Internet of Things 14 (2021) 100391

Contents lists available at ScienceDirect

Internet of Things

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

Research article

DeL-IoT: A deep ensemble learning approach to uncover anomalies in IoT

Enkhtur Tsogbaatar[a,*], Monowar H. Bhuyan[b], Yuzo Taenaka[a], Doudou Fall[a], Khishigjargal Gonchigsumlaa[c], Erik Elmroth[b], Youki Kadobayashi[a]

[a] *Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma 630-0192, Japan*
[b] *Department of Computing Science, Umeå University, SE-901 87 Umeå, Sweden*
[c] *School of Information and Communication Technology, Mongolian University of Science and Technology, Sukhbaatar 14191, Mongolia*

A R T I C L E I N F O

Article history:
Received 3 September 2020
Revised 10 March 2021
Accepted 11 March 2021
Available online 16 March 2021

Keywords:
Anomaly detection
Software-Defined Networking (SDN)
Deep ensemble learning
Autoencoders
Probabilistic Neural Networks (PNNs)
Internet of Thing (IoT)

A B S T R A C T

Internet of Things (IoT) devices are inherently vulnerable due to insecure design, implementation, and configuration. Aggressive behavior changes, due to increased attacker's sophistication, and the heterogeneity of the data in IoT have proven that securing IoT devices trigger multiple challenges. It includes complex and dynamic attack detection, data imbalance, data heterogeneity, real-time response, and prediction capability. Most researchers are not focusing on the class imbalance, dynamic attack detection, and data heterogeneity problems together in Software-Defined Networking (SDN) enabled IoT anomaly detection. Thus, to address these challenging tasks, we propose DeL-IoT, a deep ensemble learning framework for IoT anomaly detection and prediction using SDN, having three primary modules including anomaly detection, intelligent flow management, and device status forecasting. The DeL-IoT employs deep and stacked autoencoders to extract handy features for stacking into an ensemble learning model. This framework yields efficient detection of anomalies, manages flows dynamically, and forecasts both short and long-term device status for early action. We validate the proposed DeL-IoT framework with testbed and benchmark datasets. We demonstrate that in even a 1% imbalanced dataset, the performance of our proposed method, deep feature extraction with a deep ensemble learning model, is around 3% better than the single model. The extensive experimental results show that our models have a better and more reliable performance than the competing models showcased in the relevant related work.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

The rapid evolution of the Internet of Things (IoT) has brought billions of internet-enabled devices into our daily life to make it smarter by bridging the gap between the physical and the virtual world. Frost and Sullivan [1] have predicted that the number of connected devices will increase up to 45.41 billion by 2023. Autonomous decision making, information to end-users, machine-to-machine and machine-to-user interaction have boosted the acceptance of IoT as a critical asset in the service chain. IoT systems open up several opportunities in areas of autonomous transportation and industrial automation

* Corresponding author.
E-mail addresses: tsogbaatar.enkhtur.ta4@is.naist.jp (E. Tsogbaatar), monowar@cs.umu.se (M.H. Bhuyan), yuzo@is.naist.jp (Y. Taenaka), doudou-f@is.naist.jp (D. Fall), khishigjargal@must.edu.mn (K. Gonchigsumlaa), elmroth@cs.umu.se (E. Elmroth), youki-k@is.naist.jp (Y. Kadobayashi).

https://doi.org/10.1016/j.iot.2021.100391
2542-6605/© 2021 Elsevier B.V. All rights reserved.

[2]. Manufacturers hastily produce new IoT devices without basic security and privacy checks; thus, allowing attackers to easily and swiftly identify vulnerabilities that allow them to evade, manipulate, and take over IoT networks [3]. The failure to implement proper security and privacy measures have already resulted in dire consequences for certain IoT manufacturers and service providers in terms of reputation and financial penalties [4]. Hence, security is becoming crucial to protect IoT devices and applications from cyberattacks in both small and large-scale networks comprised of physical and virtual infrastructures.

The rapid growth of IoT-related vulnerabilities, which is proportional to the exponential production of IoT devices and applications, has created new categories of attacks and malware. For example, Mirai Botnet caused massive Distributed Denial of Service (DDoS) attacks to a DNS server of Dyn DNS provider [5] and brought down several sites including GitHub[1], Reddit[2], Netflix[3], and Airbnb[4]. Multiple zero-day attacks have been observed in complex networks [6]. It is imperative to devise efficient methods for detecting attacks in IoT networks. Traditionally, there are two primary categories of attack detection methods: signature-based and anomaly-based. The signature-based method needs to frequently update the attack signature for known attacks from released signatures of Intrusion Detection System (IDS) vendors [5]. The anomaly-based detection method employs legitimate behavior patterns to detect unknown attacks based on deviation from them [7]. These detection problems intensify in the context of IoT in terms of the following points. *First*, IoT devices do not generate unusual traffic due to limited communication, such as status update and sensor data reading. *Second*, developing a dynamic model that considers device heterogeneity opens up another challenge. *Third*, IoT devices have resource limitations that complicate the deployment of a detection system at the device level. However, the class imbalance problem is one of the main bottlenecks of attack detection algorithms because attack classes are rarer than legitimate classes [8]. For instance, in most of the cases, the number of samples from the anomaly (outlier) class is below 10% of the total number of samples in the entire training set [9]. This data imbalance problem introduces a bias in the machine learning models, that degrades the performance substantially.

SDN has appealing features such as flexibility, dynamicity in network operations and resource management; nevertheless, it has limited scalability [10]. By leveraging SDN, we can provide several attractive benefits for IoT security through controlling the flow dynamically and mitigating attacks at an early stage by rate-limiting the flow at the SDN switches. Hence, developing a deep ensemble learning model as a detection module and deploying it in the SDN framework provide: (i) an isolation of compromised devices, (ii) early detection and mitigation, (iii) a reduction of resource wastage, (iv) an improvement of the accuracy by deploying sophisticated algorithms. Mostly, detection models suffer from class imbalance problems that incur significant performance loss for detecting attacks in IoT. This work takes the benefits of the SDN controller at switches by grabbing the features of data back-and-forth into IoT devices.

Ensemble learning uses multiple algorithms to get better predictive performance than any single one of its constituent algorithms could [11–13]. There are several evidences that prove that ensemble learning model can handle data heterogeneity or class imbalance problems in anomaly detection [14]. Thus, we are motivated to develop a deep ensemble learning model by utilizing deep feature extraction with deep or stacked autoencoder, and feed them to ensemble of Probabilistic Neural Networks (PNNs) for detecting anomalies in IoT. To address the above challenges, we leverage our recent work [15] to present DeL-IoT, an SDN-enabled deep ensemble learning framework for IoT anomaly detection, dynamic flow management, and prediction that utilizes the appealing features of deep autoencoders, probabilistic neural networks and long short-term memory (LSTM). Our proposal uses both device and network switching level features to build an efficient detection and prediction system, which will ensure the increase of IoT device uptime, performance, and forecast device status. We make the following contributions.

- DeL-IoT, a deep ensemble learning approach to uncover anomalies in IoT using SDN.
  - It utilizes the principles of deep autoencoders and probabilistic neural networks.
  - It can detect dynamic attacks and handle the data imbalance problem.
- The deployment of a learned model in SDN controller makes the DeL-IoT system effective and reliable for dynamic flow management.
- DeL-IoT introduces an IoT device status prediction mechanism for forecasting anomalies and taking preventive measures before interrupting a device.
- Systematic and extensive experimental analysis using testbed and benchmark datasets, showing the proposed DeL-IoT is superior to competitors in terms of accuracy, $F1$, Matthews Correlation Coefficient (MCC) measures and prediction scores.

The rest of the paper is organized as follows: Section 2 reviews the related works. Section 3 describes the proposed DeL-IoT framework. Section 4 presents the experimental results and the analysis of our proposed deep ensemble learning model on the testbed and benchmark datasets. Finally, in Section 5, we summarize our findings and suggest possible directions for future work.

[1] https://github.com/
[2] https://www.reddit.com/
[3] https://www.netflix.com/
[4] https://www.airbnb.com/

2. Related work

Several significant works [16–21] have been proposed for anomaly detection in IoT using machine learning approaches with and without SDN. However, most of them were focused to protect either device level or network level, and they did not address the problems together with data imbalance problem, flow management, data heterogeneity, and device status prediction. Bhunia and Gurusamy [17] present a machine learning-based attack detection and mitigation method in IoT traffic using SDN. It uses the SVM algorithm at the SDN controller to monitor and learn the behavior of IoT devices over time, and to detect attacks. However, they evaluated only with a simulation environment known as Mininet, it remains to be seen if the method works in real-time environments.

The anomaly detection studies [9,11,20] show that ensemble learning models have been used to improve the performance of anomaly detection. Because the ensemble learning uses multiple algorithms to get better predictive performance than any single one of its constituent algorithms could [11]. Timcenko and Gajin [20] present an anomaly detection method for the IoT environment using single and ensemble learning approaches and achieved 99% accuracy using the UNSW-NB15 dataset. However, the method lacks testbed experiments and dynamic attack scenarios. Debastrita et al. [9] report an ensemble learning method for outlier detection to handle problems in imbalanced datasets. This method employs stacked autoencoder (SAE) to extract deep features and input them to an ensemble of probabilistic neural network model for single and multiple outliers detection. The use of autoencoder makes the method's performance better and more stable.

Unfortunately, only few studies address the data imbalance problem in IoT anomaly detection [22]. For instance, Zolanvari et al. [22] evaluate the efficiency of Artificial Neural Network (ANN) for anomaly detection in imbalanced Industrial IoT (IIoT) testbed dataset. This method uses Synthetic Minority Over-Sampling Technique (SMOTE) for improving the performance of ANN in imbalanced IIoT dataset. Also, they analyze the limitations of machine learning-based intrusion detection solutions.

In recent years, advances in deep learning have transformed many areas of data-driven modeling [16,18,23–25]. Yair et al. [18] come up with a network-based anomaly detection method that uses deep autoencoders and to detect compromised traffic of IoT devices. This method is validated by using nine different commercial IoT devices in a laboratory environment with two famous IoT-based botnets, i.e., Mirai and BASHLITE. This method lacks variation in attack scenarios. To combat this, Nickolas et al. [23] present a BoT-IoT dataset, comprising legitimate and simulated IoT network traffic along with various types of attacks. Each existing datasets assessed against diversity, complexity, and experimental environments for evaluation of IoT attack detection algorithms. They validate the reliability of the Bot-IoT dataset by using statistical and machine learning algorithms. Thien et al. [24] present DÏoT, an autonomous self-learning distributed system for detecting compromised IoT devices using GRU (Gated Recurrent Units). DÏoT system creates device specific profiles without human intervention nor labeled data. The results enumerated that DÏoT detects IoT attacks in 257 milliseconds on average with a 95.6% True Positive Rate (TPR). Mahmudul et al. [16] propose an anomaly detection method for IoT sensors in IoT sites using machine learning approaches. This method is evaluated by using an open-source dataset [26] with seven classes of attacks. They achieve 98.2– 99.4% accuracy for random forest and artificial neural networks. In a more recent work, Hwang et al. [25] present an unsupervised deep learning model for early network traffic anomaly detection in IoT using Convolutional Neural Network (CNN) and autoencoder known as D-PACK. D-PACK detects malicious traffic with nearly 100% accuracy with less than 1% false positive rate where it examines two packets per flow. Still, several opportunities are left to address IoT security problems in small and large-scale industries with a comprehensive focus in either centralized or distributed environments. Some problems include the increased uptime of devices that are deployed in the edge of the network, data or device heterogeneity, reliability, device status prediction, dynamic flow management, and dynamic attack detection. A comparison of the existing IoT anomaly detection methods is summarized in Table 1.

3. System model

We present DeL-IoT, a deep ensemble learning framework to uncover IoT anomalies using SDN, primarily to detect anomalies and dynamic attacks for increasing device uptime, detection efficiency, switch-level dynamic flow management, and device status prediction. This framework aims to provide security for IoT devices by monitoring traffic and system metrics together in SDN switches. Also, it can handle the data imbalance problem where attack classes are rarer than legitimate classes. An architecture of DeL-IoT system is given in Fig. 1.

This framework has three primary components: including SDN controllers, SDN switches, and IoT devices. Further, the proposed framework has data collection and preprocessing, a learning module, a detection module, a flow management module, and the maintenance of a status table for forecasting device status. The network operators employ the SDN-enabled framework to isolate the services, increase reachability, improve service-oriented policies at the switch-level. The SDN controller disintegrates policies into service-specific rules and colonizes into flow tables of SDN-switches through the standard channel like OpenFlow [27]. Each packet is forwarded based on the enabled rules in the flow table. Each rule has three most common fields including *matching field, actions*, and *flow counters*. If a packet header matches a rule, then the controller must take actions (e.g., forwarding to a specific device) and update the counters immediately. We assume that the controller has complete information of network topology and can make a request for each counter rule from switches [28]. A new rule is installed or updated reactively when new flows come to the network without any matching rules. In the following subsections, we discuss the anomaly detection, flow management and maintenance of the status table for forecasting device status.

Table 1
Comparison of existing IoT anomaly detection methods .

Author and Year	Detection method	Botnets	Dynamic attack detection	Deployment level	Datasets	Performance	Class imbalance problem
Bhunia et al. [17], 2017	SVM	unknown	Yes	SDN controller	Mininet emulator	Precision-98% Recall-94%	No
Timcenko et al. [20], 2018	LADTree, Random Forest, REPTree, MultiBoost, SMO	unknown	No	Networks	UNSW-NB15	ROC-55%-99%	No
Maede et al. [22], 2018	ANN with SMOTE	unknown	No	Networks	Testbed IIoT control system	MCC-19%-99.86%	Yes but partially
Yair et al. [18], 2018	Autoencoder and other machine learning algorithms	Mirai, Bashlite	Yes	Network	N-BaIoT	TPR-75%-100%	No
Nickolas et al. [23], 2019	LSTM, RNN, SVM	unknown	No	Networks	Bot-IoT	Accuracy-88%-99%	No
Thien et al. [24], 2019	A federated self-learning Gated Recurrent Units (GRUs)	Mirai	No	Networks	Testbed IoT devices DÏoT	TPR-95.6%	No
Mahmudul et al. [16], 2019	machine learning algorithms	unknown	No	Networks	DS2OS traffic traces	Accuracy-98.2%-99.4%	No
Hwang et al. [25], 2020	CNN and autoencoder	Mirai	No	Networks	USTC-TFC2016, Mirai-RGU, Mirai-CCU	Accuracy-99.39%-100%	No
DeL-IoT	deep ensemble learning	Mirai, Bashite, Bonesi	Yes	SDN controller, Networks	N-BaIoT, Testbed IoT devices data	F_1 Score 99.5–99.9%, MCC 91.04%-99.95%	Yes

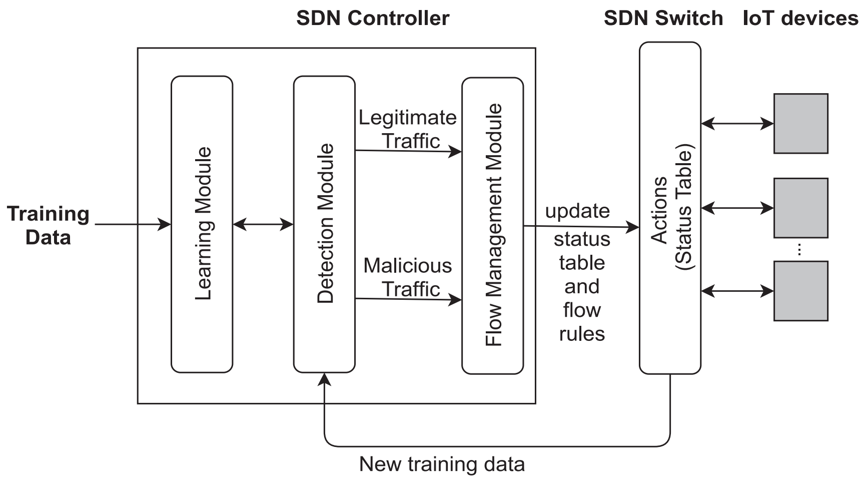


Fig. 1. DeL-IoT system architecture.

3.1. Anomaly detection

The DeL-IoT framework aims to uncover anomalies in IoT based on the dynamic observation of both packet and flow level traffic instances that pass through SDN switches as well as system metrics. We deploy the detection module that can monitor traffic and system metrics of deployed devices as well as applications for anomaly detection.

3.1.1. Learning module

Anomaly detection models have a vital requirement to have aggregated data either at an endpoint or from multiple sources. The model adopts packet level, flow level, and system metrics data to detect anomalies in IoT. Because the attackers primarily target different performance or system metrics. We validate this model by using both testbed and benchmark datasets. For testbed data, we set up a testbed comprising multilevel hierarchical architecture from physical infrastructures to IoT devices. The preprocessing function extracts significant features that the attackers typically utilize and labels them based on the behavioral analysis. In addition, the feature extraction procedure of testbed data is shown in Algorithm 1. For the benchmark data, we use a recent dataset, called N-BaIoT [18] for our experimentation. We provide extended explanations of each dataset in Section 4.

Algorithm 1 Flow Feature Extraction

```
Input: Raw flow parameters
Output: Extracted flow features
function MAIN(path)                                          ▷ get raw flows
    filter=[duration, protocol, srcIP, desIP, srcPort, desPort, packets, bytes, tos, idle_age]
    flowDic=OPENFILE(path, filter)          ▷ extract raw parameters and store in csv format
end function
function OPENFILE(path, filter)
    for line to file do
        if line.startswith ”NXST_FLOW” in line:
            continue
        dic = PARSEF(line, filter)
        flowDic.append(dic)
    end for
    return flowDic
end function
function PARSEF(line, filter)
    for F in lists do                                        ▷ feature $F$ in the list
        if (l > 1) then                                      ▷ if it's not a protocol
            estimate $T = (t_2 - t_1)$ and $F_n = \{f_1, f_2, \cdots f_n\}$   ▷ estimate interflow time, T, and extract all features, F
            flowDic.append(T, F)
        else
            flowDic.append(P)                                ▷ if it's a protocol
        endif
    endfor
    return dic
end function
```

To make an efficient and automated representation of features, we use autoencoder and deep feature representation by a non-linear transformation of features set before feeding data into the learning model. This module employs both legitimate and anomalous features or system metrics to learn the model for anomaly detection in IoT. Let's assume that $X_n = \{x_1, x_2, \dots, x_n\}$ is the input data, $n \in \mathbb{R}$, $X'_n = \{x'_1, x'_2, \dots, x'_n\}$ is the encoded output, $F_n = \{f_1, f_2, \dots, f_n\}$ is the features set, and h_n is the set of hidden layers.

Autoencoder and deep feature representation are multilayer neural networks having multiple hidden layers, h, to encode the input and to reconstruct the output as similar as possible to the input.The network has two parts: an encoder and a decoder. An encoder is defined as $E_n = f(w_1 X_n + b_1)$, where f is the encoding function with w_1 as weight vector, and b_1 is the bias. A decoder is defined as $X'_n = g(w_2 h_n + b_2)$, where g is the decoding function with weight matrix w_2, and bias b_2 [29]. Further, each parameter of the autoencoder is optimized to minimize the reconstruction error. We employ two categories of autoencoders, stacked autoencoder (SAE) and deep autoencoder (DAE) to extract and represent the features set obtained from the preprocessing function. The primary advantage of using deep autoencoder is having multiple hidden layers as shown in Fig. 2. More hidden layers incur better feature representation, which is advantageous for an anomaly detection model [30].

Additionally, stacked autoencoder learns from the initial input data and obtained features make input to the next layer for reduced and compact features set (see Fig. 3). The under-complete autoencoders have a lower number of nodes in hidden layers.

3.1.2. Detection module

This module employs the outcome of a learning module for detecting anomalies in IoT within an SDN-enabled framework. We explain the components of the detection module below.

Probabilistic Neural Networks (PNN) is a multilayered feedforward network with four primary layers, including input, pattern, summation, and output layers, as shown in Fig. 4. The PNN is represented as a Kernel Discriminant Analysis (KDA), which is the generalization of Linear Discriminant Analysis (LDA), to find the linear combination of features that separate classes. A PNN consists of several sub-networks that estimate the Parzen window probability density function of each class using the samples of training set. Each node of the network calculates the probability density function for input x, training sample x_i, and class c_k according to the following equation.

$$p(x|x_i, c_k) = \frac{1}{\sigma}\omega\left(\frac{x - x_i}{\sigma}\right) \tag{1}$$

where x_i is the i^{th} sample and x is the input instance (unknown), $\omega()$ is the weighting function and σ is the smoothing parameter. The nodes are grouped according to the classes of the training sample in the pattern layer, and each group sums

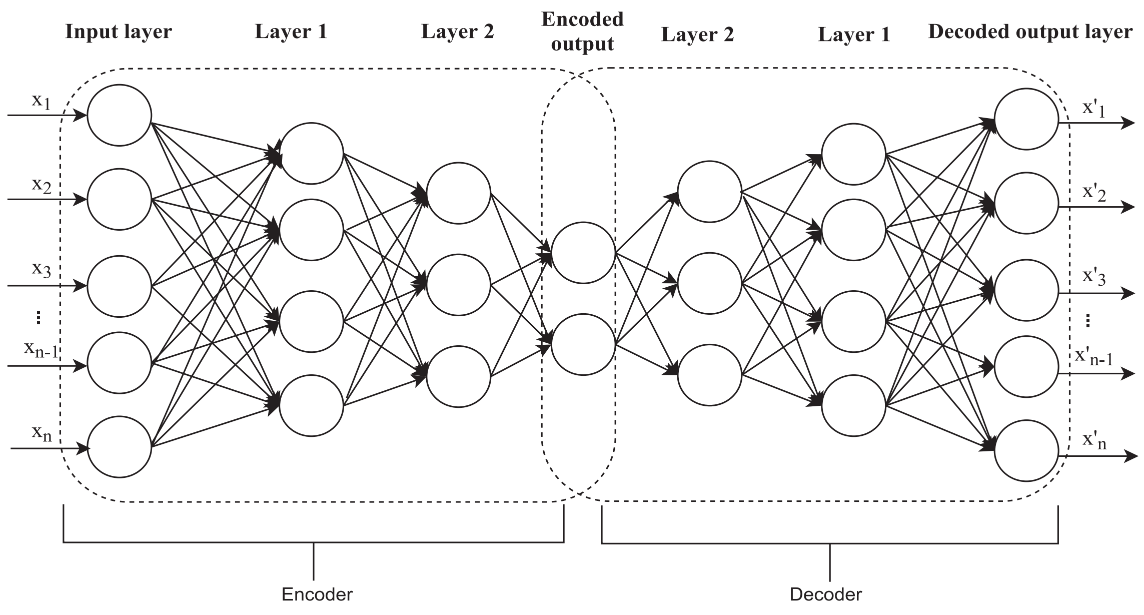


Fig. 2. Deep autoencoder.

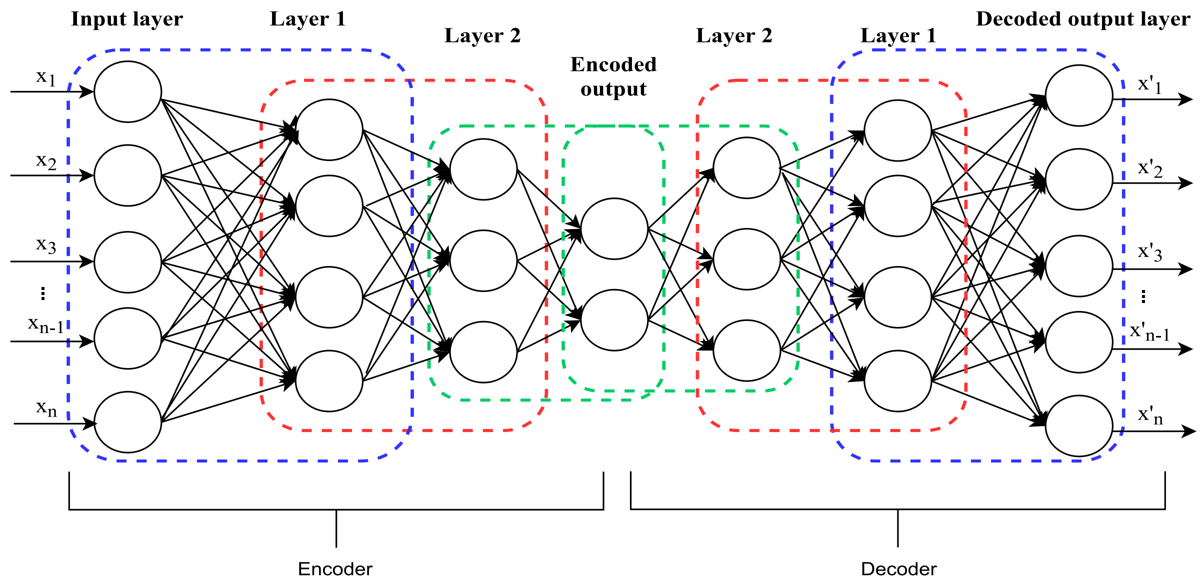


Fig. 3. Stacked autoencoder.

up for the next layer to get the class-wise probability. In the summation layer, the c_k^{th} nodes aggregate the values from the pattern layer of c_k^{th} classes. This summation is estimated based on mixed Gaussian or Parzen window estimator as defined in the following equation.

$$f_{c_k}(x|x_i, c_k) = \frac{1}{n\sigma}\sum_{i=1}^{n_{c_k}} \omega\left(\frac{x - x_i}{\sigma}\right). \tag{2}$$

where n_{c_k} is the number of samples in c_k^{th} classes. Hence, the summation layer maps the c_k^{th} nodes to the c_k^{th} classes. For new samples, $f_{c_k}(x)$ can be estimated without retraining. The PNN needs more samples to achieve a high probability of mapping score from input instances to underlying classes where $f_{c_k}(x)$ has a maximum posterior probability of a class.

The weight function $\omega()$ is chosen as a kernel function (e.g., Radial Basis Function (RBF)) to compute the distance between the known and unknown sample points. If the distance is nearest, then it has more influence on the end class. The use of PNN provides multiple benefits, including insensitiveness to an outlier in the data, new input patterns stores in the

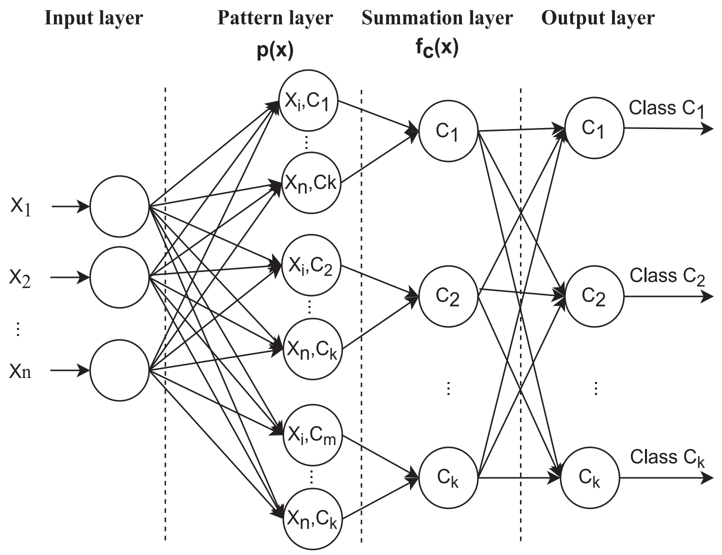


Fig. 4. Architecture of Probabilistic Neural Networks (PNN).

network, and the smoothing parameter σ. Additionally, it reduces the retraining of the network if the training samples become large. Each sub-network of PNN implies a Parzen density estimator for a particular class. These features boost the detection of exceptional events in the data. However, it has been observed that PNN alone cannot provide better results. Because PNN, as a system, has a large storage requirement. Hence, we have used to handle such a high storage requirement by dimensional reduction using deep and stacked autoencoders and integration with ensemble probabilistic neural networks to detect anomalies in IoT.

Deep autoencoder and stacked autoencoder ensemble probabilistic neural networks (DAE-EPNN and SAE-EPNN) are autoencoders integrated with ensemble probabilistic neural networks. They encode the input by using multilayer neural networks instead of stacked layers. In anomaly detection, anomalous classes are rare, whereas legitimate classes are frequent. Hence, binary classifier gets more biased performance [8]. Such classifiers can be used to refine the decision boundary between the rare and frequent classes. In the proposed method, we have used deep and stacked encoders PNN for encoding the input and feed them into PNN for classification. These inputs include samples from both rare and frequent classes. In anomaly detection, we have to make the trade-off between generalization and specialization to refine the decision boundary for achieving high accuracy. Most anomaly detection models are not specialized except just giving a bias to rare classes. The proposed model mitigates these drawbacks and gains substantial performance improvement thanks to deep ensemble learning.

In PNN, the smoothing parameter σ determines the spread of RBF when it reaches a peak in the center of weighting. Selecting optimal values of σ implies a better spread of RBF in PNN. A shallow value of σ causes the model to over-fit whereas an extremely high value may cause the model to under-fit. However, both factors are essential to address the data imbalance problem during anomaly detection in IoT. These problems motivate us to develop a deep ensemble learning model, which we explain below.

Deep Ensemble Learning employs multiple PNNs with multiple layers as weak classifiers to address the biases by fine-tuning the smoothing parameter σ. This model takes inputs from the encoded features of the deep autoencoder PNN to construct inputs for the next layer. Let's assume that $A = \{x_{a1}, x_{a2}, \ldots, x_{an}\}$ is the set of anomalous instances, $L = \{x_{l1}, x_{l2}, \ldots, x_{ln}\}$ is the set of legitimate instances, D is the training dataset, Y is the test dataset, tr and te indicate training and testing instances.

As the legitimate instances are more than attack instances, we divided the legitimate instances into N subsets and we kept anomalous instances as one class for training. We used N number of PNNs with multiple layers for deep ensemble learning, where i^{th} PNN is trained with i^{th} subset of datasets. Hence, we chose i^{th} PNN for training with $(i+1)^{th}$ PNN for binary classification. However, we chose N^{th} PNNs for the majority of the classes and one more PNN for an anomalous class, specifically for the multiclass problem. For this, we used deep encoder, since we know that encoder provides the non-linear transformation of input data to reduce features set. A pictorial representation of the proposed model is given in Fig. 5.

Let gbe a non-linear activation function with weight w and bias b then the deep encoder is formulated as in Eq. (3). The choice of parameters is explained in Section 4.

$$\begin{aligned} \varepsilon &= \Big(g(wx+b)\Big) \\ DAE(x) &= \Big(\varepsilon_1(\varepsilon_2(\varepsilon_3(\cdots \varepsilon_h(x))))\Big) \end{aligned} \tag{3}$$

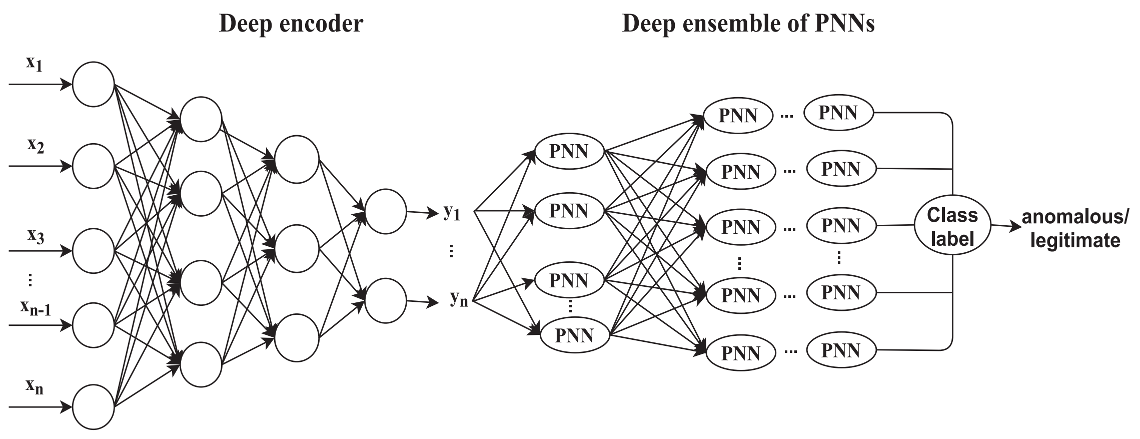


Fig. 5. The proposed method: an integration of deep encoder and deep ensemble of PNNs.

where $\varepsilon()$ is the encoding function whereas $\varepsilon_i()$ is the i^{th} deep encoder, h is the number of hidden layers, each feature vector x is transformed to $\hat{x}$ using DAE(x) defined in Eq. (4). The encoded features set, $\hat{x}_1, \hat{x}_2, \ldots \hat{x}_{F'}$ is embedded to deep ensemble learning model.

$$z^n_{c_k}(\hat{x}) = f^n_{c_k}\Big(DAE(x)\Big) \tag{4}$$

Each input instance $\hat{x}$ is assigned to c_k classes based on intermediate RBF score received from the n^{th} PNNs, where f estimates intermediate RBF scores based on encoded features set to decide belongingness of a class. However, if there are N-PNNs for the ensemble, then each instance $\hat{x}$ is assigned to c_k classes based on Eq. (5). The classification score is computed using Eq. (6) for each instance $\hat{x}$ that belongs to a specific class and where n represents batch size.

$$z_{c_k}(\hat{x}) = \sum_{n=1}^{N}\left(z^n_{c_k}(\hat{x})\right) \tag{5}$$

$$s_{c_k}(\hat{x}^n_1) = \frac{z_{c_k}(\hat{x}^n_1) - min(z_{c_k}(\hat{x}^n_1))}{max(z_{c_k}(\hat{x}^n_1)) - min(z_{c_k}(\hat{x}^n_1))} \tag{6}$$

Once we get the classification score for each instance, we label the unknown instance $\hat{x}$ as anomalous or legitimate based on the node's maximum probability, $p_{c_k}(\hat{x}) = max(s_{c_k}(\hat{x}))$. Each layer of deep ensemble learning employs majority voting to classify anomalies in IoT. This process repeats for the next layers of the deep PNNs to improve overall performance. The major steps in deep ensemble learning to uncover anomalies are defined in Algorithm 2.

Algorithm 2 DeL-IoT : a deep ensemble learning

```
Input: IoT dataset, D
Output: Uncover $D$ as legitimate or anomalous
 1: normalize original dataset $D$ using MinMaxScalar, and to get $D_1$
 2: train SAE or DAE on $D_1$ with hyper-parameters to obtain $D_2$    ▷ $D_2$ represents deep and compact features set.
 3: if ($D_2 := s_a$) then                                             ▷ $s_a$ indicates single attack
 4:     chose adam optimizer and binary_crossentropy as a loss function
 5: else
 6:     chose adam optimizer and categorical_cross_entropy as a loss function
 7: endif
 8: construct feature set, $D_2$, by relu activation values of hidden layers in SAE or DAE using Eq. (3–4)
 9: Split $D_2$ for training dataset $D_{tr}$ and for testing dataset $D_{ts}$
10: for $j = 1$ to number of ensemble PNNs do
11:     train PNN on $D_{tr}$ using Eq. (1–2)
12:     for $i = 1$ to number of $D_{ts}$ do
13:         compute maximum class probability score using Eq. (6)
14:         classify $D_{ts}[i]$ samples using Eq. (5)
15:     end for
16: end for
```

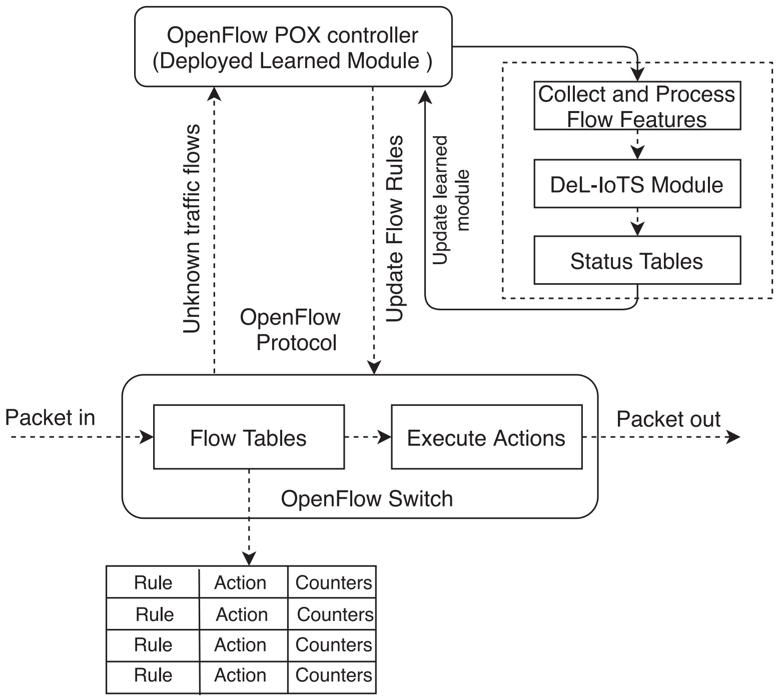


Fig. 6. Reactive flow management architecture.

Table 2
The proposed IoT device status table .

Device ID	Time	Status
dl_source	system_time	1-Anomalous 0-Legitimate

3.2. Flow management

This module is enabled to prepare appropriate rules for each attack and to dynamically update the rules in the SDN switch as a set of actions. IoT devices are connected to the SDN switches. We employ the features of the POX framework [31] with OpenFlow protocol [28] to enable flow control and management at the SDN switch. In addition, SDN switches continuously monitor traffic flows of IoT devices and provide statistical information to the SDN controller.

We consider three scenarios for dynamic flow management in DeL-IoT framework that leverage SDN-enabled POX controller [31]. First, if the input flow is legitimate then it passes through the switches immediately without any interruptions. Further, if the flow is unknown then the SDN switch sends it to the deployed detection module for further investigation and, for the time being, applies rate-limiting to the traffic to control intended malicious flow within the networks. For instance, this unknown flow may end-up with an intended attack that trims the overall network performance. Second, if the input flow is detected as anomalous then a set of rules are applied to control them. At the beginning, the flow is dropped immediately and the deployed detection module investigates the source of the attacks to blacklist them. Hence, the attackers cannot reach any IoT devices to damage the entire network. Further, if a number of devices are under attack, then network-wide rules will be updated and applied to maintain network performance. Third, if the IoT devices are compromised then two actions are usually taken. The flow of traffic will be immediately blocked and investigated further for verifying the kind of malicious flows: malware or physical attacks.

The SDN enabled POX controller for managing dynamic traffic flow and generating status table for IoT devices is shown in Fig. 6. Based on the IoT device profile, this module inserts an entry into the status table while considering the status for a time period. It enters 0 for legitimate status and 1 for anomalous status. This status information is further utilized to forecast the device status for short and long-term using sequence modelling algorithm.

3.3. Forecasting device status

This module takes input from the flow management module and generates profiles for each IoT device with the outcome of metrics such as packet, flow, and system. This table is comprised of three columns, including time, device ID, and status, as shown in Table 2. The status of each IoT device is marked either as 0 (legitimate) or as 1 (anomalous) for each time point.

Based on the status table information, we will predict device future status using a sequence modelling algorithm based on LSTM, steps are shown in Algorithm 3. The system manager utilizes this feature to easily handle large-scale IoT devices as

Algorithm 3 LSTM-based IoT device status forecasting

Input: IoT device status information
Output: Forecasting IoT device status

```
1: initialize time, $t$ and next time step, $t+1$, loop_back=1, dataset D
2: Split $D$ for training dataset $D_{tr}$ and for testing dataset $D_{ts}$
3: $D_{tr}$ and $D_{ts}$ convert the status array into data matrix, $Dx_{tr}$=t,$Dy_{tr}$=t+1, $Dx_{ts}$=t and $Dy_{ts}$=t+1
4: $Dx_{tr}, Dy_{tr}$=CREATE_DATASET($D_{tr}$,loop_back) and$Dx_{ts}, Dy_{ts}$=CREATE_DATASET($D_{ts}$,loop_back)
5: trainPredict = lstm.predict($Dx_{tr}$) andtestPredict = lstm.predict($Dx_{ts}$)    ▷ device status prediction
6: function CREATE_DATASET(dataset, loop_back=1)
7:     dataX, dataY = [], []
8:     for i in range(len(dataset)-look_back-1) do
9:         a = dataset[i:(i+look_back), 0]
10:        dataX.append(a)
11:        dataY.append(dataset[i + look_back, 0])
12:    end for
13:    return dataX, dataY
14: end function
```

well as their services to the end-users.

4. Performance evaluation

This section reports and explains the intensive experimental results obtained from a testbed and benchmark datasets. We begin with datasets description and proceed with experimental results.

4.1. Datasets

The DeL-IoT framework is evaluated using two datasets: (i) testbed data, and (ii) benchmark data. The testbed data is generated with a significant amount of attacks on IoT devices including applications in the physical infrastructures.

Testbed data: The experiment is performed in a virtualized environment with a hierarchical deployment of IoT devices to physical servers in the testbed. Fig. 7 illustrates the architecture of the testbed setup. The testbed is comprised of multiple servers and applications at a physical level, virtualized level, and IoT devices. We consider multiple VMs, one of the VMs is a target of attackers. We generate both Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks using the Targa2[5] attack generator, the D-ITG internet traffic generator [32], the BoNeSi botnet simulator[6], and the stress-ng[7] system resources load generator tools. We generate multiscale attacks [33] to the IoT devices including applications in the physical infrastructures when they are deployed in the virtualized environment. We collect multiple metrics (e.g., packet, flow, system metrics (Key Performance Indicators), device status, etc.) from devices to physical infrastructures for learning and deploying the proposed model. The number of instances of the testbed dataset is given in Figs. 8, 9, 10 where we consider 1–9% attack instances for data imbalance scenarios in our experiments. POX[8] is an OpenFlow controller used to deploy the learning algorithm to detect anomalies in IoT devices based on dynamic policy updating within the SDN framework and also for short and long term forecasting of IoT device status.

Benchmark data: Due to the non-availability of benchmark datasets, we used the N-BaIoT [18] dataset for our experiments. This dataset was prepared using two attack generation tools, i.e., Mirai (scan, ACK flooding, SYN flooding, UDP flooding, UDPplain attacks) and Bashlite (scan, junk, UDP flooding, TCP flooding, COMBO attacks), with 9 commercial IoT devices. There are 5 Bashlite, 5 Mirai, and 1 legitimate datasets, having 115 features in each of them. We created an imbalanced dataset by separating 98,514 legitimate and 9850 anomalous instances with a combination of 10 different kinds of Mirai and Bashlite attacks.

In addition to dynamic attack detection in IoT, we also address the data imbalance problems. Hence, we have created imbalanced datasets with variation of dynamic attacks from N-BaIoT dataset. Figs. 8–10 illustrate the multiple scenarios of imbalanced dataset, where we consider 1–9% attack instances.

[5] http://packetstormsecurity.com/
[6] https://github.com/Markus-Go/bonesi
[7] https://kernel.ubuntu.com/~cking/stress-ng/
[8] https://github.com/noxrepo/pox

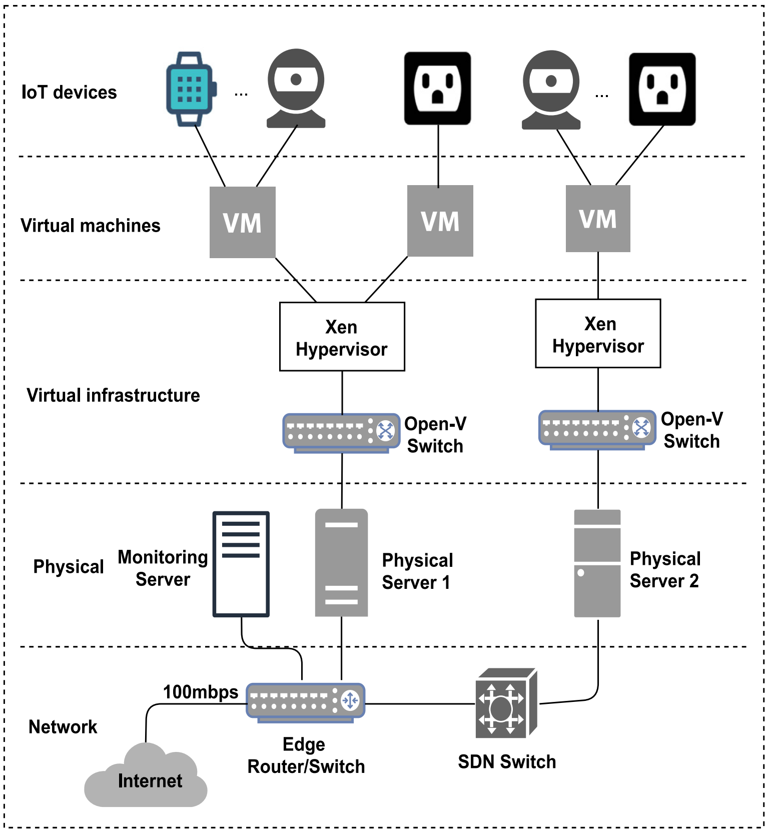


Fig. 7. Testbed setup.

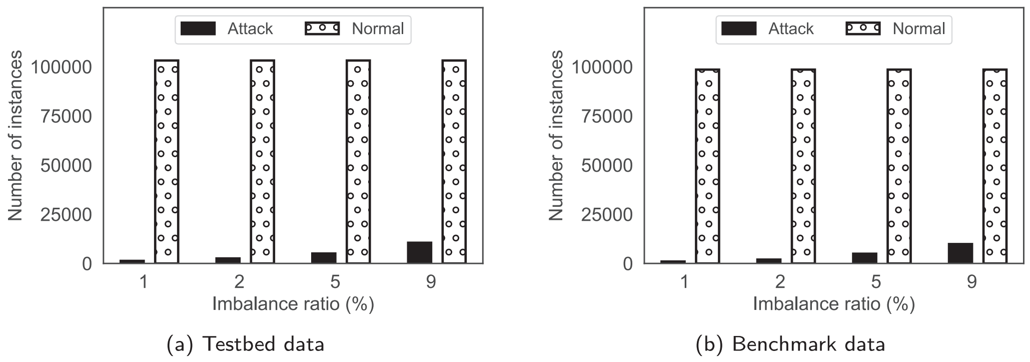
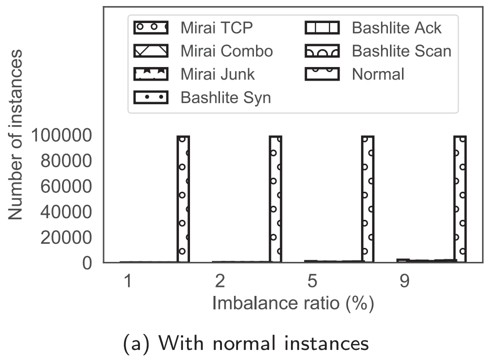


Fig. 8. Single attack imbalanced datasets.

4.2. Performance measures

This section explains the performance measures used to validate the DeL-IoT framework. We explain the multiple measures to complement each other and ensure efficacy measures of the proposed models. These measures include confusion matrix, precision, recall, F1-measure, Matthews Correlation Coefficient (MCC). Using these performance metrics, one can decide which model is performed well and best suited for the proposed framework when addressing multiple problems in IoT.

Confusion matrix (CM) is used to describe the performance of a classification model on a set of test data. The diagonal represents the correct classification. The confusion matrix for attack detection is defined as a 2-by-2 matrix, since there are only two classes known as anomalous and legitimate. Thus, the TNs (True Negative) and TPs (True Positive) that represent the correctly predicted cases lie on the matrix diagonal while the FNs (False Negative) and FPs (False Positive) are on the



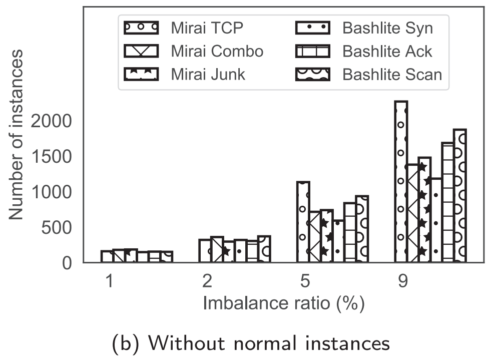
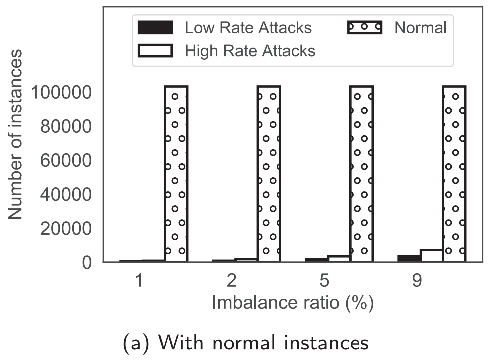


Fig. 9. Multiple attack imbalanced benchmark datasets.



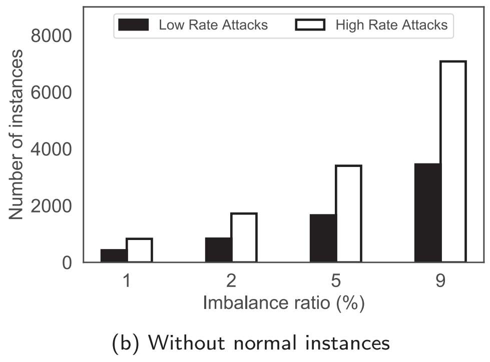


Fig. 10. Multiple attack imbalanced testbed datasets.

right and left sides, having wrong predictions [34]. We employ accuracy, precision, recall and F_1 measures as evaluation metrics in addition to MCC.

Matthews Correlation Coefficient (MCC) measures the quality of the classification, showing the correlation agreement between the observed values and the predicted values [22]. MCC is a great metric, especially in case of imbalanced datasets, it is given in the following equation.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{7}$$

4.3. Results

The proposed framework is evaluated using both testbed and benchmark datasets. We present the results based on testbed datasets that consider multilevel metrics such as packet, flow, and system metrics in synchronization for validation. In the single attack experiment, we used deep and stacked autoencoders with an ensemble of PNNs with the 'adam' optimizer and 'binary-crossentropy' as a loss function. Likewise, in the multiple attacks, we used deep and stacked autoencoders with an ensemble of PNNs with the 'adam' optimizer and 'categorical cross-entropy'as a loss function. First of all, we do some experiments as to how our proposed model results vary with values of σ and the depth of the DAE or SAE network used. Second, based on the experimental results, the hyper-parameters were chosen by optimizing the validation set. Finally, the model was pre-trained with 100 epoch, 100 batch size, and the "relu" activation function by considering four hidden layers with the compositions 17-16-16-15 for testbed data and 115-100-50-25 for benchmark data. Regarding the testbed and benchmark datasets, we have used non-overlapping datasets: 70% for training and 30% for testing, so that the class imbalance problem remains the same in both the training and the test set. However, full data was considered for training of the deep or stacked autoencoders and then we fed them to deep ensemble of PNNs.

4.3.1. Characterization of data

To observe the behaviour of both testbed and benchmark datasets, we estimate the cumulative distribution function and kernel density for legitimate and attack instances shown in Figs. 11 and 12, respectively. From the Figures, it is clear that the distribution of legitimate instances differs from attack instances.

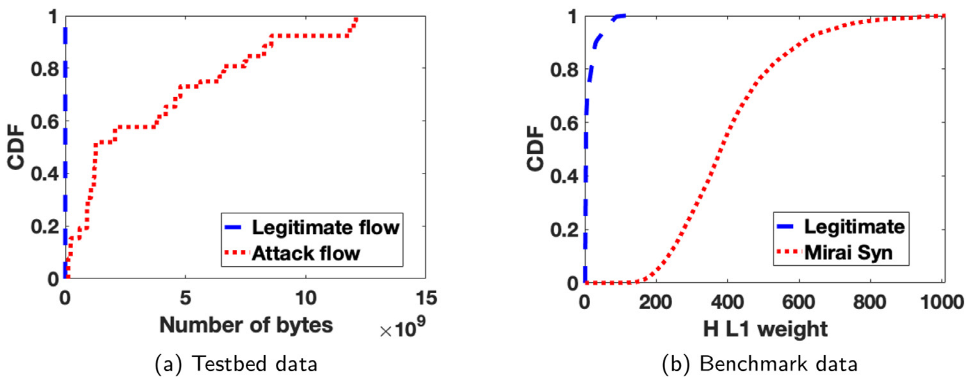


Fig. 11. Characterization of data: cumulative distribution function for legitimate vs. attack instances over feature set.

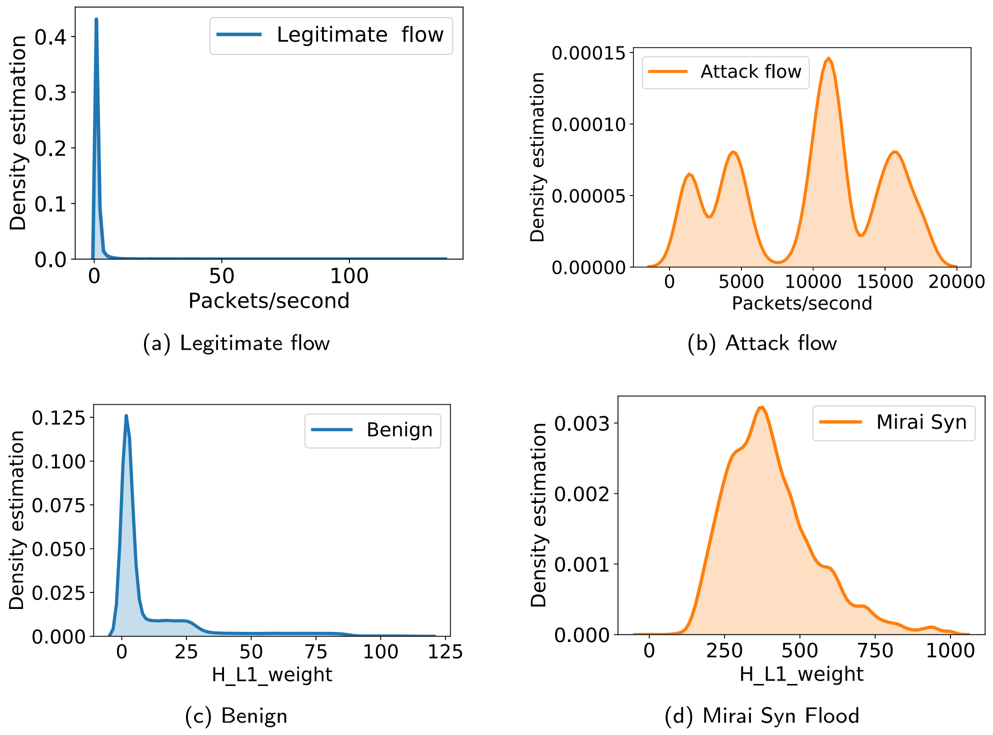


Fig. 12. Characterization of data: Kernel density estimation of legitimate vs. attack instances with testbed (a,b) and benchmark (c,d) datasets.

4.3.2. Dependency analysis on deep layers

This section investigates the performance of the proposed framework based on the number of deep layers. We have used stacked and deep autoencoder to extract features from both testbed and benchmark datasets. Truly, it is noted that an increasing number of hidden layers in the autoencoder excels the performance of the framework. However, finding the optimal depth of hidden layers for a specific domain and addressing the data imbalance problem in parallel is still tricky. Fig. 13 provides the empirical investigation on the number of deep layers (3 for the testbed and 4 for the benchmark datasets) to acquire the best performance of the model. Our model recommends that deep neural networks improve the model performance significantly, but the extremely deeper number of layers may overfit the model.

4.3.3. Dependency analysis on σ

The importance of dependency analysis on σ lies on smoothing the parameter of PNN. The σ is the spread of the Gaussian function or the width parameter which can take σ value between 0 and 1 [35]. The proposed method is evaluated on the dependency of σ for achieving a high detection rate. Notably, the lower values of σ provide a lower false-negative rate and higher values of σ yield a lower false-positive rate. However, we heuristically identify the values of σ as 0.7–0.8 for testbed data and 0.1–0.2 for benchmark data, respectively, when considering imbalanced datasets. Fig. 14 illustrates the per-

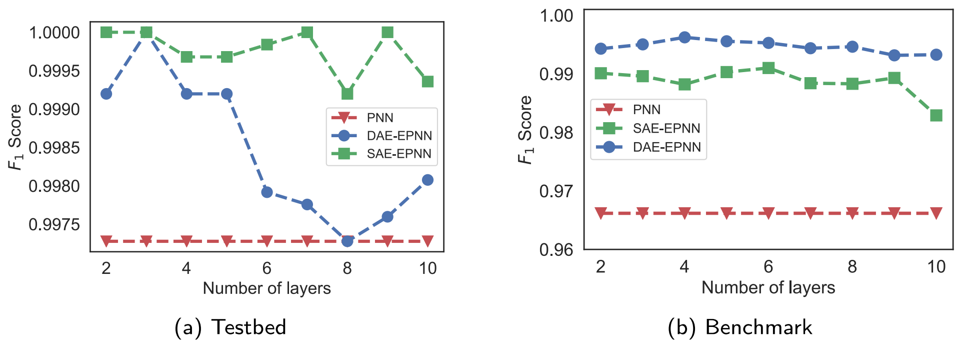


Fig. 13. Performance variation of F_1 score with respect to the number of deep layers.

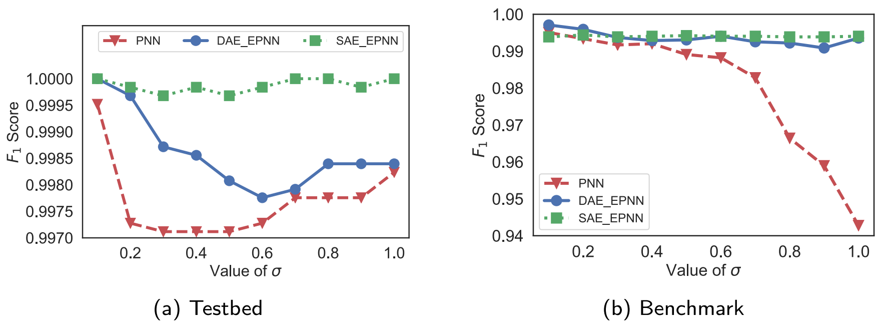


Fig. 14. Performance variation of F_1 score σ of PNN.

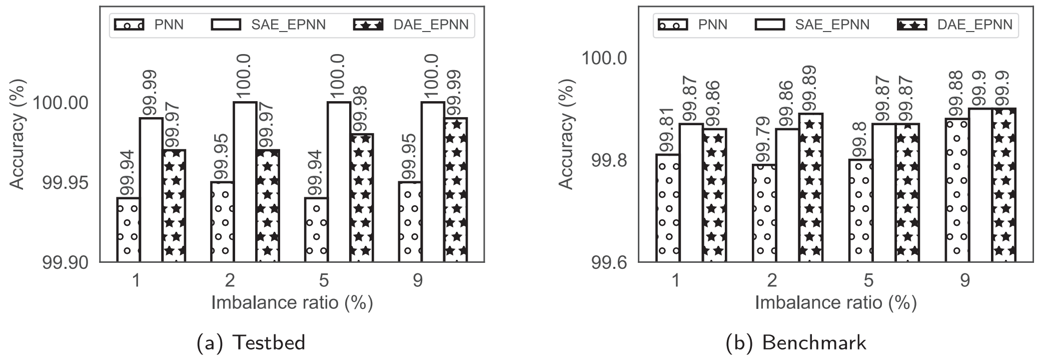


Fig. 15. Accuracy variation for single attacks imbalanced datasets.

formance in the testbed and benchmark datasets, respectively. As shown in Fig. 14, in the benchmark dataset our proposed method's detection rate is above 99%, but single PNN's detection rate decreases from 99% to 94%. Hence, we observe that our proposed method's detection rate is more stable than a single PNN model and a single PNN detection rate decreases when the sigma value increases.

4.3.4. Performance on single attacks

To validate the efficiency of the proposed method on the single attack scenarios, we have used balanced and imbalanced N-BaIoT datasets. First of all, Table 3 shows the accuracy of the proposed method in the balanced N-BaIoT [18] datasets. Second, these sets of experiments address the data imbalance problem by considering a single attack and legitimate datasets. Hence, we created from the N-BaIoT [18] dataset's Provision-PT-838 Security Camera 5 kinds of imbalanced datasets shown in Fig. 8. From Tables 3–4 and Figs. 15–17, we observe that the integration of deep and stacked autoencoders with the deep ensemble of PNNs allowed the improved performance of the model while addressing the data imbalance issues and detecting attacks in IoT. As shown in Fig. 15, there are slight differences in accuracy performance. However, it is not true. In anomaly detection scenarios with the imbalanced dataset, accuracy is not the representative best metric to evaluate the performance. Since a large portion of training data is legitimate traffic, the algorithms are biased toward estimating all the data as legitimate and ignoring the small portion of the attack instances [22]. Hence, MCC and F_1 score measures can

Table 3
The detection accuracy for single attack scenarios when running with the balanced N-BaIoT dataset .

No	IoT devices	Bashlite	PNN	DAE-EPNN	SAE-EPNN	Mirai	PNN	DAE-EPNN	SAE-EPNN
1	SimpleHome-XCS7-1003-WHT Security Camera	Scan TCP Junk UDP Combo	0.9943 0.9985 0.9990 0.9984 0.9979	0.9993 **0.9986** 0.9997 **0.9986** **0.9998**	**0.9994** **0.9986** **0.9998** 0.9985 **0.9998**	Scan Syn UDPplain Ack UDP	0.9976 **0.9999** 0.9996 0.9997 0.9992	**0.9997** **0.9999** 0.9998 **0.9999** 0.9999	0.9996 **0.9999** **1.0000** **0.9999** **1.0000**
2	Ecobee-Thermostat	Scan TCP Junk UDP Combo	0.9947 0.9992 0.9987 0.9990 0.9984	**0.9988** 0.9992 **0.9996** **0.9991** **0.9996**	0.9933 **0.9988** 0.9975 0.9986 0.9823	Scan Syn UDPplain Ack UDP	0.9989 0.9999 0.9970 0.9992 0.9983	**0.9993** **1.0000** **1.0000** 0.9999 **0.9999**	0.9950 0.9998 0.9967 **1.0000** 0.8034
3	Ennio-Doorbell	Scan TCP Junk UDP Combo	0.9880 0.9986 0.9992 0.9985 0.9967	0.9974 0.9986 **0.9999** 0.9986 0.9998	**0.9998** **0.9996** 0.9998 **0.9993** **1.0000**	No Mirai Attacks	-	-	-
4	Provision-PT-838 Security Camera	Scan TCP Junk UDP Combo	0.9973 0.9987 0.9996 0.9988 0.9988	0.9974 **0.9988** **0.9999** 0.9990 **0.9999**	**0.9998** 0.9990 **0.9999** **0.9991** **0.9999**	Scan Syn UDPplain Ack UDP	0.9988 0.9969 0.9999 0.9997 **0.9999**	0.9998 0.9993 **1.0000** 0.9997 **0.9999**	**1.0000** **1.0000** **1.0000** **0.9999** **0.9999**
5	Danmini Doorbell	Scan TCP Junk UDP Combo	0.9937 0.9987 0.9960 0.9987 0.9948	0.9994 0.9992 0.9993 0.9990 0.9998	**0.9995** **0.9996** **0.9997** **0.9993** **0.9999**	Scan Syn UDPplain Ack UDP	0.9993 0.9998**0.99991.0000** **1.0000**	**0.9999** **1.00000.9999** 0.9999 **1.0000**	0.9992 0.9425**0.9999** 0.9803 0.9996
6	Samsung-SNH-1011-N Webcam	Scan TCP Junk UDP Combo	0.9969 0.9987 0.9994 0.9986 0.9992	0.9989 0.9992 **0.9998** 0.9989 **0.9999**	**0.9995** **0.9993** **0.9998** **0.9994** **0.9999**	No Mirai Attacks	-	-	-
7	SimpleHome-XCS7-1002-WHT Security Camera	Scan TCP Junk UDP Combo	0.9986 0.9959 0.9999 0.9986 0.9990	**0.9974** **0.9987** **0.9996** **0.9988** **0.9998**	0.9994 0.9986 0.9998 0.9985 0.9998	Scan Syn UDPplain Ack UDP	0.9986 0.9959 0.9999 0.9999 0.9999	0.9995 **0.9999** **1.0000** **1.0000** 0.9999	**0.9997** **0.9999** **1.0000** 0.9999 **1.0000**
8	Provision-PT-737E Security Camera	Scan TCP Junk UDP Combo	0.9965 0.9978 0.9967 0.9977 0.9987	**0.9993** 0.9989 **0.9998** **0.9988** 0.9999	0.9991 **0.9990** **0.9998** **0.9988** **1.0000**	Scan Syn UDPplain Ack UDP	0.9968 0.9850 **0.9999** **0.9999** **1.0000**	**1.0000** 0.9994 **0.9999** **0.9999** 0.9999	**1.0000** **0.9999** **0.9999** **0.9999** **1.0000**
9	Philips-B120N10 Baby Monitor	Scan TCP Junk UDP Combo	0.9971 0.9993 0.9995 0.9993 0.9985	0.9992 **0.9995** **0.9999** 0.9994 **0.9999**	**1.0000** 0.9994 **0.9999** **0.9995** **0.9999**	Scan Syn UDPplain Ack UDP	0.9995 0.9999 0.9998 0.9999 0.9995	**1.0000** **1.0000** **1.0000** **1.0000** **1.0000**	**1.0000** **1.0000** **1.0000** **1.0000** **1.0000**

Table 4
The detection accuracy for single attack scenarios when running with the imbalanced N-BaIoT dataset .

No	IoT devices	Benign	Attacks	PNN	DAE-EPNN	SAE-EPNN
1	SimpleHome-XCS7-1003-WHT Security Camera	19528	1950	0.9933	**0.9959**	0.9944
2	Ecobee-Thermostat	13113	1310	0.9953	**0.9979**	0.9967
3	Ennio-Doorbell	39100	3910	0.9899	**0.9960**	0.9958
4	Provision-PT-838 Security Camera	98514	9850	0.9901	**0.9987**	0.9983
5	Danmini Doorbell	49548	4950	0.9962	0.9970	**0.9971**
6	Samsung-SNH-1011-N Webcam	52150	5215	0.9976	**0.9988**	0.9985
7	SimpleHome-XCS7-1002-WHT Security Camera	46585	4650	0.9918	**0.9951**	0.9944
8	Provision-PT-737E Security Camera	62154	6210	0.9864	**0.9973**	0.9966
9	Philips-B120N10 Baby Monitor	175240	17520	0.9925	0.9960	**0.9988**

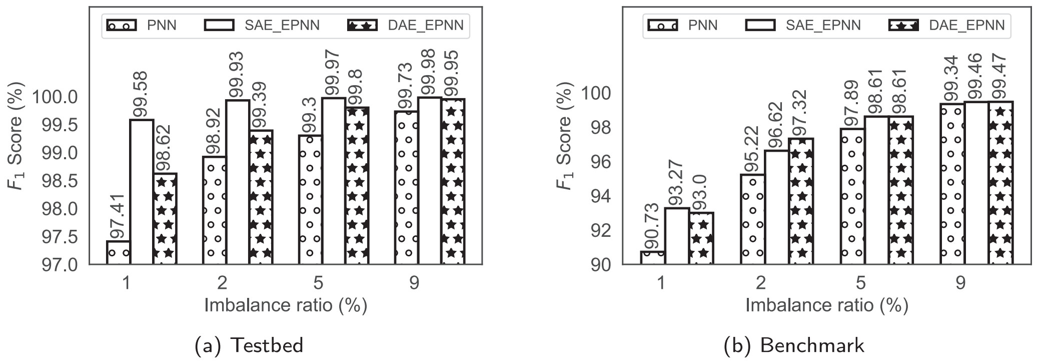


Fig. 16. F_1 score comparison for single attacks imbalanced datasets.

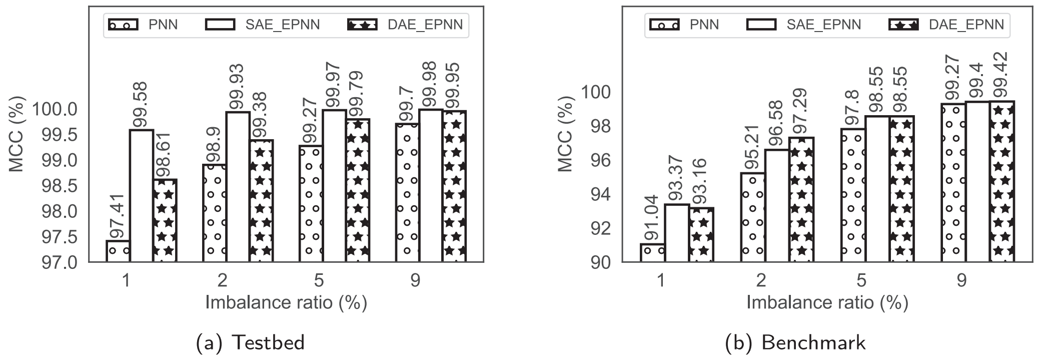


Fig. 17. MCC score variations for single attacks imbalanced datasets.

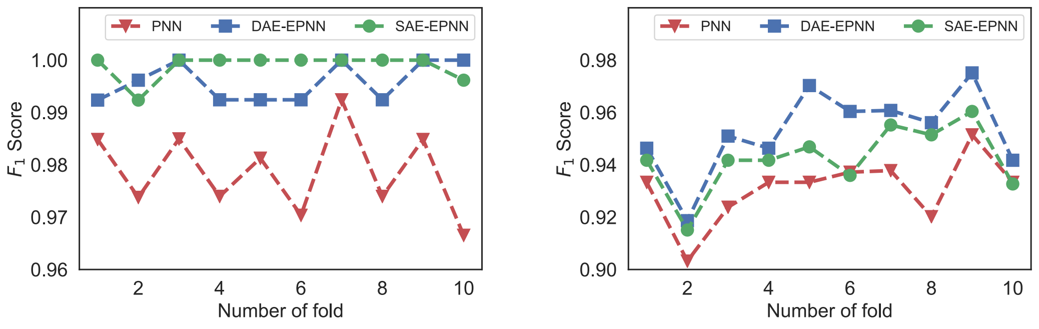
evaluate the performance better in spite of having imbalanced datasets. In addition, the class imbalance problem introduces a bias in the machine learning models that degrades performance. For instance, from Figs. 16–17, we observe that the machine learning models' MCC and F_1 score descrease when the dataset's imbalanced ratio decrease.

As shown in Figs. 16–17, we can also observe that in the 1%, 2%, and 5% imbalanced datasets, our proposed method's MCC and F_1 performances are 1–3% better than a single model.

Finally, since 1% is our experiment's lowest imbalanced dataset, ten-fold cross-validation was performed on the 1% imbalanced testbed and benchmark datasets using a single model and the proposed methods. Fig. 18 shows the improved performance results of our proposed framework after ten-fold cross-validation. The k-fold cross validation is sensitive to poor partition of data. As a result, the accuracy of the baseline model varies due to poor partition of highly imbalanced data.

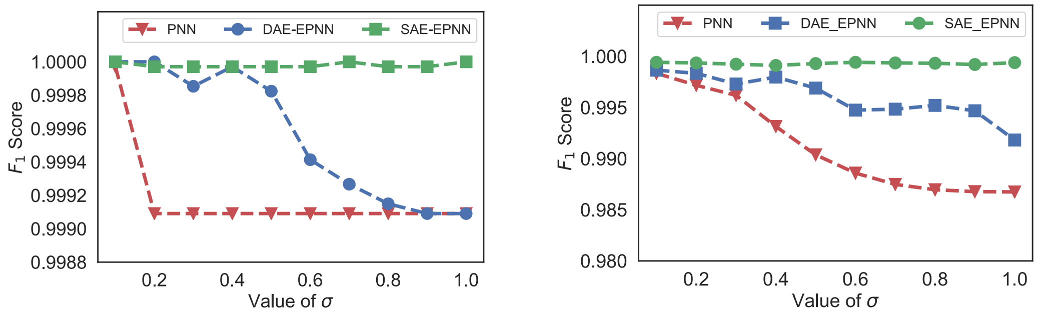
4.3.5. Performance on multiple attacks

The proposed framework is evaluated further for multiple attacks using testbed and benchmark datasets. In the testbed data, we consider multiscale attacks (i.e., DoS, DDoS) with data imbalance scenarios. As shown in Fig. 9, we have found multiple attacks for our experiment from the benchmark datasets. Figs. 19–20 illustrate the improved performance of the proposed framework while detecting multiple attacks in IoT.



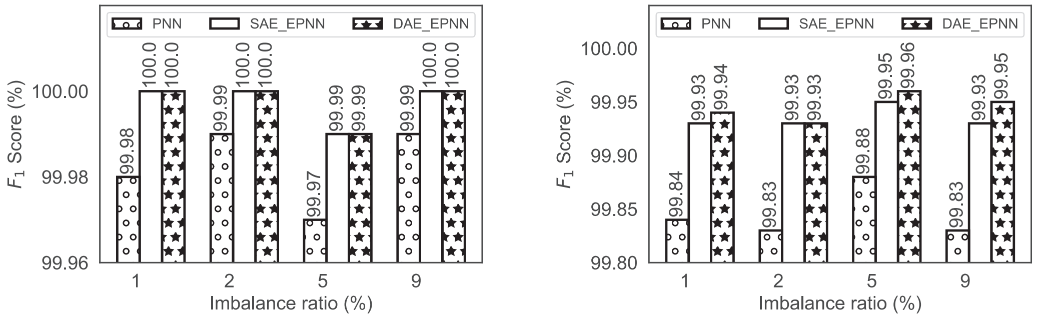
(a) Testbed (b) Benchmark

Fig. 18. F_1 score of models after 10 fold cross validation for detecting single attacks.



(a) Testbed (b) Benchmark

Fig. 19. Multiple attacks performance variation of F_1 score σ of PNN.



(a) Testbed (b) Benchmark

Fig. 20. Multiple attacks F_1 score for imbalanced datasets.

4.3.6. *Performance on dynamic flow management*

To observe the performance of dynamic flow management, we consider two different cases with and without attacks in the testbed setup environment. Fig. 21 illustrates the flow management by examining the throughput in the absence and presence of attacks (low-rate and high-rate), respectively. Fig. 22 shows the dynamic management of flows that pass through the software-enabled switch. From the Figs. 21–22, we observe that the traffic flow are managed well in both period with and without attacks.

4.3.7. *Performance on device status forecasting*

The status table is obtained from the flow management module based on the status of each IoT device in the testbed setup. Table 5 is an example status table obtained by examining the time period 13:49 to 13:53 within testbed environment. DeL-IoT offers an additional features that can predict both short and long-term anomalies in IoT devices. We exploit the Long Short-Term Memory (LSTM) model for the prediction of anomalies as shown in Fig. 23. In addition, our proposed

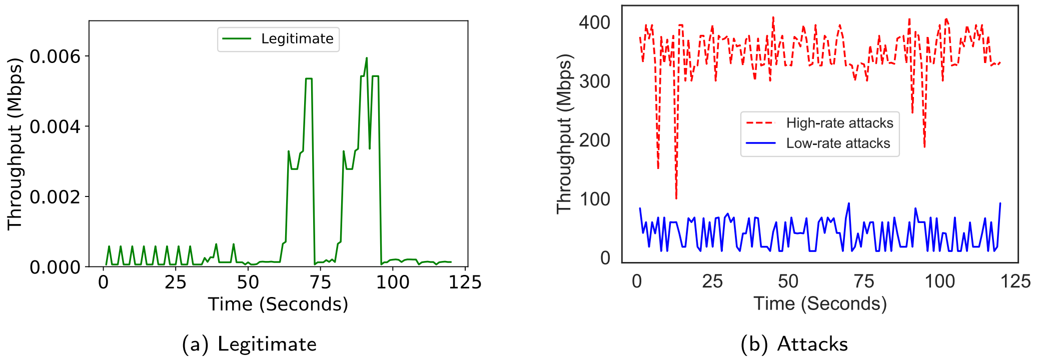


Fig. 21. Throughput - (a) legitimate flows, and (b) attack flows.

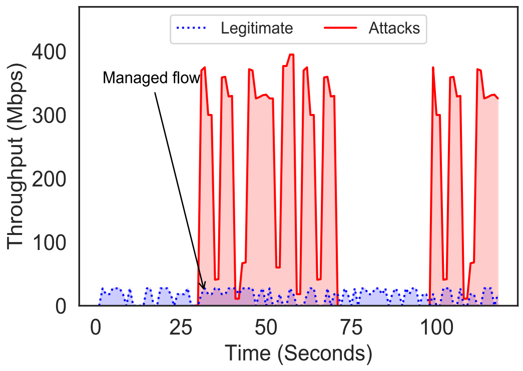


Fig. 22. Throughput - managing flows dynamically.

Table 5
An example of status table .

Device ID	Time	Status
10:0e:7e:c9:cb:f0	13:49	0
10:0e:7e:c9:cb:f0	13:50	0
10:0e:7e:c9:cb:f0	13:51	0
10:0e:7e:c9:cb:f0	13:52	0
98:f2:b3:f3:2a:38	13:53	1

LSTM model achieved an accuracy of 81.48%. This prediction increases the effectiveness to protect the IoT devices against emerging attacks.

In this experiment, we used 84 min of data for training and 36 min of data for testing with an LSTM-based sequential model for predicting the status of IoT devices. Based on empirical evaluation, we found that the model with 64 hidden neurons has lower mean square error (MSE) than the models with 4, 32, 128 hidden neurons. For this reason, we determine that the number of hidden neurons of LSTM is 64. Hence, the network has a visible layer with 1 input (a device status), a hidden layer with 64 LSTM blocks or neurons, and an output layer that makes a single value prediction. The default sigmoid activation function is used for the LSTM blocks and the network is trained for 100 epochs with batch size 1.

4.4. Comparison with competing methods

To assess the efficiency of the proposed framework for detecting anomalies in IoT, we compared it with existing methods including SoftThings [17], network-based IoT anomaly detection [18], and DÏoT [24]. SoftThings [17] can detect and mitigate dynamic attacks in IoT using SDN with 98% precision. It employs linear and non-linear Support Vector Machine (SVM) with default hyper-parameters when detecting IoT attacks at the SDN controller in a Mininet simulation environment. The network-based IoT anomaly detection [18] employs the deep autoencoders with four hidden layers to detect anomalies in IoT. The system has optimized hyperparameters such as learning rate, number of epochs, anomaly threshold, and windows size of autoencoders for each IoT device. For instance, in their experiment, the Danmini doorbell, that is one of the IoT devices, is used with the optimized hyperparameters of the autoencoders such as learning rate 0.012, the number of epochs 800, anomaly threshold 0.042, and window size 82. The system is evaluated using testbed datasets which have nine commercial

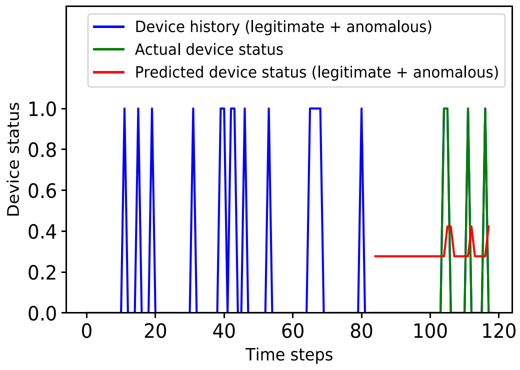


Fig. 23. Forecasting IoT device status to uncover anomalies.

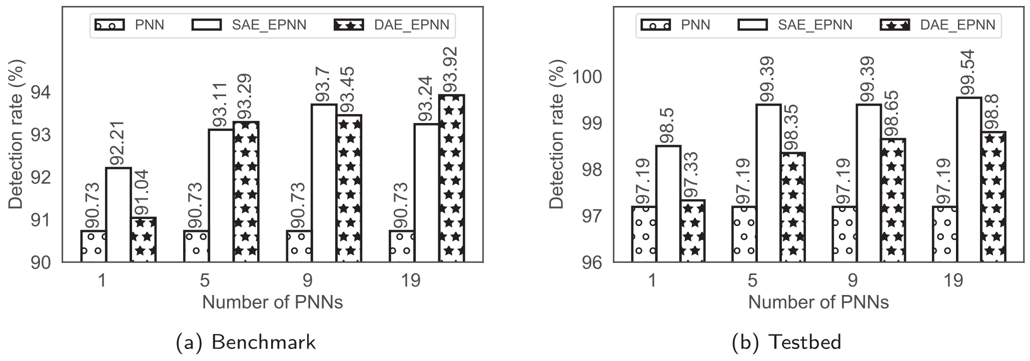


Fig. 24. Performance variation of detection rate with respect to the number of PNNs in the 1% imbalanced dataset .

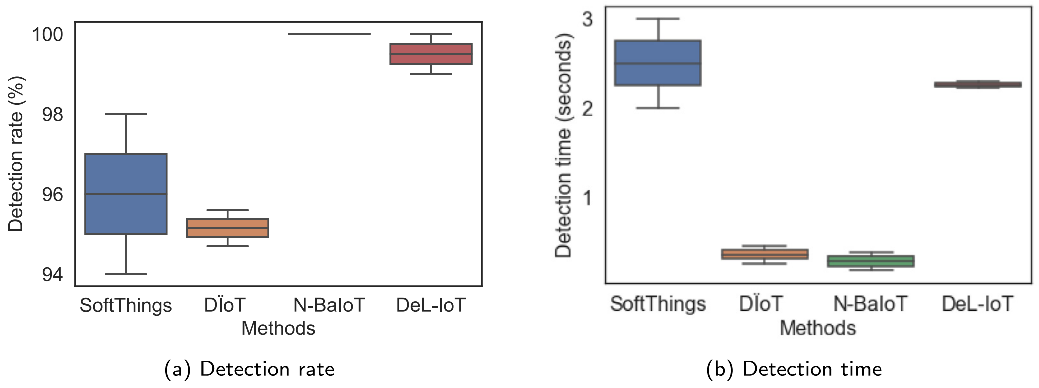


Fig. 25. Comparison of detection rate and time with competing methods.

IoT devices and it achieves a 100% detection rate. The DÏoT [24] reports an autonomous self-learning anomaly detection system for IoT. A GRU network with three hidden layers of size 128 neurons each was used in the system. The system provides evidence to detect anomalies with 95.6% detection rate.

Although the existing methods have positive performance, the majority of existing methods or systems were evaluated either as testbed data or in simulated environments. Additionally, our framework shows its superiority in terms of the following points: (a) detects anomalies with 99.8% detection rate for testbed and 99.9% detection rate for benchmark datasets, (b) addresses the data imbalance problem by using a deep ensemble learning, (c) demonstrates dynamic flow management in the presence of attacks, (d) increases device uptime, and (e) forecasts IoT device status for anomalies in short and long term. To observe the performance of deep ensemble learning when choosing different number of PNNs in each layer, we found numerous distinctions in detection rate using 1% imbalanced benchmark and testbed datasets to uncover anomalies in IoT. The hyperparameters used in our experiment the explained in Section 4.3. Fig. 24 illustrates how the deep ensemble learning of PNNs improves the detection rate to uncover anomalies in IoT. Fig. 25 shows the comparison of DeL-IoT with competing methods.

5. Conclusion and future work

In this work, we present a novel deep ensemble learning model based framework called DeL-IoT for IoT anomaly detection using SDN primarily to detect anomalies or dynamic attacks for increasing detection performance, switch-level dynamic flow management, and forecasting short and long term device status. We suggest the deep and stacked auto-encoders to extract features for stacking into an ensemble of PNNs learning model for performance improvement while addressing the data imbalance problem. Additionally, we propose a novel mechanism for dynamic flow management in presence of attacks and forecasting device status based on the status table. The system manager can utilize this forecasting features for early action against dynamic attacks from the device to physical infrastructures. We have demonstrated the improved performance in the testbed and benchmark datasets with 99.8% and 99.9% detection rates, respectively, better than the existing methods. Our proposed DeL-IoT method results show that in the 1% imbalanced datasets for both single and multi-class anomalies F_1 and *MCC* measures are around 2–3% better than a single model. Finally, we conclude that the use of the deep or stacked autoencoder in combination with ensemble PNN improves the performance of IoT anomaly detection in data imbalance problem and also can forecast device status efficiently.

We are underway to extend this work for detecting and predicting multiscale VSI-DDoS (very short-intermittent) attacks in both devices and specially mission-critical applications at the edge.

Declaration of Competing Interest

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

Acknowledgment

Part of this study was funded by the ICS-CoE Core Human Resources Development Program. Additional support was provided by the JST CREST Grant Number JPMJCR1783, Japan and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, Sweden.

References

[1] Global Digital Transformation Research Team at Frost & Sullivan, IoT security market watch-key market needs and solution providers in the IoT landscape (2017) 4.
[2] A.H.M. Aman, E. Yadegaridehkordi, Z.S. Attarbashi, R. Hassan, Y.-J. Park, A survey on trend and classification of internet of things reviews, IEEE Access 8 (2020) 111763–111782.
[3] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, E.K. Markakis, A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues, IEEE Commun. Surv. Tutor. 22 (2) (2020) 1191–1221.
[4] I. Farris, T. Taleb, Y. Khettab, J. Song, A survey on emerging and NFV security mechanisms for IoT systems, IEEE Commun. Surv. Tutor. 21 (1) (2018) 812–837.
[5] R. Doshi, N. Apthorpe, N. Feamster, Machine learning DDoS detection for consumer internet of things devices, in: 2018 IEEE Security and Privacy Workshops (SPW), IEEE, San Francisco, CA, USA, 2018, pp. 29–35.
[6] V. Sharma, J. Kim, S. Kwon, I. You, K. Lee, K. Yim, A framework for mitigating zero-day attacks in IoT, in: Information Security and Cryptography (CISC-SÍ7), 2018, pp. 1–6. Sinchang-Asan, South Korea
[7] C. Krügel, T. Toth, E. Kirda, Service specific anomaly detection for network intrusion detection, in: Proceedings of the 2002 ACM Symposium on Applied Computing, in: SAC '02, 2002, pp. 201–208. New York, NY, USA
[8] A. Ghosh, Big data and its utility, Consult. Ahead 10 (1) (2016) 52–68.
[9] D. Chakraborty, V. Narayanan, A. Ghosh, Integration of deep feature extraction an ensemble learning for outlier detection, Pattern Recognit. 89 (2019) 161–171.
[10] M. He, A.M. Alba, A. Basta, A. Blenk, W. Kellerer, Flexibility in softwarized networks: classifications and research challenges, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2600–2636.
[11] N. An, H. Ding, J. Yang, R. Au, T.F.A. Ang, Deep ensemble learning for Alzheimer's disease classification, J. Biomed. Inform. 105 (2020) 103411.
[12] A. Al-Abassi, H. Karimipour, A. Dehghantanha, R.M. Parizi, An ensemble deep learning-based cyber-attack detection in industrial control system, IEEE Access 8 (2020) 83965–83973.
[13] F. Jia, S. Li, H. Zuo, J. Shen, Deep neural network ensemble for the intelligent fault diagnosis of machines under imbalanced data, IEEE Access 8 (2020) 120974–120982.
[14] O. Sagi, L. Rokach, Ensemble learning: a survey, WIREs Data Min. Knowl. Discov. 8 (4) (2018) e1249, doi:10.1002/widm.1249.
[15] E. Tsogbaatar, M.H. Bhuyan, Y. Taenaka, D. Fall, K. Gonchigsumlaa, E. Elmroth, Y. Kadobayashi, SDN-enabled IoT anomaly detection using ensemble learning, in: Proceedings of the 16th International Conference on Artificial Intelligence Applications and Innovations (AIAI), 2020, pp. 268–280. Halkidiki, Greece
[16] M. Hasan, M.M. Islam, M.I.I. Zarif, M. Hashem, Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches, Internet Things; Eng. Cyber Phys. Hum. Syst. 7 (2019).
[17] S.S. Bhunia, M. Gurusamy, Dynamic attack detection and mitigation in IoT using SDN, in: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), 2017, pp. 1–6. Melbourne, VIC, Australia
[18] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-BaIot:network-based detection of IoT botnet attacks using deep autoencoders, IEEE Pervasive Comput. 17 (3) (2018) 12–22.
[19] A. Hamza, H.H. Gharakheili, T.A. Benson, V. Sivaraman, Detecting volumetric attacks on IoT devices via SDN-based monitoring of MUD activity, in: Proceedings of the 2019 ACM Symposium on SDN Research, in: SOSR '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 36–48.

[20] V. TimÄenko, S. Gajin, Machine learning based network anomaly detection for IoT environments, in: 8th International Conference on Information Society and Technology (ICIST 2018), 2018, pp. 196–201. Belgrade, Serbia
[21] H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang, Z. Han, Capsule network assisted IoT traffic classification mechanism for smart cities, IEEE Internet Things J. 6 (5) (2019) 7515–7525.
[22] M. Zolanvari, M.A. Teixeira, R. Jain, Effect of imbalanced datasets on security of industrial iot using machine learning, in: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), 2018, pp. 112–117. Miami, FL, USA
[23] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: bot-Iot dataset, Fut. Gener. Comput. Syst. 100 (2019) 779–796.
[24] T.D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, A.-R. Sadeghi, DIoT: a federated self-learning anomaly detection system for IoT, in: IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019, pp. 756–767. Dallas, Texas, USA
[25] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, V.-L. Nguyen, An unsupervised deep learning model for early network traffic anomaly detection, IEEE Access 8 (2020). 1–1
[26] M.-O. Pahl, F.-X. Aubet, DS2OS traffic traces, 2019.
[27] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74.
[28] O.N. Foundation, OpenFlow Switch Specification, Report, Open Networking Foundation, 2015.
[29] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, pp. 493–495.
[30] D.S. Berman, A.L. Buczak, J.S. Chavis, C.L. Corbett, A survey of deep learning methods for cyber security, Information 10 (2019).
[31] S. Kaur, J. Singh, N.S. Ghumman, Network programmability using POX controller, in: ICCCS International Conference on Communication, Computing & Systems, IEEE, 138, 2014, pp. 134–138. Punjab, India
[32] A. Botta, A. Dainotti, A. Pescapè, A tool for the generation of realistic network workload for emerging networking scenarios, Comput. Netw. 56 (15) (2012) 3531–3547.
[33] M.H. Bhuyan, E. Elmroth, Multi-scale low-rate DDoS attack detection using the generalized total variation metric, in: 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018, pp. 1040–1047. Orlando, Florida, USA
[34] M.H. Bhuyan, D.K. Bhtaacharyya, J.K. Kalita, Network anomaly detection: methods, systems and tools, IEEE Commun. Surv. Tutor. 16 (1) (2014) 303–336.
[35] A. Mehran, A. Hojjat, Enhanced probabilistic neural network with local decision circles: a robust classifier, Integr. Comput.-Aided Eng. 17 (3) (2010) 197–210.