$ value at the index of the point in consideration will be 0.93795. Now, if the point is positive, the modified label will be (0.93795, 0), while for the negative point, the modified label will be (0, 0.93795). The goal of this approach is that the points outside the boundary will be labeled as ([0-0.5), [0-0.5)), so that after rounding the label of the samples will be (0, 0). Table I shows the expected model outcome for different labels. It is to be noted that the DeepCAD DNN model should not outcome any label in range ([0.5 -1], [0.5 - 1]) since the rounded prediction will result in (1, 1), which is neither a positive nor benign sample. This type of outcome is only possible when the model is not well trained, which is also a good indicator of convergence.

TABLE I
DECISION MADE BY DEEPCAD DNN MODEL FOR DIFFERENT LABELS.

Label 1	Label 2	Decision
0	0	Anomaly
1	0	Positive
0	1	Negative
1	1	Invalid

B. Model Training

The DNN model training of the DeepCAD framework consists of 4 steps- DNN model configuration, verification sample generation, loss calculation, and model tuning.

DNN Model Configuration After processing the data, the DNN models are configured. In the current stage, we consider two input and two output labels. Based on the non-linearity of the optimal boundary, we add layers and corresponding nodes in the layer. The input samples are passed through the model gets, multiplied with weights, added with biases, and processed by activation functions. The goal of the model configuration is to create a DNN model architecture from which the positive, negative, and anomalous samples can be correctly classified after proper tuning of weights and bias.

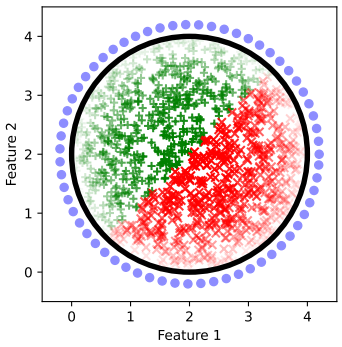

Fig. 3. Verification Samples (Outside blue circles). The positive samples are colored as green '+' and the negative samples are labeled as red 'x'. The black circle is the MEC.

Verification Sample Generation The proposed DeepCAD framework creates several verification samples outside the boundary, which helps to label the anomalous samples correctly. The absence of verification samples in the model training starts misclassifying anomalous samples at a certain distance from the boundary. The verification sample generation process is shown in Line 2-7. The $verificationSampleGeneration$ function takes the MEC circle properties ($circle$), several verification sample to create ($numVerSamples$), and the distance from the boundary to create samples ($threshold$) as input. We create samples circularly distributed around the MEC. We generate angles(in radian) using the $GenTheta$ function around the MEC from the positive x-axis. Line 4-7 uses the $theta$ angles and the $threshold$ distance to get the exact position of the verification samples, and that's how the $verificationSamples$ vector is generated. Figure 3 shows the verification samples from the synthetic dataset for the visualization purpose.

Loss Calculation and Model Tuning The DNN model parameters (weights and the bias) are tuned based on the loss function. For obtaining optimal classification and anomaly detection performance, the loss at each epoch is calculated using two different loss functions. Line 12 of Algorithm 1 shows the loss calculation process. The loss calculation for the regular samples and verification samples are a bit different. Before the loss calculation, the prediction from the model is calculated for both regular and verification samples through a forward pass and fed into $predictions$ and $verificationPredictions$ respectively (Line 22, 23). The loss calculation for the regular samples uses the mean squared error in between the $prediction$ and actual label. The loss calculation for the verification samples uses mean ReLU error. We know that ReLU is an activation function that takes negative values as input and outputs zero while taking positive values as input outputs the same value as input. We can see from Line 11 that $LossFunction$ is taking $verThreshold$ as input. The $verThreshold$ is a value less than 0.5, using which we want to restrict the $verificationPredictions$. The ReLU activation function penalizes the model when $verificationPredictions$ exceeded $verThreshold$. There is another input to the $LossFunction$ naming $alpha$, which weights the MSE and ReLU loss functions for loss calculation. The DNN model tuning is an optimization process that minimizes the calculated loss over the iteration by updating the model parameters. An optimization function is used for this purpose.

VI. Experiments and Results

This section provides the experimentation to show the analysis of the performance of DeepCAD on a state-of-the-art healthcare dataset for diabetes prediction. We consider an SHS that generates the necessary insulin delivery control signal to actuate an automated insulin pump. The expectation from the DeepCAD DNN model is to identify the patient's status (diabetes o not), which will assist in deciding the need for insulin delivery. Moreover, if the patient sensor measurements are somehow manipulated through an adversarial attempt or fault, the proposed framework will also be able to detect that.

A. Environmental Setup and Dataset Description

We conducted the experimentation on Dell Precision 7920 Tower workstation with Intel Xeon Silver 4110 CPU @3.0GHz, 32 GB memory, 4 GB NVIDIA Quadro P1000 GPU. The DNN model is trained using the PyTorch library [46]. As discussed before, the MEC is found using Welzi's algorithm.

We experiment with two state-of-the-art datasets- the Pima Indians diabetes dataset and Parkinson dataset for analyzing the performance of the proposed DeepCAD framework. The Pima Indians dataset contains several features that are used for medical prediction (patients' age, insulin level, BMI, number of pregnancies, etc.) and a target variable (i.e., diabetes status). We encode the target variable into two variables as described in the Section V. The Pima Indians dataset contains a number of indistinguishable samples. Hence, some data samples have been removed. The dataset contains 9 features. However, we prepossess the dataset by reducing the number of features using principal component analysis. Figure 4 shows the feature space of the preprocessed dataset, where there are 444 diabetic patient samples and 145 non-diabetic samples. On the other hand, the Parkinson dataset is composed of a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD). The Parkinson dataset is preprocessed similar to the Pima Indians dataset processing and there are 48 parkinson affected patients' samples and 147 samples for non-parkinson samples after preprocessing. The preprocessed Parkinson dataset is visualized in Figure 5.

B. Performance Analysis of Classification Task

Here, we assess the performance of the DeepCAD DNN model for the classification task. The performance has been measured using 4 metrics - accuracy, precision, recall, and f1-score and compared with the regular DNN model [47]. The accuracy metric quantifies the correctly predicted samples (both positive and negative) over all samples. The quantification of model predicted positive class samples being an actual member of the positive class sample can be measured using precision

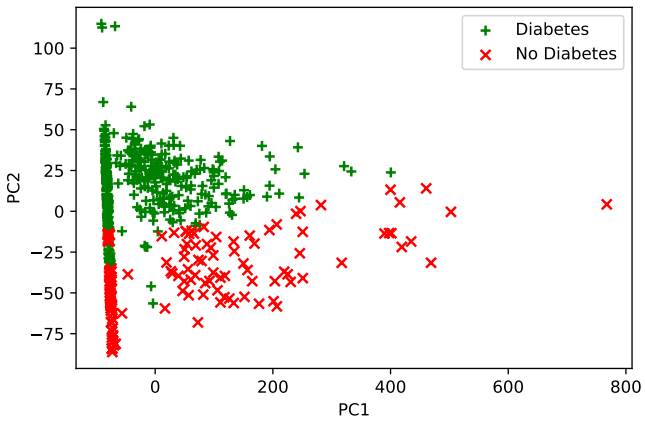

Fig. 4. Preprocessed Pima Indins dataset. The features with diabetes labels are shown as green '+' and the no diabetes labels are shown as red 'x'.

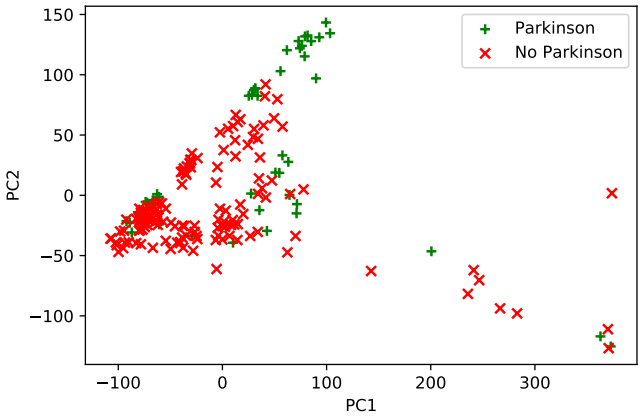

Fig. 5. Preprocessed Parkinson dataset. The features with parkinson labels are shown as green '+' and the no parkinson labels are shown as red 'x'.

metric. The recall metric determines the amount of predicted positive class samples overall actual positive class samples. The f1-score is the harmonic mean of precision and recall metric. Hence, the value of the f1-score is more aligned with the minimal in between precision and recall metric.

The regular DNN model is trained with the same features. However, unlike the DeepCAD DNN model, the regular DNN model has a single label, in which positive samples are encoded as 1, and negative samples are encoded as 0. Both models have been trained same similar hyper-parameters (i.e., 1000 epochs, 6 layers, 50 nodes in each layer). The loss function of the regular DNN model is calculated using the binary cross-entropy loss function. The loss function of the DeepCAD model is explained in Section V. Tables II and III show the performance comparison between DeepCAD and regular DNN models. From the metric, it is evident that both models demonstrate similar performance. The minor degrading performance of the DeepCAD DNN model can be reasoned due to the optimization of a more complex loss function compared to the regular DNN model.

TABLE II
PERFORMANCE COMPARISON IN BETWEEN REGULAR DNN AND DEEPCAD DNN (PIMA INDIANS DIABETES DATASET).

Performance Metric	Accuracy	Precision	Recall	F1-Score
Regular DNN	98.5	98.6	95.8	97.1
DeepCAD DNN	98.2	98.1	95.2	96.6

TABLE III
PERFORMANCE COMPARISON IN BETWEEN REGULAR DNN AND DEEPCAD DNN (PARKINSON DATASET).

Performance Metric	Accuracy	Precision	Recall	F1-Score
Regular DNN	95.5	94.9	93.2	94.0
DeepCAD DNN	95.1	93.8	92.4	93.1

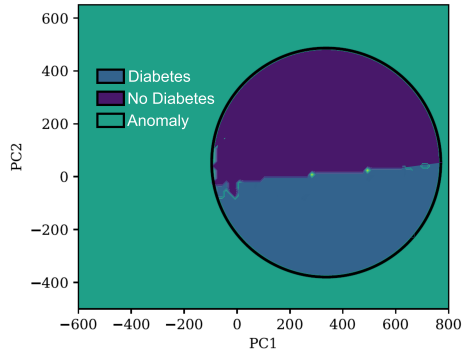

Fig. 6. Contour plot of Pima Indians Diabetes dataset.

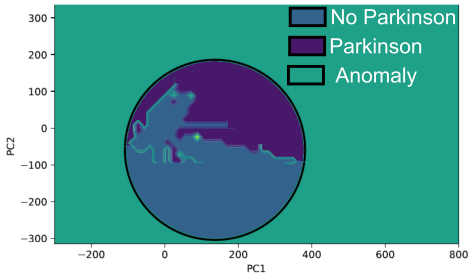

Fig. 7. Contour plot of Parkinson dataset.

C. Performance Analysis of Anomaly Detection

The DeepCAD DNN model can perform anomaly detection in addition to classification. Figure 6 shows the contour plot of the Pima Indians Diabetes dataset. It is evident from the figure that each and every point out of the MEC is classified as an anomaly. Moreover, the out-of-distribution points inside the MEC are also labeled as an anomaly. It can be argued that the data distributions enclosed by the circle are not perfectly enclosing the data distribution. In this work, we have experimented with circular boundaries only. However, the proposed concept can be used in the case of adding any boundary around the data distribution. The only difference will be, instead of scaling the sample prediction equally from the center to circle circumference, the scaling will be different based on distance from the boundary for any other regular or irregular shapes. Figure 7 shows the contour plot of the Parkinson dataset.

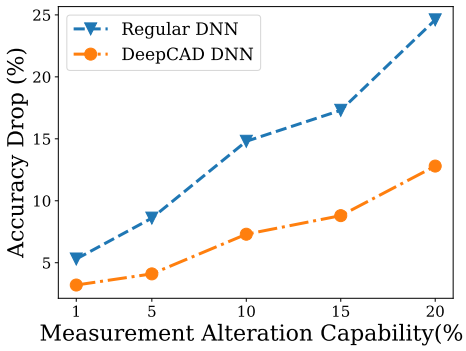

Fig. 8. Robustness comparison for the DeepCAD DNN model against FGSM attack (Pima Indians Diabetes Dataset).

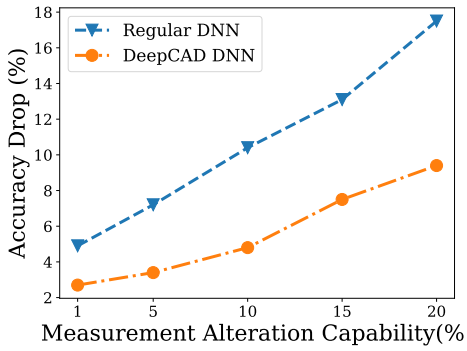

Fig. 9. Robustness comparison for the DeepCAD DNN model against FGSM attack (Parkinson Dataset).

D. Robustness analysis of the DeepCAD model

We evaluate the robustness of the DeepCAD DNN model by assessing the performance of the model against adversarial ML-based adversarial samples. We use accuracy drop (i.e., decrease in accuracy after adversarial ML attack) as the performance metric for robustness analysis. The adversarial samples are generated from regular DNN models with different attacker capabilities. Figure 8 shows the accuracy drop of the regular DNN and DeepCAD DNN models for Pima Indians Diabetes dataset. It can be clearly depicted that the accuracy drop of the DeepCAD DNN model is significantly less than the regular DNN model. The regular DNN model classifies samples into two classes - positive and negative, whereas the DeepCAD DNN model is also capable of classifying the samples into anomaly classes. The accuracy drop in the regular DNN model is due to the fact that with the given adversarial capability, some samples have been manipulated in a way that they are misclassified by the model. Almost half of the misclassified samples are labeled as an anomaly by the DeepCAD DNN model. However, the other half of adversarial samples have been misclassified by the DeepCAD DNN model because although those adversarial samples are being misclassified, they are still maintaining the data distribution. The more capability adversary possesses, the more adversarial samples will be generated to exploit the regular DNN model. However, most of those samples will be identified as an anomaly by the DeepCAD DNN model since, with drastic alteration, the adversarial samples will lose stealthiness by going outside the boundary. The similar kind of accuracy drop relation is observed in the case of Parkinson dataset as well as shown in Figure 9.

VII. Conclusion

A novel DNN model training framework, DeepCAD, is proposed in this work that can perform both classification and anomaly detection for SHS. Two different ML models were pipe-lined for these tasks in the traditional solutions. The real-time samples from the patients were first fed into the anomaly detection model. The patient status classification came into the picture only when anomaly detection models identified the samples as benign. However, the process initiates delay in the decision generation, which in turn can be responsible for the adverse physical condition of patients. Hence, we propose a stand-alone ML model for performing both tasks using a single model, thus reducing decision-making latency. The proposed framework is verified on a real dataset. The experimental analysis supports that the DeepCAD DNN model experiences around 50% less accuracy drop compared to a regular DNN model. The existing work is proof of concept. We want to address and solve; we consider a 2-dimensional feature space for this work since we consider the MEC with a circle using Welzi's algorithm. However, since Welzi's algorithm can draw a minimum enclosing n-sphere, the same algorithm should work for n-dimensional feature space. We will verify this in our future extension. Moreover, we plan to integrate the boundary of any shape in the extended version. The current version is tested for binary classification problems. However, the framework should be working for multi-class classification problems without alteration, which will be verified in the upcoming work. Moreover, we are planning to experiment and extend the framework with other ML algorithms.

References

[1] Haluk Demirkan. A smart healthcare systems framework. *It Professional*, 15(5):38–45, 2013.

[2] Mark É Czeisler, Kristy Marynak, Kristie EN Clarke, Zainab Salah, Iju Shakya, JoAnn M Thierry, Nida Ali, Hannah McMillan, Joshua F Wiley, Matthew D Weaver, et al. Delay or avoidance of medical care because of covid-19–related concernsâunited states, june 2020. *Morbidity and mortality weekly report*, 69(36):1250, 2020.

[3] Delay or avoidance of medical care because of covid-19ârelated concerns â united states, june 2020. https://www.cdc.gov/mmwr/volumes/69/wr/mm6936a4.htm, 2020. Accessed: 2021-04-21.

[4] Elizabeth Lawrence. Nearly half of americans delayed medical care due to pandemic. https://khn.org/news/nearly-half-of-americans-delayed-medical-care-due-to-pandemic/, 2020. Accessed: 2021-04-21.

[5] Conor Stewart. Total amount of global healthcare data generated in 2013 and a projection for 2020. https://www.statista.com/statistics/1037970/global-healthcare-data-volume/, 2021. Accessed: 2022-02-22.

[6] Trong Thanh Han, Huong Yen Pham, Dang Son Lam Nguyen, Yuki Iwata, Trong Tuan Do, Koichiro Ishibashi, and Guanghao Sun. Machine learning based classification model for screening of infected patients using vital signs. *Informatics in Medicine Unlocked*, 24:100592, 2021.

[7] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[8] Ivan Y Tyukin, Desmond J Higham, and Alexander N Gorban. On adversarial examples and stealth attacks in artificial intelligence systems. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2020.

[9] Matthew Watson and Noura Al Moubayed. Attack-agnostic adversarial detection on medical data using explainable machine learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8180–8187. IEEE, 2021.

[10] Understanding the iot digital attack surface and threat mitigation. https://www.tokenex.com/blog/understanding-the-iot-digital-attack-surface-and-threat-mitigation, 2018. Accessed: 2022-02-22.

[11] Hp study reveals 70 percent of internet of things devices vulnerable to attack. https://www.hp.com/us-en/hp-news/press-release.html?id=1744676.YhUL0pZOm5c, 2014. Accessed: 2022-02-22.

[12] T. Wehbe, V. Mooney, A. Javaid, and O. Inan. A novel physiological features-assisted architecture for rapidly distinguishing health problems from hardware trojan attacks and errors in medical devices. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 106–109, 2017.

[13] Vahab Pournaghshband, Majid Sarrafzadeh, and Peter Reiher. Securing legacy mobile medical devices. In *International Conference on Wireless Mobile Communication and Healthcare*, pages 163–172. Springer, 2012.

[14] Darlene Storm. Medjack: Hackers hijacking medical devices to create backdoors in hospital networks. https://www.computerworld.com/article/2932371/medjack-hackers-hijacking-medical-devices-to-create-backdoors-in-hospital-networks.html, 2015. Accessed: 2020-01-08.

[15] Ahmad Almogren, Irfan Mohiuddin, Ikram Ud Din, Hisham Al Majed, and Nadra Guizani. Ftm-iomt: Fuzzy-based trust management for preventing sybil attacks in internet of medical things. *IEEE Internet of Things Journal*, 2020.

[16] Rashmi V Deshmukh and Kailas K Devadkar. Understanding ddos attack & its effect in cloud environment. *Procedia Computer Science*, 49:202–210, 2015.

[17] Ponemon Institute. *2019 Global State of Cybersecurity in Small and Medium-Sized Businesses*, October 2019. https://www.keeper.io/hubfs/PDF/2019

[18] Laira Dyrda. Inside uvm medical center's ransomware attack: 11 details. https://www.beckershospitalreview.com/cybersecurity/inside-uvm-medical-center-s-ransomware-attack-11-details.html, 2020. Accessed: 2021-07-05.

[19] 5 common cybersecurity vulnerabilities in the iomt. https://securityscorecard.com/blog/common-cybersecurity-vulnerabilities-in-the-iomt, 2021. Accessed: 2022-02-22.

[20] AKM Iqtidar Newaz, Nur Imtiazul Haque, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A Selcuk Uluagac. Adversarial attacks to machine learning-based smart healthcare systems. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.

[21] Samuel G Finlayson, John D Bowers, Joichi Ito, Jonathan L Zittrain, Andrew L Beam, and Isaac S Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019.

[22] Mehran Mozaffari-Kermani, Susmita Sur-Kolay, Anand Raghunathan, and Niraj K Jha. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE journal of biomedical and health informatics*, 19(6):1893–1905, 2014.

[23] Samuel G Finlayson, Hyung Won Chung, Isaac S Kohane, and Andrew L Beam. Adversarial attacks against medical deep learning systems. *arXiv preprint arXiv:1804.05296*, 2018.

[24] Abdur Rahman, M Shamim Hossain, Nabil A Alrajeh, and Fawaz Alsolami. Adversarial examplesâsecurity threats to covid-19 deep learning systems in medical iot devices. *IEEE Internet of Things Journal*, 8(12):9603–9610, 2020.

[25] Nur Imtiazul Haque, Mohammad Ashiqur Rahman, and Hossain Shahriar. Ensemble-based efficient anomaly detection for smart building control systems. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 504–513. IEEE, 2021.

[26] Nur Imtiazul Haque, Alvi Ataur Khalil, Mohammad Ashiqur Rahman, M Hadi Amini, and Sheikh Iqbal Ahamed. Biocad: Bio-inspired optimization for classification and anomaly detection in digital healthcare systems. In *2021 IEEE International Conference on Digital Health (ICDH)*, pages 48–58. IEEE, 2021.

[27] Juan Zhao, Sachin Shetty, Jan Wei Pan, Charles Kamhoua, and Kevin Kwiat. Transfer learning for detecting unknown network attacks. *EURASIP Journal on Information Security*, 2019(1):1–13, 2019.

[28] Jack W Smith, James E Everhart, WC Dickson, William C Knowler, and Robert Scott Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care*, page 261. American Medical Informatics Association, 1988.

[29] SP Rajamhoana, C Akalya Devi, K Umamaheswari, R Kiruba, K Karunya, and R Deepika. Analysis of neural networks based heart disease prediction system. In *2018 11th international conference on human system interaction (HSI)*, pages 233–239. IEEE, 2018.

[30] Subhashree Mohapatra, Janmenjoy Nayak, Manohar Mishra, Girish Kumar Pati, Bignaraj Naik, and Tripti Swarnkar. Wavelet transform and deep convolutional neural network-based smart healthcare system for gastrointestinal disease detection. *Interdisciplinary Sciences: Computational Life Sciences*, 13(2):212–228, 2021.

[31] Mehedi Masud, Amr E Eldin Rashed, and M Shamim Hossain. Convolutional neural network-based models for diagnosis of breast cancer. *Neural Computing and Applications*, pages 1–12, 2020.

[32] Ibrahim Alrashdi, Ali Alqazzaz, Raed Alharthi, Esam Aloufi, Mohamed A Zohdy, and Hua Ming. Fbad: Fog-based attack detection for iot healthcare in smart cities. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0515–0522. IEEE, 2019.

[33] Leandros A Maglaras, Jianmin Jiang, and Tiago Cruz. Integrated ocsvm mechanism for intrusion detection in scada systems. *Electronics Letters*, 50(25):1935–1936, 2014.

[34] AKM Iqtidar Newaz, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A Selcuk Uluagac. Healthguard: A machine learning-based security framework for smart healthcare systems. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 389–396. IEEE, 2019.

[35] Amaal Al Shorman, Hossam Faris, and Ibrahim Aljarah. Unsupervised intelligent system based on one class support vector machine and grey wolf optimization for iot botnet detection. *Journal of Ambient Intelligence and Humanized Computing*, 11(7):2809–2825, 2020.

[36] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.

[37] Juan Zhao, Sachin Shetty, and Jan Wei Pan. Feature-based transfer learning for network security. In *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, pages 17–22. IEEE, 2017.

[38] Sungyong Seo, Sercan Arik, Jinsung Yoon, Xiang Zhang, Kihyuk Sohn, and Tomas Pfister. Controlling neural networks with rule representations. *Advances in Neural Information Processing Systems*, 34, 2021.

[39] Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049, 2010.

[40] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

[41] Kun Wang, Xiaoying Bai, Jing Li, and Cong Ding. A service-based framework for pharmacogenomics data integration. *Enterprise Information Systems*, 4(3):225–245, 2010.

[42] Rumen Kyusakov, Jens Eliasson, Jerker Delsing, Jan Van Deventer, and Jonas Gustafsson. Integration of wireless sensor and actuator nodes with it infrastructure using service-oriented architecture. *IEEE Transactions on industrial informatics*, 9(1):43–51, 2012.

[43] Hagan Demuth Beale, Howard B Demuth, and MT Hagan. Neural network design. *Pws, Boston*, 1996.

[44] Zijun Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2. IEEE, 2018.

[45] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[46] Automatic Differentiation In PyTorch. Pytorch. 2018.

[47] Kendrick Boyd, Kevin H Eng, and C David Page. Area under the precision-recall curve: point estimates and confidence intervals. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 451–466. Springer, 2013.