IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 3, 2020, accepted December 22, 2020, date of publication January 13, 2021, date of current version January 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051530

Privacy-Preserving Distributed IDS Using Incremental Learning for IoT Health Systems

ALIYA TABASSUM[1], (Student Member, IEEE), AIMAN ERBAD[2], (Senior Member, IEEE), AMR MOHAMED[1], (Senior Member, IEEE), AND MOHSEN GUIZANI[1], (Fellow, IEEE)

[1]Department of Computer Science and Engineering, College of Engineering, Qatar University, Doha, Qatar
[2]Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Qatar Foundation, Doha, Qatar

Corresponding author: Aiman Erbad (aerbad@hbku.edu.qa)

This work was supported by Qatar University - Grant IRCC [2020-003]. The findings achieved herein are solely the responsibility of the authors.

ABSTRACT Existing techniques for incremental learning are computationally expensive and produce duplicate features leading to higher false positive and true negative rates. We propose a novel privacy-preserving intrusion detection pipeline for distributed incremental learning. Our pre-processing technique eliminates redundancies and selects unique features by following innovative extraction techniques. We use autoencoders with non-negativity constraints, which help us extract less redundant features. More importantly, the distributed intrusion detection model reduces the burden on the edge classifier and distributes the load among IoT and edge devices. Theoretical analysis and numerical experiments have shown lower space and time costs than state of the art techniques, with comparable classification accuracy. Extensive experiments with standard data sets and real-time streaming IoT traffic give encouraging results.

INDEX TERMS Deep learning, internet of things (IoT), intrusion detection system (IDS), incremental learning, pre-processing.

I. INTRODUCTION

The Internet has become an indispensable part of our lives at work and in our home applications, such as personal health assistants, remote health monitoring, and smart home security. The remote monitoring applications especially health applications generate a large volume of continuous data from vital signs and other signals (e.g., EEG, ECG). This health data is confidential so we want to ensure that the data privacy is not violated. Any tampering of the data by an attacker can cause a serious issue, such as false diagnosis, delay in an emergency, and other health complications, which can lead to death [1]. In this new era, a shift to digital transactions in different applications mandates a shift in our mindset to secure all these transactions as they traverse small IoT sensors or home appliances. Security of the Internet of Things (IoT), especially the security of IoT networks and end-devices, is essential for our safety and security. Despite various security practices such as authentication, encryption, and access controls, security practitioners recommend active network monitoring and adaptable Intrusion Detection System (IDS) [2], [3]. Various Intrusion detection systems are proposed to detect malicious activities on IoT devices using Machine Learning (ML) and Deep Learning (DL) techniques [4]. Usually, an IDS system involves 2 major steps: pre-processing and classification [5]. Pre-processing tasks using ML techniques, extract or select features from the network traffic and sends it to the classifier. The performance of ML classifiers depends on the features selected. Traditional ML algorithms can either select the features manually or using a predefined feature selection algorithm. Sometimes, the feature selection can cause the classification models to underfit or overfit, affecting the accuracy when dealing with massive data. Classical ML models suffer from a lack of scalability and low detection rate in massively distributed IoT nodes. On the other hand, Deep Learning (DL) models can choose features automatically and provide adequate representation of features [4]. Among these, Convolutional Neural Network (CNN) is a deep learning technique proven to give higher detection accuracy and better feature analysis with massive traffic [6].

Moreover, diverse and novel attacks emerge every day, hence, the IDS must identify those correctly. Most of the new attacks are variants of existing and the same attacks are

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Xiao.

launched using different techniques. When the IDS is unable to categorize unknown network intrusion activity properly, it gives higher False Positive (FP) and True Negative (TN) rates. Researchers have proposed IDS based on incremental feature learning to retrain the model to cope up with the emerging attacks [7]. However, most of the incremental learning techniques assume that the data arriving is labeled, which is not practical and makes the IDS incapable of dealing with new attacks [8], [9]. Some of the incremental semi-supervised algorithms were proposed to deal with unlabelled incoming data [10]. However, these algorithms do not achieve appropriate performance during the testing phase because of the redundant pre-processing, high computational demands and high memory consumption, as the huge amount of data is transferred from IoT devices to cloud/edge platforms (where the IDS is placed) to provide various services. The streamed IoT data is massive and heterogeneous with a mix of data with known and unknown labels arriving one by one or chunk by chunk, which may overwhelm the classifier and lead to slow classification [11]. Therefore, the accuracy of incremental deep learning models can decrease. The response time (latency) between the IoT and the deep learning models has to be synchronized to handle the vast amount of real-time traffic. The operational stability of the IoT IDS system might also get compromised when there is a delay between the IoT domain and deep learning models detecting intrusions [12].

To sum up, the accuracy reduction is not the only problem with the existing incremental IDS algorithms. The processing cost, memory consumption, and decision time are high when deployed in a real-time IoT network, where the traffic patterns are dynamic. Sometimes, the classifier is jammed or gives incorrect results. Only a classifier with less response time and higher accuracy can ensure the security of the network. To meet the requirements of quick and correct response from the deep learning model, there has to be a shift from the traditional way of data transfer from the IoT devices. In this paper, we aim to distribute the model with parallel pre-processing on IoT devices to reduce the overhead on the deep learning classifier model. To speed up the classification and minimize the input of the classification model, we pre-process the data on the IoT devices and on the bridge IoT network so that less data is sent to the remote classifier. When the raw data is sent directly to the classifier, various privacy concerns and confidentiality issues arise [13]. Data is vulnerable to be intercepted or reverse engineered by some adversaries, which may lead to privacy and security threats. In our model, the pre-processed data is sent to the classifier, which is different from the raw input and guarantees data privacy. Also, the risk of reverse-engineering the data decreases because the data processing is distributed over multiple distributed networks (i.e., IoT devices, IoT bridge network, and the remote classifier network).

The contributions of this article are as follows:

1) **A distributed IDS model** to reduce the overhead on the centralized edge classifier is proposed such that the subsequent latency between pre-processing and decision making phase is minimum.
2) **Parallel pre-processing on IoT end-devices** that utilizes computational power and memory resources of the IoT devices to ensure the security and privacy of the raw data. Sharing the pre-processing task among IoT devices reduces computational resources and delays at the source, and gives comparable performance to the centralized classification task.
3) **An incremental learning model** is proposed to update the classifier seamlessly with emerging features in order to detect new attacks. Adding new features minimizes the objective function residuals and merging similar features avoids overfitting and gives a compact representation of the features. Our incremental model gives better accuracy than other approaches.
4) **A Comprehensive evaluation** of our distributed intrusion detection model using different experiments with standard datasets and real-time IoT traffic. We also perform decision time analysis for the classification process and compare it to other centralized models.

The rest of the paper is organized as follows. Section II discusses the related work. Section III presents the novel distributed intrusion detection model based on parallel pre-processing on IoT devices and incremental learning. It includes a detailed presentation of the data pre-processing, feature comparison and selection, and the CNN classification algorithm. Section IV discusses the technical implementation details, while the evaluation metrics and experimental results are presented in Section V. Finally, we draw conclusions and future research directions in Section VI.

II. RELATED WORK

This section reviews related work on incremental learning algorithms. We discuss problems of intrusion detection with the streaming IoT data, the threat model of smart home IoT health system, and other relevant information.

A. SMART HOME HEALTH IoT SYSTEM SCENARIO

We study a smart home consisting of various IoT devices, such as medical sensors, home appliances, and utilities as shown in Figure 1. To be more concrete, we consider the new medical applications that are becoming significant in our homes, such as early diagnosis and real-time patient monitoring. Real-time monitoring using vital signs and automated emergency response reduces the dependency of the patient-caregiver and decreases the healthcare costs. Different wearable IoT sensors interact with each other for communication using different protocols. The amount of traffic is large and the data is flowing continuously. The nature of the data is confidential and needs to be secured. This diverse environment provides a potential attacking surface and the attacker may manipulate the traffic patterns. If the manipulated patterns are not detected at the early stages, it may lead to the unavailability of the system. If an adversary compromises the smart home IoT health system network by

attacking any one of the wearable IoT sensors, for instance, the vital signs can be changed threatening the life of a patient. An attacker may gain control of the whole network by compromising devices through performing different types of attacks. Our task is to identify the compromised device using IDS before it gets multiple devices compromised in the network.

B. INTRUSION DETECTION IN STREAMING IoT DATA

Classification of streaming data is difficult due to the dynamic nature of the incoming traffic. This is a major problem in data mining and machine / deep learning applications. The learning algorithms are trained at one time but need to be re-trained due to the continuously evolving nature of data. Re-training involves huge time overhead, computational resource requirement, and memory footprint due to the large scale IoT datasets arriving in real-time from the environment. While many learning algorithms can work with the raw input features but their behavior degrades as the number of samples grows. The performance degradation and the inability to classify is due to the high number of features that has to be processed for decision making, which is referred to as the ''curse of dimensionality'' [14].

C. DIMENSIONALITY REDUCTION USING AUTOENCODERS

Feature selection is one of the approaches to reduce input features/ values given to the classifier. Usually, feature selection techniques analyze each variable independently, which is an NP-hard combinatorial problem. Sometimes, the variables that do not provide any information individually, give useful information when put together with other variables. Therefore, feature extraction methods are proposed to construct meaningful features or to extract high-level information from the raw features [15]. Data compression is the reduction in the number of bits to represent the data. Auto Encoders (AE) are unsupervised neural networks that perform data compression. AE is a simple learning neural network that reconstructs input into output with the least possible distortion, i.e., the output to be as close to the input. It applies backpropagation, similar to Artificial Neural Network (ANN), to set the target values to be equal to the inputs. AEs can be used for supervised or unsupervised learning, it means it can handle data that is unlabelled but labeled data has more information that can be used.

AEs are efficient in reducing noise, dimensionality reduction, and learning important features in the data while reconstructing the input (Pre-training). The learned features reveal the non-linear properties of the data. Extracted features provide a good discriminative ability for the classification task [16]. The input features are crucial parameters for efficient intrusion detection. Different features change the detection performance of the IDS model. There are standard algorithms for dimensionality reduction, such as Principal Component Analysis (PCA), but AE gives higher efficiency due to the deep extraction of non-linear properties of the features. AE gives a representation of the output at each layer, giving multiple transformations at different dimensions of the input parameters. Data projections and visualizations using AE are more accurate than PCA and other dimensionality reduction techniques. In addition, outliers' detection is a by-product of any AutoEncoder technique. PCA, LDA, and other standard ML feature extraction/ data reduction methods are used as they are easier to implement. However, these methods are less capable in modeling nonlinear structures of data, compared to the deep learning methods, especially in large datasets. The compressed representation of data with deeper AE networks can improve model performance [17].

One of the hybrid feature-extraction methods is proposed by merging Sparse AutoEncoders (SAE) and Principal Component Analysis (PCA) to extract low-level features and is applied to various classifiers [18]. The results prove that the large numbers of nodes in hidden layers and deep information extraction from features are two critical parameters for achieving high performance [19]. Traditional auto encoders fail to inspect relationships of data samples. They generate new features by minimizing only the reconstruction loss of the data. To resolve this problem, we minimize the reconstruction loss for data and relationships between data features.

We have investigated various autoencoder networks that may fit our problem. Among all, we chose Sparse AutoEncoders, as it allows us to activate a selected number of nodes, which is the first requirement for this research challenge. We merge correlated features into one, where we have to activate and de-activate nodes and merge them by clustering into one. Besides, SAE consists of a single hidden layer which can be deployed on any of the IoT device irrespective of its resource constraints. The loss function of the network is constructed by penalizing the activations within the layer [20]. SAE is efficient in encoding the input data by approximating minimum error, which guarantees less loss of information, and extraction of the best feature representation. From literature [21], we have known that, even with the simplest algorithm, it is possible to extract useful features and achieve higher performance by focusing on the hyper-parameter choices rather than on the complexity of the algorithm. We have incorporated these choices diligently after experimenting with various values on SAE.

D. INCREMENTAL LEARNING AND DISTRIBUTED DEEP LEARNING MODELS

Recently researchers have shown much interest in incremental learning algorithms [22], also referred to as Transfer Learning [23], Online Learning [24], and Federated Learning [25] to adapt to the complex real environments. Existing supervised incremental learning algorithms are Learn++ [26], incremental SVM [9], and incremental Support Vector Machines (SVMs) that was proposed to incrementally learn from a reduced number of support vectors [27]. Incremental learning based on user-provided labels or pairwise constraints is not suitable for dynamic data. Using streaming data, it is not always possible to label all the arriving data. If the streaming data is available at time-stamp t, the labeling of those samples

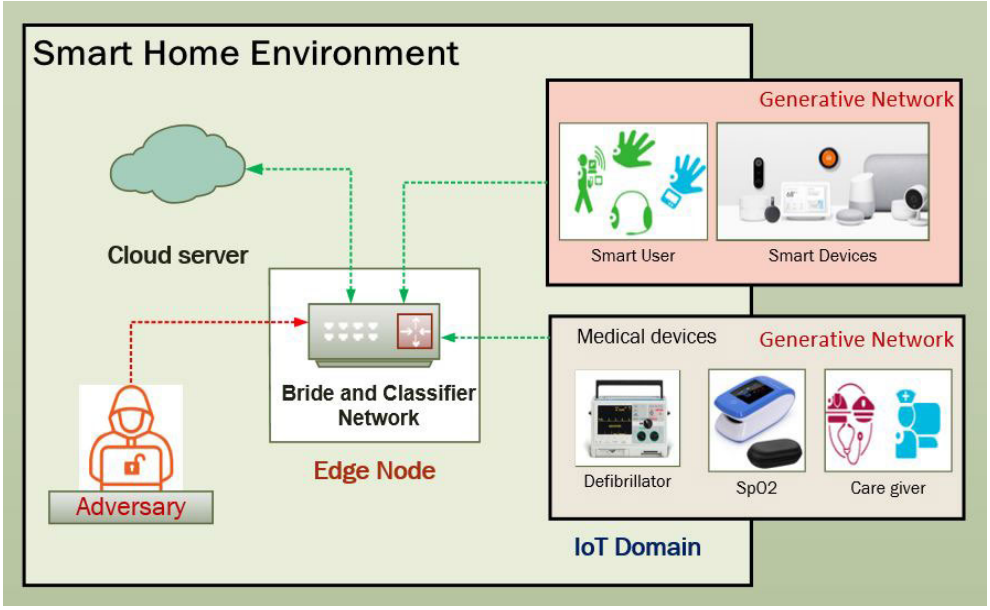

FIGURE 1. System scenario: A Smart Home Environment.

either manually or automatically by any method may occur at time-stamp $t + 1$, which is challenging and time-consuming. Some researchers proposed semi-supervised algorithms to resolve this issue by using Bayesian learning, Subspace learning [28], Clustering [29], and Classification techniques [30]. Fitting unlabelled data based on an inappropriate model misleads the learning process and degrades the performance.

A distributed self-governing model is proposed where the decision is taken locally on the data collection devices (i.e, the positioning of the hierarchical processing layers on the IoT devices). Distributed processing mitigates the latency between the data transfer and decision making [31] as well as helps the resource-limited devices to meet the demands of the deep learning solutions for video/ image classification. This model is unsuitable to be applied for intrusion detection as the raw data is shared among the IoT devices for decision making and it does not guarantee the privacy and security requirements. In literature, we don't have any incremental algorithm that is developed in a distributed fashion.

E. NOTATIONS

In this paper, we denote F_i for input traffic (feature set) from an IoT device, where each value is denoted as x_i which is unprocessed. After pre-processing the values are denoted as $\hat{x}_i$ and the whole set of pre-processed values from an IoT device is denoted by G_i. The total number of IoT devices in the smart home health system is denoted by p, the number of input traffic (unprocessed) is denoted by n and after processing the number is denoted by m. The final optimal set of features used for training is denoted by H and after every update, we denote it by H'. Based on these notations, C_i defines the nodes of the generative network with hidden layers as hl_i, with weight matrix and biases denoted by W and b respectively.

III. METHODOLOGY

The traditional feature selection and classification algorithms do not perform well on massive real-time data as it is heterogeneous with a huge noise and other irrelevant information. One of the major reasons where researchers fail to train models on real-time traffic is the unavailability of real-time traffic data sets. Organizations do not share real data for training purposes due to privacy and security concerns. Besides, using real-time data from different sources for training is vulnerable to manipulation attacks where the attacker injects malicious traffic patterns to corrupt the training model. In this section, we describe a privacy-preserving distributed IDS model based on incremental learning, that identifies a Denial of Service (DoS) attacks.

A. DISTRIBUTED ARCHITECTURE

Our distributed architecture is inspired by a method based on Incremental semi-supervised learning on streaming data for video classification which consists of 3 layers [32]. The first layer learns features from the incoming streaming data, the third layer regularises the network by building similarity constraints. These two layers are connected by a bridge layer. Likewise, our model is segregated into 3 deep networks, namely, Generative network, Bridge network and a Classifier network. The tasks that are implemented in these networks are: pre-processing, comparison and classification, respectively. The detection process is distributed over these networks right from pre-processing of the incoming streaming data to the classification results.

The proposed IDS system identifies DoS attacks in IoT home network scenarios. The reason for selecting a DoS attack, is because it generates huge traffic which may shutdown or overwhelm the classifier before it could categorize

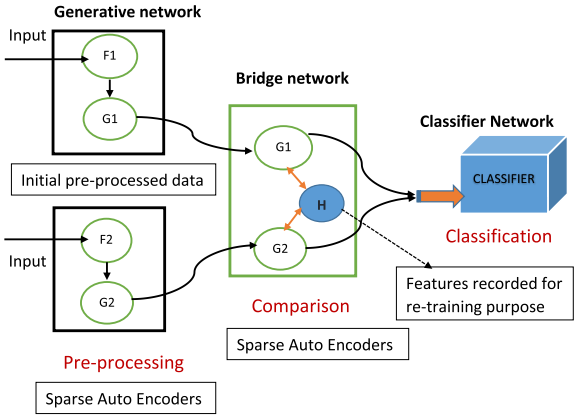

FIGURE 2. System model.

the attack and notify the attack detection. It is considered to be the most vulnerable attack in IoT devices due to their resource constraints, and especially for IoT health devices, it can be devastating if the device is shutdown. So, we have targeted DoS and its categories. As our model is incremental, it can be re-trained for many other attack identification tasks. In the following subsections, we provide a detailed explanation of the tasks of each network.

1) GENERATIVE NETWORK

In this network, massive traffic (either normal or malicious) events are captured from different IoT devices. We build a layer of sparse autoencoders (AEs) on each IoT device to analyze traffic within the device. The collected raw data from each IoT device is recorded and pre-processed for unique feature identification. All AEs pre-process independently and unique features are extracted from the incoming traffic from each of the IoT devices simultaneously. Here, the unique features mean the features without redundancies and similarities. Similar features are merged based on the heuristics proposed. Once unique features are extracted on each IoT device, the useful features are sent to the Bridge network for further analysis and recording of incremental features. The Bridge network is separated from the classifier network, but both are deployed on one Edge device.

The networks and their tasks are illustrated in Figure 2, which shows 3 networks, Generative, Bridge, and Classifier. The Generative network takes input from the IoT devices, which is incoming traffic. Let us assume the incoming traffic on each IoT device as $x_1, x_2, x_3, \ldots\ldots\ldots x_n$, we refer it as input set of values F. The input set of values of each of the IoT device is referred to as $F_1, F_2, F_3, \ldots\ldots F_p$, where '$p$' is the number of IoT devices in that smart home health system. Each input set of values are pre-processed to extract unique features. The input set of values $F_1 = x_1, x_2, x_3, \ldots\ldots\ldots x_n$ on IoT device 1 are converted to $G_1 = \hat{x_1}, \hat{x_2}, \hat{x_3}, \ldots\ldots\ldots \hat{x_m}$, which are the unique extracted features. Likewise, $G_1, G_2, \ldots\ldots G_p$ are all sent to the Bridge network.

We propose a novel algorithm for feature extraction from the streaming IoT data and automatically compare features for duplicity and fuse it into informative deep features which is elaborated in the next subsection in Algorithm 1.

2) BRIDGE NETWORK

The bridge network receives unique feature sets ($G_1, G_2, \ldots G_p$) from all IoT devices. It forwards those values to the classifier network for results. Apart from this, it analyzes the received feature sets by comparing them to the existing features on which our classifier model is trained. After which, it records any new feature and forms a new filter in the network. The working of this network is explained in Algorithm 2 in the next section. Bridge network construction is similar to the AE layers on the IoT devices. This network can be either placed on a sub-edge node such as IoT Hub or the same node as the classifier. In our case, we have placed it on the same device as that of the Classifier. From Figure 2, $G_1, G_2, \ldots\ldots G_p$ are sent to the classifier for results, where the classification results are given as Anomaly or Normal. Besides, $G_1, G_2, \ldots\ldots G_p$ are processed through the filters of the Bridge network to know if any new features are recorded from the incoming traffic and is named as a new feature set 'H'. This is used for re-training purposes.

3) CLASSIFIER NETWORK

Inspecting every packet is time-consuming and computationally expensive in streaming data to identify intrusions. We pre-process packets diligently to send only useful data to the classifier. We used a deep learning CNN model to detect the intrusions over the network, which has higher accuracy and bigger data handling capability [6]. CNN helps to reduce false alarms and unnecessary service visits. It has multiple levels of abstraction that discriminate features easily without overfitting the model. In this network, classification is executed with faster response time to alleviate the latency between IoT devices and IDS decision making. The classifier network gives results after processing the feature sets from the Bridge network. The activations are connected to small regions of the neurons and not in a fully connected manner.

IV. DETAILED DESIGN

The intrusion classification process is carried out in 3 phases in the distributed deep networks as shown in Figure 3 and as explained below.

A. GENERATIVE NETWORK: PRE-PROCESSING PHASE

We train a deep sparse AutoEncoder network on IoT devices for feature extraction using ReLU activation function. We used AutoEncoders as it detects rare events (outliers) and extracts high-level features which are helpful to identify newer attacks. We perform the following steps in the preprocessing phase. Initially, the data is adjusted in one format and normalized so that we get improved results with our classifier. This is applied in both cases, Training and Testing.

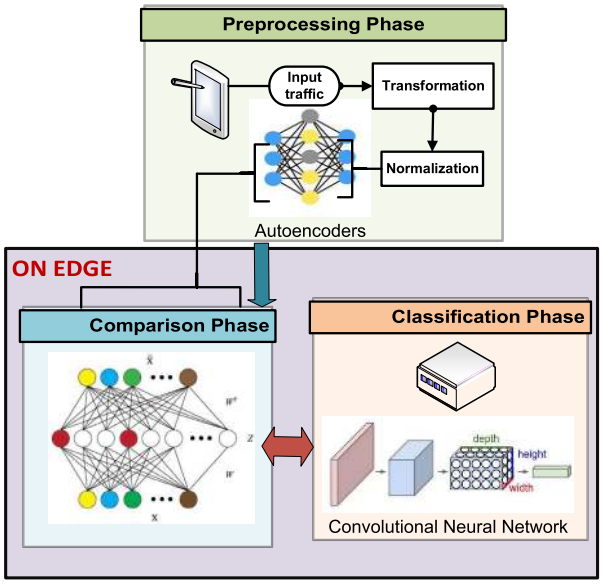

FIGURE 3. Process flow.

1) FEATURE TRANSFORMATION AND NORMALIZATION

The incoming traffic is transformed into one format such as the text values are converted into numbers. Then, the transformed features are normalized using the Z-score function.

$$Z(i) = \frac{(v(i) - \mu)}{\sigma} \tag{1}$$

where μ is the mean of the n values for a given feature $(v(i)\,(i \in 1, 2, 3, \ldots n))$ and σ is the standard deviation.

2) UNIQUE FEATURE EXTRACTION AND FUSION

Extracting the right features is often complex and difficult task which can be solved by using deep neural networks [33]. In this process, we use some heuristics that cluster and merge features by agglomerative clustering and reduction of hidden layer size. The most popular type of clustering techniques are agglomerative hierarchical and K-means [34], among which hierarchical clustering is better except in terms of time complexity. In our technique, we aim to utilize the benefits of agglomerative clustering and also reduce the time it takes for this process. The two major tasks: feature extraction and merging are explained below:

- To extract a high-level unique feature set, we try to reduce the number of filters in the sparse autoencoder network while preserving the sparsity. To eliminate redundant features from the network, we add a non-negative weights constraint to the network. The separation capability and sparsity of hidden layers is increased by utilizing RELU activation function and a penalty factor. Traditional Autoencoders fail to consider the relationships between data samples. In our network, we consider reconstruction loss of relationships by evaluating correlation and similarity between data features and by filtering weak and trivial relationships, which helps in identifying similar features for the merging process.
- For feature fusion, the filters/ nodes of the Sparse AE network that are identical/ similar are merged using Agglomerative clustering. Average similarity threshold is calculated before the merging decision using minimal distance between two data features, and those are merged based on Agglomerative clustering. This deep neural network approximates the input vector with the minimum possible error. The minimal distance between two features is calculated using Equation (2), where W is the weight of that feature.

$$\hat{M} = argmin\{x_i, x_{i+1}\} d(W_{x_i}, W_{x_{i+1}}) \tag{2}$$

The following algorithm illustrates the process of feature extraction and fusion.

Algorithm 1: Feature Extraction and Fusion

Result: Unique useful feature set = $\hat{x_1}, \hat{x_2}, \hat{x_3}, \ldots\ldots\ldots \hat{x_m}$

Input: $x_1, x_2, x_3, x_4, \ldots\ldots\ldots\ldots, x_n$;
while corr $(x_i, x_{i+1}) \leq 90$ **do**
  initialize W and b;
  initialize minRL;
  while $RL \leq minRL$ **do**
    $W, b = Train\_AE$ (*dataset, W, b, iterations*);
    $RL =$ *reconstruction loss* (*dataset, W, b*);
    $L1 + KL$;
    $minRL = RL$;
    Control penalty term;
    Add non-negative weight constraint;
    Initialize threshold T ;
    $T = argmin\{x_i, x_{i+1}\}\, d\,(W_{x_i}, W_{x_{i+1}})$
    if *function*$(x_i, x_{i+1}) \leq T$ **then**
      Add $\hat{x_i}$ & $\hat{x_{i+1}}$ *to feature set*;
    else
      Merge (x_i, x_{i+1}) *as* $\hat{x_i}$;
    end if
  end while
end while

We begin with the original filters (nodes of the SAE network). Let us assume it as C_i. Likewise, we have n number of nodes $C_1, C_2, C_3, \ldots\ldots C_n$. Each node on the SAE collects some incoming traffic, which we assume as $x_1, x_2, x_3, \ldots\ldots x_n$. At the beginning step, we check for its correlation and then initialize the weight and biases of the network (W,b). We model the SAE network in such a way that it extracts high-level features with the least loss of information by minimizing the reconstruction error. We calculate the reconstruction loss every time by back-propagation until we reach the least value. Next, we add non-negative weight constraint to the network to eliminate redundant features, which also guarantees: less reconstruction error, extraction of distinct features and an increased sparsity. Then, we apply

agglomerative clustering on the two most similar nodes C_i and C_j until the similarity is greater than the chosen threshold of distancing measure. We regularize the network to avoid over-fitting the data. The nodes C_i to C_n takes input values from x_i to x_n on each of the IoT device and applies Algorithm 1 to convert those values into unique values $\hat{x_i}$ to $\hat{x_m}$. Each hidden layer computation is defined as $[hl_i(x) = f(w_i^T.X + b_i)]$. The RELU activation function that is designed to obtain a better representation of the input features is represented as: $[hl_i(X, W, b) = RELU(w_i^T.X + b_i)]$. The following 3 components control the network which are the 3 major components in the objective function of the sparse autoencoder:

1) Reconstruction error over the entire dataset (that has to be minimized). The reconstruction error is formulated as below for Sparse Autoencoders.

$$L(W, b) = \frac{1}{m}\sum_{k=1}^{m} || \sigma(W^{(2)}\sigma(W^{(1)}x^{(k)} + b_x) + b_h) - x^{(k)} ||_2^2 \tag{3}$$

2) Regularizer to prevent overfitting by reducing the magnitude of the weights (make sure that they are close to zero and non-negative). We incorporate L1 Regularization in our loss function, which is shown as follows:

$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i | a_i^{(h)} | \tag{4}$$

3) Kullback Leibler Divergence: It defines relative entropy, which is a standard measure of the difference between the two distributions used to regularize the encoder. We add this as an extra penalty term to increase sparsity between the nodes.

$$\mathcal{L}(x, \hat{x}) + \sum_j KL(\rho \mid \hat{\rho}_j) \tag{5}$$

ρ is a sparsity parameter that denotes the average activation of a neuron over a collection of samples, which is calculated as:

$$\hat{\rho}_j = \frac{1}{m}\sum_i [a_i^{(h)}(x)] \tag{6}$$

The sparsity parameter is used to make some of the hidden nodes inactive in turn increasing the sparsity between the nodes of the hidden layer.

B. BRIDGE NETWORK: COMPARISON PHASE

The bridge network is made up of filters on which our model is trained using sparse autoencoders. The bridge network takes input from the generative network (from all IoT devices) and transfers the same input to the classifier network. At the same time, it compares that input with the filters of the autoencoder network that we formed initially. If ever the incoming features from the Generative network do not match any of these filters then a new filter is added to this network. It means a new feature has been added to our feature set on which we can categorize attacks. After which, we use this new set of features for re-training. The output of generative network sparse autoencoders is fed as input to the autoencoder of the Bridge network. Assume that the model is trained initially on a set of features $x_1, x_2, x_3, \ldots, x_m$, then we form filters $f_1, f_2, \ldots, f_m$ based on these features on which the model is trained. Let us consider the features arriving from various IoT devices after extraction are $\hat{x_1}, \hat{x_2}, \hat{x_3}, \hat{x_4}, \ldots, \hat{x_n}$. The following algorithm explains the working of the bridge network.

Algorithm 2: Incremental Retraining Module

```
Result: Decision: Yes / No
x(1)_1, x(2)_2, x(3)_3, ....., x(n)_n ;
while corr (x̂_i, y_j) ≤ 90 do
    FILTER (x̂_1, x̂_2, x̂_3, x̂_4, ....., x̂_n)
    if (newFeature! = 0) then
        Retrain = 1;
    else
        Retrain = 0;
    end if
end while
-----------------------------
function FILTER (x̂_1, x̂_2, ....., x̂_n)
while i ≤ n & j ≤ m do
    if argmin(x̂_i, f[j]) then
        return 0;
    else
        return 1;
    end if
end while
```

In the testing phase, this algorithm takes input values from all IoT devices and the input values are checked by passing through the filters of the sparse autoencoder network. The FILTER function, which is called in Algorithm2, describes the task of new filter addition in the Bridge network. A minimum distance is calculated for each of the features based on each filter. If that minimum distance is satisfied then the feature characteristics are similar to the filter. So we will not consider that feature as new and the value returned to the Algorithm is 0. We set the minimum distance by cosine distance measure, which is used to identify the similarities of feature characteristics by calculating the normalized inner product [32]. If ever the cosine distance is higher than the minimum value calculated, then it is considered as a new feature and added as a new filter to the network. That gives a new set of features, which are used for re-training purpose.

C. CLASSIFIER NETWORK: CLASSIFICATION MODULE

In the training process, we apply a distributed training strategy on three different networks while the testing process is the

same. When all the networks training is completed for the first time, the CNN model is fine-tuned with a small learning rate. We apply the drop-out before the output layer to avoid over-fitting. We model the network traffic patterns as time-series data. The input structure of the first layer in our network depends on the number of inputs received. We convert the 1-dimensional feature vector in to aXb two-dimensional feature matrix. The network is composed of a 2D convolution layer, a 2D pooling layer and a fully connected layer with a 3×3 matrix of the convolutional kernel.

The specifications of the CNN used in the experiment are as follows:

- The 1-dimensional input (x_m) is converted into a matrix axb (in the case of NSL KDD, after the pre-processing phase, 24 features are converted into 3×8 matrix).
- The first convolutional layer consists of a kernel size $= 3 \times 3$, step size $= 1$ and 32 convolutional kernels. Followed by a 2×2 pooling layer and RelU activation function. Two fully connected layers and one drop out layer for regularization. The final output layer is built using softmax activation to give out 2 classes: anomaly or normal.
- The average classification loss is determined between the predicted label $\hat{y}$ and the actual label y by the cross-entropy function. The output label is a binary vector with softmax activation to get one label out.

V. EXPERIMENTAL RESULTS

In this section, we discuss the performance of the proposed method in reducing the complexity of preprocessing and classification tasks. We also describe the datasets used to test our model. Then, we compare the proposed methods with some existing approaches. Besides, we evaluate the model by varying the hyper-parameters. Initially, we focused on feature extraction and later on detection accuracies of the CNN classifier for different datasets. All experiments were performed on the devices acting as the Generative, Bridge, and Classifier networks with the following configurations: Intel(r) Core i7 CPU @ 3.40 GHz and a 64 GB of RAM running 64-bit windows connected in a home network of various smart IoT devices. We have used standard datasets and also real-time data generated from the various devices connected in the smart home IoT health system. The data of each IoT device was preprocessed separately and then sent to the second network (Bridge) for comparison and recording the future incremental features. Then, the same data values are transferred to network 3 for the classification. The time duration is the time elapsed for various tasks which is recorded in seconds and milliseconds. The time duration is measured for the preprocessing task and also for the entire training of the deep neural network.

A. DATASET

1) KDD 99 CUP is the most widely used dataset for intrusion detection training and testing purposes. Although many criticize this dataset for having redundant records that result in biased classification, we have used this dataset for evaluating our model [35] to ensure that our preprocessing technique is efficient in eliminating redundancies and also to record the time that our model takes for different datasets.
2) NSL KDD dataset was proposed to overcome issues of KDD 99 [35] and is considered as a benchmark dataset for intrusion detection. Experimental results using NSL- KDD dataset demonstrate that our method achieves higher accuracy than the other incremental models mentioned in the literature for DoS attack identification.
3) We collected the realistic traffic generated using smart IoT devices. We have collected 2 GB of data by passing normal traffic and also by performing the attacks. The data collected is a combination of benign and malicious traffic. The performed attacks are DoS, Ping of the Death, and Smurf attacks. These attacks were launched one after the other to test if our incremental model is efficient in recording newer features from different types of attacks.

We evaluate three of the significant areas of our proposed methodology; pre-processing task, classification task and incremental learning.

B. PREPROCESSING TASK

The first input vector of the Autoencoder is the incoming traffic from the real-time IoT networking or the standard datasets described above. For each dataset, we divided it into 3 parts: training, validation, and testing with 60%, 20%, 20% data samples in each part respectively. Because our proposed model is distributed, we pre-process the input data on different autoencoder networks (IoT devices) in the training and testing phase. In the case of NSL KDD dataset, we used 30,000 training examples for DoS attack detection with 41 features, which consisted of normal and DoS attack samples in a balanced ratio. The next step is to split this dataset into training testing and validation parts. After which, we divided the input (41 features) on 3 different autoencoder pre-processing networks, with 15, 15, and 11 features on each of the networks. Likewise, we split the feature sets of KDD 99 and the real-time IoT dataset on 3 different AE networks.

1) NETWORK PERFORMANCE WITH DIFFERENT CONSTRAINTS

We plot the Root Mean Square Error (RMSE) of the pre-processing network (Sparse autoencoder network) to record the loss of information after the initial task of learning features. The values shown in Figures 4, 5, 6 describe that for a change in each parameter of the network, the results are different. In these three Figures, SAE represents a normal sparse AutoEncoder using sigmoid activation function, while NSAE and T-P-NSAE represent sparse autoencoder with ReLU activation by incorporating Non-negativity, threshold, and penalty constraints in the network. We have tested RMSE

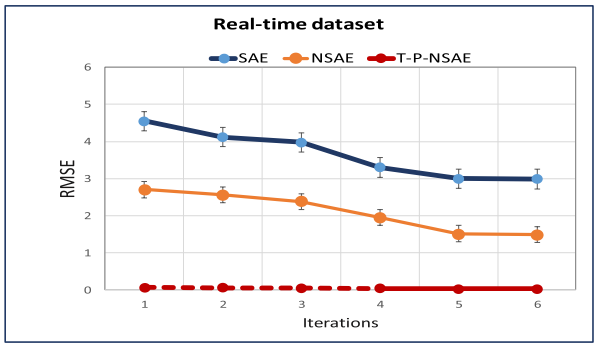

FIGURE 4. RMSE scores of the features on real-time dataset.

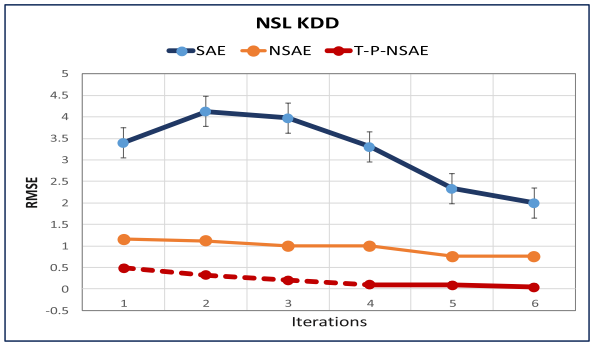

FIGURE 5. RMSE scores of the features on NSL KDD dataset.

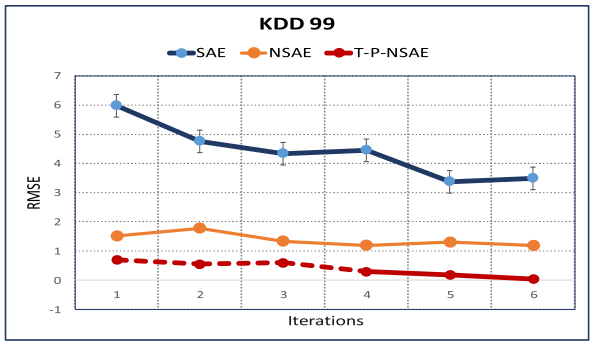

FIGURE 6. RMSE scores of the features on KDD 99 dataset.

score in a step by step procedure by applying each of our constraints that we used in Algorithm 1. It is understood that RMSE score without any constraints for each iteration is very high, whereas it is decreased when we included the non-negativity constraint and it is reached at its lowest that we could achieve by incorporating similarity threshold measure using Agglomerative clustering and penalty factor. With three of the datasets, we achieved fewer scores after the application of all of the constraints discussed previously.

2) RECONSTRUCTION LOSS OF NETWORK WITH A CHANGE OF THRESHOLD

Now we look at the features learned from our preprocessing scheme on 3 different datasets (NSL KDD, KDD99, and real-time IoT traffic). We plot reconstruction error VS similarity threshold so that we understand the performance of our model in extracting unique features. We begin the similarity threshold from 15 (calculated from the heuristic from Algorithm1) and altered it until we reached the lowest reconstruction loss on the data. This comparison is a deciding factor for the value of the threshold that we determined for our algorithm using the Agglomerative clustering technique. Figure 7 shows the initial value of the reconstruction loss of the network that we obtained when we used 15 as the Threshold. For simplicity purposes, we converted the threshold value into natural numbers. Later, we tested by increasing and decreasing the threshold value to know how good the network is performing in extracting features. In three of the cases, increasing the threshold worked for us and we fixed at value 20 for the threshold limit while training the model.

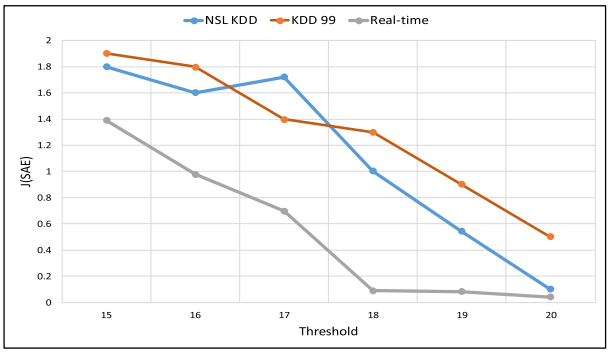

FIGURE 7. Sparse auto encoders reconstruction loss vs threshold.

TABLE 1. Similarities of the features reconstructed.

NSL-KDD	$\leq Min(f_i, f_j)$	KDD99	$\leq Min(f_i, f_j)$	Real-time	$\leq Min(f_i, f_j)$
f1, f2	✓	f1, f2	✓	f1, f2	✓
f1, f3	✓	f1, f3	✓	f1, f3	✓
f1, f4	✓	f1, f4	✓	f1, f4	✓
f2, f3	✓	f3, f2	✓	f4, f2	✓
f5, f6	✓	f9, f8	×	f11, f12	✓
f9, f6	✓	f12, f14	✓	f13, f17	×
f1, f9	✓	f13, f17	✓	f10, f2	✓

3) FEATURES DISSIMILARITY

We evaluated the quality of the extracted features by the distance measure. We chose some random features from different datasets and evaluated their similarity using the cosine distance. We select a minimum distance as per this article where the authors used cosine distance for evaluating the similarity of the features [32]. Table 1 shows the distance between the reconstructed features is less than the minimal cosine distance that is defined for evaluating the uniqueness of the features. From table, we infer that more than 90% of the features generated are unique with no similarities.

C. CLASSIFICATION TASK

In the classification task, Accuracy, Precision, Recall, F1 score, and ROC curves of the model are employed as the evaluation metrics. Precision is the ratio of the number of correctly identified samples to the total number of identified samples. It shows how good our model is in identifying the input samples, while the recall is the ratio of correctly identified samples to the actual correct samples. These two metrics are highly beneficial in benchmarking deep learning

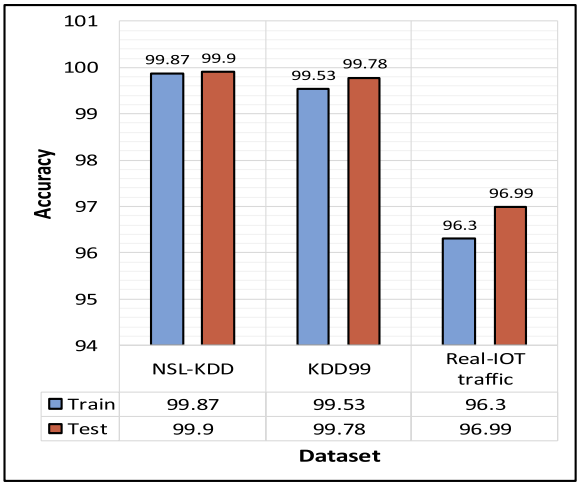

FIGURE 8. Accuracy of the proposed model.

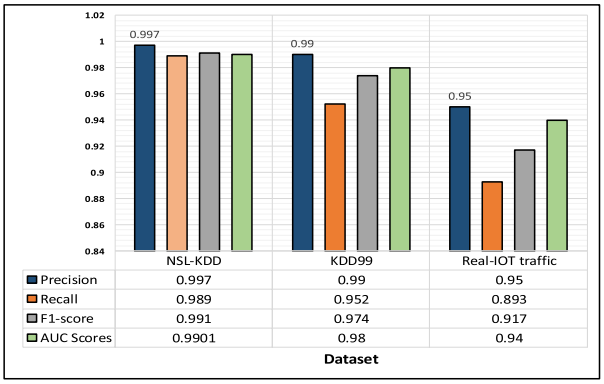

FIGURE 9. Full classification report of the proposed model.

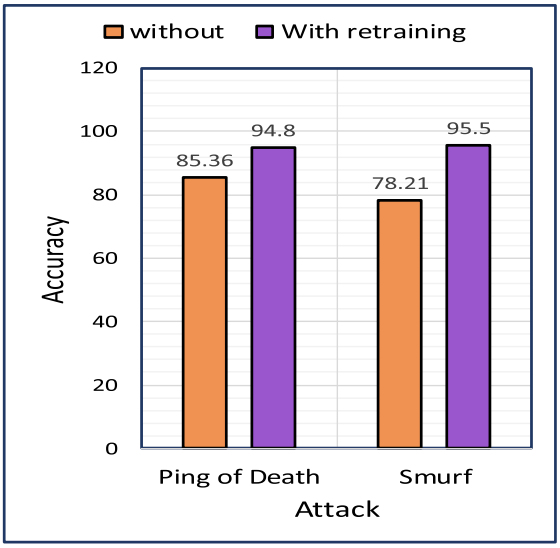

FIGURE 10. Accuracy before and after retraining for 2 categories of attacks.

models. We have divided the dataset into three parts: train, validation, and test so that we validate the data to avoid overfitting. We have also regularized the CNN by incorporating L2 regularization and dropout layers. We evaluate the performance of our classifier model in the following way.

We plot the classification accuracy of our model in the training and testing phases. Figure 8 shows the accuracy and the full classification report is shown in Figure 9 that we achieved by applying our preprocessing technique and Convolutional Neural Network (CNN) classifier. It can be implied that our proposed model performance is better compared to most of the attack detection deep learning models for all the three datasets, with the highest accuracy of 99 for NSL KDD and KDD99, and in between 96 - 97 for real-time IoT traffic for DoS attack detection. The accuracy that we achieved on real-time traffic is because the incoming IoT raw data is imbalanced which affects the classification results. Apart from this, the precision, recall, F1 and AUC scores are uniform with the accuracy results. Training and Testing loss is another significant metric that has to be checked to ensure that our model does not overfit the data and also to confirm that it has acceptable biases and variances. For our model, we have obtained less value in the testing phase when compared to the training phase so we conclude that our model is not overfitting.

D. INCREMENTAL LEARNING MODULE

Once the data arrives from the first network, network 2 (bridge network) records those data values and checks if it fits in the filters on which the Bridge network is made. If any value does not match the filter characteristics based on the heuristic that we have explained in Algorithm 2, then that new value is formed as a new filter. The model is then re-trained based on new and old values. To experimentally check the heuristic proposed, we launched ping of the death attack in IoT smart home environment. The data collected during this attack was passed to the preprocessing networks. After the preprocessing phase and deep feature extraction by our technique, we finalized 6 unique features for that attack identification. In the next stage, we have launched Smurf (another category of DoS) attack. Applying the same pre-processing technique, we found 11 features that are useful in this category of attack identification. When these 11 features were passed to the Bridge network, it filtered those into 9 features, which were useful in both attacks identification. The model accuracy and performance before and after re-training with the newer features are shown in Figure 10.

Initially, the model that was trained using the NSL KDD feature set is tested against the data collected for Ping of the Death attack, for which the classifier gave 85% accuracy. While the pre-processing and comparison phase (on the bridge network) gave a different set of unique features from the data collected during the ping of the death attack. This unique set of features are added as new FILTERS in the Bridge network and the CNN classifier is re-trained using the new set of final features. After the re-training, the accuracy of testing was increased to 94.8% for the same attack. Moreover, the data collected for another attack, Smurf, is also tested using the initial trained model on NSL KDD which gave

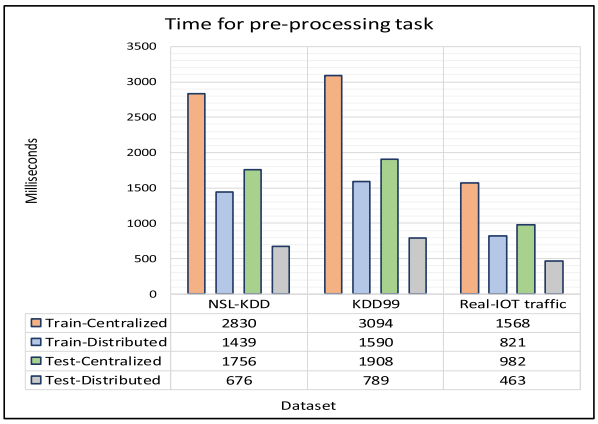

FIGURE 11. Time taken for preprocessing task.

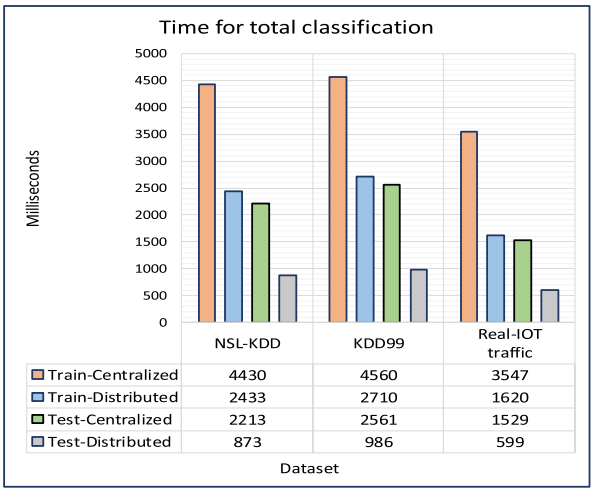

FIGURE 12. Time taken for the classification task.

TABLE 2. Summary of the results.

Dataset / Details	NSL-KDD	KDD99	Real-time data
Total samples	50,000	50,000	50,000
Training split	60%	60%	60%
Testing split	20%	20%	20%
Validation split	20%	20%	20%
Total number of features	41	41	23
No. of different pre-processing networks (no. of IoT devices)	3	3	3
Features split	15, 15, 11	15, 15, 11	9, 8, 7
Extracted features from each	11, 6, 7	7, 12, 6	3, 4, 2
Total no. of features extracted	24	25	9
Time complexity increased (Training) from centralized to distributed	49%	48%	47%
Improvement in Time complexity (Testing) from centralized to distributed	50%	48%	52%

78% of accuracy and was increased to 95.5% after re-training with the features extracted and finalized through generative and bridge networks. The increase in accuracy of detection suggests that the incremental model for recording the newer features is appropriate for identifying the different types of newer attacks that the model is unaware of.

E. TIME COMPLEXITY

After testing the performance of the improved feature extraction technique, we also measure the time taken by our technique in a centralized way and in a distributed way (in parallel by sharing the input vectors among three Autoencoder networks). We record the time during training and testing phases to ensure it is suitable for real-time scenarios. The training complexity of autoencoders is determined by the complexity of the network structure (the type of perceptrons). In our case, we use the multilayer perceptron with one hidden layer of k number of nodes, then the encoder and decoder will have time complexity of O(m. (n + k)) [36] in the pre-processing phase, where n is the number of input nodes. We performed experiment on a dual-core CPU machine with 4 logical processors, at an average speed of 2-3 GHZ. In the NSL KDD dataset, we used 30,000 training examples of 41 features (divided into 3 sets). The results are shown in Figures 11, 12 in the centralized and distributed way (proposed) for pre-processing and classification tasks.

A single training epoch involving the pre-processing steps takes on average 1.4 seconds with NSL-KDD in the distributed setting, where as for KDD99, it takes 1.59 seconds and for the real-time traffic, it takes 0.8 seconds. The input units of the autoencoder are as per the input values of the dataset and real-time values collected from the IoT devices. We also compared the performance against the pre-processing technique in the centralized way. The bar graph in Figure 11 illustrates that the time taken for training and testing phases by centralized (traditional) preprocessing is higher for all the datasets (either standard or real-time IoT traffic). Confirming that the distributed/ parallel (proposed) preprocessing method is more suitable as it takes less time and computations.

We record the time taken for our whole classification task in the training and testing phases. For the classification task, we tested the time elapsed for the whole processes in a centralized way and in a distributed way. Figure 11 shows that the distributed model of the preprocessing and classification tasks takes less time than the conventional centralized way. Finally, we compared the time complexity of our model with those of the existing models. One such CNN based model is developed using Accelerated DNN structure, which was trained on quad-core processors with 8 threads in a parallel mode which makes the execution much faster. According to their proposed mechanism, their model gave 3 times faster performance by training the DNN model in a parallel mode on a quad-core processor. In our case, we have used a dual-core CPU without any accelerator [37]. The time duration recorded suggests that the time taken for our model is less when compared to that model, without modifications to the hardware. Sending properly preprocessed informative data to the classifier saves a lot of time and computations in the training and testing phases. The summary of the results is provided in Table 2.

VI. CONCLUSION

Existing auto-encoder techniques for feature extraction and incremental learning produce duplicate features. In our preprocessing technique, we show that these redundancies can be eliminated and unique features can be extracted by

following proper extraction techniques. We use autoencoders with non-negativity constraints, which help us extract less redundant features. Besides, the distributed intrusion detection model reduces the burden on the classifier and distributes the load between IoT and edge devices. Extensive experiments of the proposed model with standard data sets and real-time IoT traffic gives encouraging results. Theoretical analysis and numerical experiments have shown lower space and time costs than the state of the art techniques, with comparable classification accuracy. In the future, we aim to retrain our incremental model with newer types of attack to make it suitable for multi-class attack scenarios.

REFERENCES

[1] H. Fotouhi, A. Causevic, K. Lundqvist, and M. Bjorkman, "Communication and security in health monitoring systems—A review," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jun. 2016, pp. 545–554.

[2] A. S. Ashoor and S. Gore, "Importance of intrusion detection system (IDS)," *Int. J. Sci. Eng. Res.*, vol. 2, no. 1, pp. 1–4, 2011.

[3] H. Cavusoglu, B. Mishra, and S. Raghunathan, "The value of intrusion detection systems in information technology security architecture," *Inf. Syst. Res.*, vol. 16, no. 1, pp. 28–46, Mar. 2005.

[4] A. Tabassum, A. Erbad, and M. Guizani, "A survey on recent approaches in intrusion detection system in IoTs," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 1190–1197.

[5] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan, "Intelligent feature selection and classification techniques for intrusion detection in networks: A survey," *EURASIP J. Wireless Commun. Netw.*, vol. 2013, no. 1, p. 271, Dec. 2013.

[6] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50850–50859, 2018.

[7] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7167–7177.

[8] Y. Wang, X. Fan, Z. Luo, T. Wang, M. Min, and J. Luo, "Fast online incremental learning on mixture streaming data," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2739–2745.

[9] Y. Yi, J. Wu, and W. Xu, "Incremental SVM based on reserved set for network intrusion detection," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7698–7707, Jun. 2011.

[10] F. Noorbehbahani, A. Fanian, R. Mousavi, and H. Hasannejad, "An incremental intrusion detection system using a new semi-supervised stream classification method," *Int. J. Commun. Syst.*, vol. 30, no. 4, Mar. 2017, Art. no. e3002.

[11] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Survey on incremental approaches for network anomaly detection," 2012, *arXiv:1211.4493*. [Online]. Available: http://arxiv.org/abs/1211.4493

[12] F. Lin, Y. Zhou, X. An, I. You, and K.-K.-R. Choo, "Fair resource allocation in an intrusion-detection system for edge computing: Ensuring the security of Internet of Things devices," *IEEE Consum. Electron. Mag.*, vol. 7, no. 6, pp. 45–50, Nov. 2018.

[13] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu, and J. Li, "A training-integrity privacy-preserving federated learning scheme with trusted execution environment," *Inf. Sci.*, vol. 522, pp. 69–79, Jun. 2020.

[14] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1957.

[15] E. Lahner, M. Intraligi, M. Buscema, M. Centanni, L. Vannella, E. Grossi, and B. Annibale, "Artificial neural networks in the recognition of the presence of thyroid disease in patients with atrophic body gastritis," *World J. Gastroenterology*, vol. 14, no. 4, p. 563, 2008.

[16] C. Xing, L. Ma, and X. Yang, "Stacked denoise autoencoder based feature extraction and classification for hyperspectral images," *J. Sensors*, vol. 2016, pp. 1–10, Jun. 2016.

[17] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture IE*, vol. 2, pp. 1–18, Dec. 2015.

[18] K. Nandhakumar and D. S. Sukumaran, "A hybrid feature extraction method for network intrusion detection system," *IOSR J. Comput. Eng.*, vol. 21, 2019.

[19] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, pp. 215–223, 2011.

[20] C. Shi, B. Luo, S. He, K. Li, H. Liu, and B. Li, "Tool wear prediction via multidimensional stacked sparse autoencoders with feature fusion," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5150–5159, Aug. 2020.

[21] C. Zhang, X. Cheng, J. Liu, J. He, and G. Liu, "Deep sparse autoencoder for feature extraction and diagnosis of locomotive adhesion status," *J. Control Sci. Eng.*, vol. 2018, pp. 1–9, Jul. 2018.

[22] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 125–141, May 2008.

[23] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 41–48.

[24] G. Chechik, U. Shalit, V. Sharma, and S. Bengio, "An online algorithm for large scale image similarity learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 306–314.

[25] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," *Appl. Sci.*, vol. 8, no. 12, p. 2663, Dec. 2018.

[26] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.

[27] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *J. Mach. Learn. Res.*, vol. 6, pp. 1579–1619, Oct. 2005.

[28] J. Li, G. Han, J. Wen, and X. Gao, "Robust tensor subspace learning for anomaly detection," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 89–98, Jun. 2011.

[29] D. Kulić, W. Takano, and Y. Nakamura, "Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden Markov chains," *Int. J. Robot. Res.*, vol. 27, no. 7, pp. 761–784, Jul. 2008.

[30] W.-F. Hsiao and T.-M. Chang, "An incremental cluster-based approach to spam filtering," *Expert Syst. Appl.*, vol. 34, no. 3, pp. 1599–1608, Apr. 2008.

[31] C. Alippi and M. Roveri, "The (Not) far-away path to smart cyber-physical systems: An information-centric framework," *Computer*, vol. 50, no. 4, pp. 38–47, Apr. 2017.

[32] Y. Li, Y. Wang, Q. Liu, C. Bi, X. Jiang, and S. Sun, "Incremental semi-supervised learning on streaming data," *Pattern Recognit.*, vol. 88, pp. 383–396, Apr. 2019.

[33] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[34] M. S. G. Karypis, V. Kumar, and M. Steinbach, "A comparison of document clustering techniques," in *Proc. TextMining Workshop KDD*, May 2000, pp. 1–20.

[35] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.

[36] O. Irsoy and E. Alpaydın, "Unsupervised feature extraction with autoencoder trees," *Neurocomputing*, vol. 258, pp. 63–73, Oct. 2017.

[37] S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced intrusion detection system," in *Proc. IEEE 21st Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2016, pp. 1–8.

ALIYA TABASSUM (Student Member, IEEE) received the integrated dual degree (IDP) (B.Tech. and M.Tech.) in computer science and engineering from Jawaharlal Nehru Technological University, Hyderabad, India, in 2015. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Qatar University. She worked as a Graduate Teaching and Research Assistant with Qatar University from 2017 to 2020. Her research interests include IoT security, wireless communication security, machine learning, and deep learning.

AIMAN ERBAD (Senior Member, IEEE) received the B.Sc. degree in computer engineering from the University of Washington, in 2004, the master of computer science (MCS) degree in embedded systems and robotics from the University of Essex, U.K., in 2005, and the Ph.D. degree in computer science from The University of British Columbia, Canada, in 2012. He is currently an Associate Professor with the College of Science and Engineering, Hamad Bin Khalifa University (HBKU). His research interests include cloud computing, edge computing, the IoT, private and secure networks, and multimedia systems. He received the Platinum Award from H.H. the Emir Sheikh Tamim Bin Hamad Al Thani at the Education Excellence Day 2013 (Ph.D. category). He also received the 2020 Best Research Paper Award from Computer Communications, the IWCMC 2019 Best Paper Award, and the IEEE CCWC 2017 Best Paper Award. He serves as an Editor for the *KSII Transactions on Internet and Information Systems* and the *International Journal of Sensor Networks (IJSNet)*. He has served as a Guest Editor for *IEEE Network*.

AMR MOHAMED (Senior Member, IEEE) received the M.S. and Ph.D. degrees in electrical and computer engineering from The University of British Columbia, Vancouver, BC, Canada, in 2001 and 2006, respectively. He worked as an Advisory IT Specialist with the IBM Innovation Center in Vancouver from 1998 to 2007, taking a leadership role in systems development for vertical industries. He is currently a Professor with the College of Engineering, Qatar University. He has over 25 years of experience in wireless networking research and industrial systems development. He has authored or coauthored over 200 refereed journal and conference papers, textbooks, and book chapters in reputable international journals and conferences. His research interests include wireless networking, and the edge computing for IoT applications. He holds three awards from IBM Canada for his achievements and leadership, and four best paper awards from IEEE conferences. He is serving as a technical editor for two international journals and has served as a technical program committee (TPC) co-chair for many IEEE conferences and workshops.

MOHSEN GUIZANI (Fellow, IEEE) received the B.S. (Hons.) and M.S. degrees in electrical engineering and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a Professor with the Department of Computer Science and Engineering, Qatar University, Qatar. Previously, he has served in different academic and administrative positions for the University of Idaho, Western Michigan University, the University of West Florida, the University of Missouri-Kansas City, the University of Colorado-Boulder, and Syracuse University. He is the author of nine books and more than 600 publications in refereed journals and conferences. He guest edited a number of special issues in IEEE journals and magazines. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid. He is also a Senior Member of ACM. Throughout his career, he received three teaching awards and four research awards. He was a recipient of the 2017 IEEE Communications Society Wireless Technical Committee (WTC) Recognition Award, the 2018 AdHoc Technical Committee Recognition Award for his contribution to outstanding research in wireless communications and Ad-Hoc Sensor networks, and the 2019 IEEE Communications and Information Security Technical Recognition (CISTC) Award for outstanding contributions to the technological advancement of security. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He has served as a member, the chair, and the general chair of a number of international conferences. He is also the Editor-in-Chief of the *IEEE Network*. He serves on the editorial boards for several international technical journals. He also serves the Founder and the Editor-in-Chief for *Wireless Communications and Mobile Computing* journal (Wiley). He has also served as the IEEE Computer Society Distinguished Speaker. He is also the IEEE ComSoc Distinguished Lecturer.