SDN-Enabled IoT Anomaly Detection Using Ensemble Learning

Enkhtur Tsogbaatar[1(✉)], Monowar H. Bhuyan[1,2], Yuzo Taenaka[1], Doudou Fall[1], Khishigjargal Gonchigsumlaa[3], Erik Elmroth[2], and Youki Kadobayashi[1]

[1] Laboratory for Cyber Resilience, NAIST, Nara 630 0192, Japan
{tsogbaatar.enkhtur.ta4,yuzo,doudou-f,youki-k}@is.naist.jp
[2] Computing Science Department, Umeå University, Umeå, Sweden
{monowar,elmroth}@cs.umu.se
[3] SICT, MUST, 133 43 Ulaanbaatar, Mongolia
khishigjargal@must.edu.mn

Abstract. Internet of Things (IoT) devices are inherently vulnerable due to insecure design, implementation, and configuration. Aggressive behavior change, due to increased attacker's sophistication, and the heterogeneity of the data in IoT have proven that securing IoT devices is a making challenge. To detect intensive attacks and increase device uptime, we propose a novel ensemble learning model for IoT anomaly detection using software-defined networks (SDN). We use a deep auto-encoder to extract handy features for stacking into an ensemble learning model. The learned model is deployed in the SDN controller to detect anomalies or dynamic attacks in IoT by addressing the class imbalance problem. We validate the model with real-time testbed and benchmark datasets. The initial results show that our model has a better and more reliable performance than the competing models showcased in the relevant related work.

Keywords: Internet of Things (IoT) · Anomaly detection · Autoencoder · Probabilistic Neural Networks (PNN) · Software defined network (SDN) · Ensemble learning

1 Introduction

The rapid evolution of the Internet of Things 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 [7] have predicted that the number of connected devices will increase by 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. As 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

Published by Springer Nature Switzerland AG 2020
I. Maglogiannis et al. (Eds.): AIAI 2020, IFIP AICT 584, pp. 268–280, 2020.
https://doi.org/10.1007/978-3-030-49186-4_23

evade, manipulate, and take over IoT networks. 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 [5]. 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.

SDN [10] attracts both academia and industry due to appealing features such as flexibility, dynamicity in network operations and resource management. By leveraging SDN, we can provide several attractive benefits for IoT security. Unfortunately, managing heterogeneous IoT networks and detecting dynamic attacks become challenging due to changing behaviour, constraint resources, and limited scalability. Hence, the ensemble learning model is developing a detection module within the SDN framework by: (i) isolating compromised devices, (ii) doing early detection and mitigation, (iii) reducing resource wastage, (iv) improving accuracy by deploying sophisticated algorithms. Mostly, detection models suffer class imbalance problems that incur significant performance loss for detecting attacks in IoT. This work takes the benefits of SDN controller at switches by grabbing the features of data back-and-forth into IoT devices.

To address these challenges, we present an SDN-enabled ensemble learning model for IoT anomaly detection that utilizes the appealing features of deep auto-encoders and probabilistic neural networks. Our proposal uses both device and network switching level features to build an efficient detection system, which will ensure the increase of IoT device uptime and performance. We make the following contributions.

- A novel SDN-enabled IoT anomaly detection using ensemble learning:
  - It utilizes the principles of deep autoencoders and probabilistic neural networks.
  - It can detect dynamic attacks and handle the class imbalance problem.
- Systematic and extensive experimental analysis using testbed and benchmark datasets, showing the proposed method is superior to competitors in terms of $F1$ measure.

2 Related Work

Several significant amounts of works [2,12–14] have been proposed to address the machine learning-driven IoT anomaly detection problems with and without the SDN. Bhunia et al. [2] present a machine learning-driven anomaly detection and mitigation method for IoT using the SDN. This method deploys the SVM model at the SDN controller to monitor and learn the behavior of IoT devices over time and detect attacks. The evaluation was ensured based on Mininet-based emulation by considering multiple attack scenarios and claimed that the method can detect and mitigate attacks within a few seconds. Zolanvari et al. [14] demonstrate the efficiency of Artificial Neural Network (ANN) in detecting anomalies with Industrial IoT (IIoT) testbed datasets having class imbalance. This method

leverages Synthetic Minority Over-Sampling Technique (SMOTE) to address the class imbalance problem in IIoT datasets to achieve expected ANN performance. Recently, Thien et al. [13] propose an autonomous self-learning distributed system for detecting compromised IoT devices using GRU (Gated Recurrent Units) known as DÏoT. This method was evaluated with 33 IoT devices and demonstrates True Positive Rate (TPR) is 95.6% and can detect compromised devices within 257 ms.

However, still, several opportunities are left to address IoT security problems in small and large-scale industries with a comprehensive solution focus either centralized or distributed fashion. Some problems including increase uptime of devices that deployed in the edge of the network, device heterogeneity, reliability, and early detection of attacks irrespective of class imbalance issues. We are motivated to address two primary aspects including early detection of dynamic attacks and class imbalance problems using the data-driven mechanism.

3 System Model

We present a novel ensemble learning-based anomaly detection model for IoT using SDN primarily to detect anomalies or dynamic attacks for increasing device uptime and detection efficiency. This model aims to provide security of IoT devices by monitoring traffic and system metrics in SDN switches. Also, it can handle the class imbalance problem where attack classes are rarer than legitimate classes. An SDN-enabled IoT anomaly detection framework is given in Fig. 1. This framework has three primary components, including an SDN controller, 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. 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 [11]. Each packet is

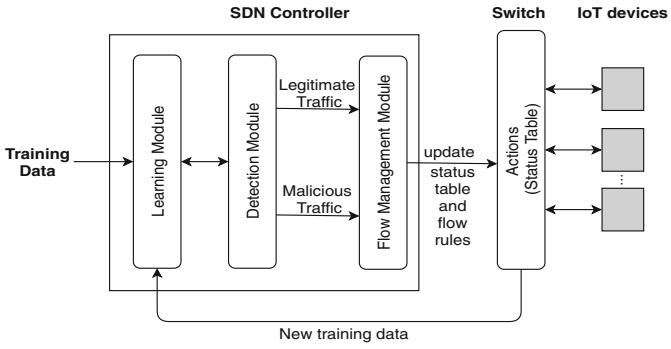


Fig. 1. SDN-enabled framework for IoT anomaly detection.

forwarded based on the enabled rules in the flow table. Each rule have 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 [6]. 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.

3.1 Anomaly Detection

The proposed model aims to detect 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.

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 using both testbed and benchmark datasets. For testbed data, we set up a real-time 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. For the benchmark, we use a recent dataset, called N-BaIoT [12] for our experimentation. We provide extended explanations of each dataset in Sect. 4.

To make an efficient and better 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 $X = \{x_1, x_2, \cdots, x_n\}$ is the input data, $X' = \{x'_1, x'_2, \cdots, x'_n\}$ is the encoded output, $F = \{f_1, f_2, \cdots, f_n\}$ is the features set, and h is the 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, $h = f(x)$ and a decoder $r = g(h)$ [9]. We employ 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. More hidden layers incur better feature representation, which is advantageous for an anomaly detection model [1]. The under-complete autoencoders have a lower number of nodes in hidden layers.

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. The PNN is represented as a Kernel Discriminant Analysis (KDA), which is a generalization of Linear Discriminant Analysis (LDA) to find a linear combination of features that separates classes. A PNN consists of several sub-networks that estimate the Parzen window probability density function of each class using the samples of the training set. Each node of the network calculates the probability density function for each training sample according to the Eq. (1).

$$p(x) = \frac{1}{\sigma}\omega(\frac{x - x_i}{\sigma}) \tag{1}$$

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 up for the next layer to get the class-wise probability. In summation layer, the l^{th} nodes aggregate the values from the pattern layer of l^{th} classes. This summation is estimated based on mixed Gaussian or Parzen window estimator as defined in Eq. (2).

$$f_l(x) = \frac{1}{n\sigma}\omega \sum_{n=1}^{n_l}(\frac{x - x_i}{\sigma}). \tag{2}$$

where n_l is the number of sample in l^{th} classes. Hence, the summation layer maps the l^{th} nodes to the l^{th} classes. For new samples, $f_l(x)$ 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 $f_l(x)$ 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 insensitive to an outlier in the data, new input patterns stores in the 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 observed that PNN alone cannot provide better results. Hence, we have used deep autoencoder probabilistic neural networks to detect anomalies in IoT.

Deep autoencoder probabilistic neural networks (DAE-PNN) is an autoencoder integrated with probabilistic neural networks. It encodes input 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 autoencoder 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 ensemble learning. In PNN, the smoothing parameter σ determines the spread of RBF where 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 model to over-fit whereas an extremely high value may cause model to under-fit. However, both factors are essential to address the class imbalance problem during anomaly detection in IoT. These problems motivate us to develop an ensemble learning model, which we explain below.

Ensemble Learning employs multiple PNNs 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 $A = \{x_{a1}, x_{a2}, \cdots, x_{an}\}$ is the set of anomalous instances, $L = \{x_{l1}, x_{l2}, \cdots, x_{ln}\}$ is the legitimate instances, S is the training dataset, Y is the test dataset, tr and te indicate training and testing instances, then we can get S and Y as in Eq. (3):

$$\begin{aligned} S &= A_{tr} \cup L_{tr} \\ Y &= A_{te} \cup L_{te} \end{aligned} \tag{3}$$

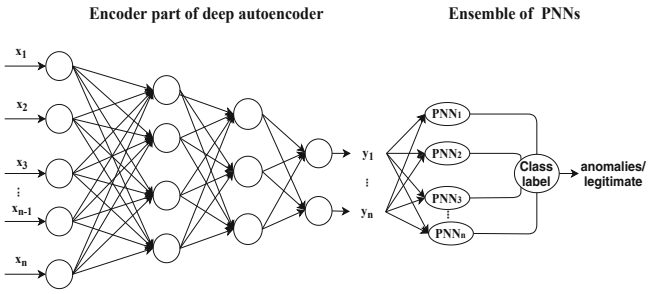


Fig. 2. The proposed model: an integration of deep autoencoder and ensemble of PNNs.

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 for 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 autoencoder, since we know that autoencoder provides the non-linear transformation of input data to reduce features set. A pictorial representation of the proposed model is given in Fig. 2. Let f be a non-linear activation function with weight Ω and bias b then the deep autoencoder is formulated as in Eq. (4):

$$\begin{aligned} \xi &= \Big(f(\Omega x + b) \Big) \\ DAE(x) &= \Big(\xi_1(\xi_2(\xi_3(\cdots \xi_h(x)))) \Big) \end{aligned} \tag{4}$$

where $\xi()$ is the encoding function whereas $\xi_i()$ 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. (5).

$$z_c^n(\hat{x}) = f_c^n \Big(DAE(x) \Big) \tag{5}$$

Each input instance x is assigned to C classes based on intermediate RBF score received from the n^{th} PNNs. However, if there are N PNNs for the ensemble, then each instance, x is assigned to C classes based on the Eq. (6). The classification score is computed using Eq. (7) for each instance x that belongs to a specific class.

$$z_c(x) = \sum_{n=1}^{N} (z_c^n(\hat{x})) \tag{6}$$

$$s_c(x) = \frac{z_c(x) - min(z_c(x))}{max(z_c(x)) - min(z_c(x))} \tag{7}$$

Once we get the classification score for each instance, we label the unknown instance x as anomalous or legitimate based on the node's maximum probability, $max(s_c(x))$.

3.2 Flow Management

This module is enabled to prepare appropriate rules and to dynamically update the switch as a set of actions. If the network flow is marked as legitimate, it can then pass the flow across the controller without any interruption. Otherwise, the controller blacklists the flow and investigates further to verify the types of attacks.

3.3 Maintenance of Status Table

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. The status of each IoT device is marked either as 0 (legitimate) or as 1 (anomalies) for each time point. The system manager utilizes this feature to easily handle large-scale IoT devices as well as their services to the end-users.

4 Performance Evaluation

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

4.1 Datasets

The proposed method is evaluated using two datasets: (i) real-time testbed data, and (ii) benchmark data. The real-time testbed data is generated with a significant amount of attacks on IoT devices and applications.

Real-Time Testbed Data: The experiment is performed in a virtualized environment with a hierarchical deployment of IoT devices to physical servers in the real-time testbed. Figure 3 illustrates the architecture of the real-time testbed setup. The testbed is comprised of multiple servers and applications at a physical level, virtualized level, IoT devices and IoT applications. We consider multiple VMs, one of the VMs is a target of attackers. We generate both DoS and DDoS attacks using the Targa2[1] attack generation tool, the D-ITG internet traffic generator tool [4], the BoNeSi botnet simulator[2], and the stress-ng[3] system resources load generator. We generate multiscale attacks [3] to the IoT devices and applications when they are deployed in the virtualized environment. We collect multiple metrics (e.g., packet, flow, system metrics, device uptime, etc.) from devices to physical infrastructures for learning the proposed ensemble model. POX[4] 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.

Benchmark Data: Due to the non-availability of benchmark datasets, we used the N-BaIoT [12] dataset for our experiments. This dataset was prepared using two attack tools, i.e., Mirai (scan, ACK flooding, SYN flooding, UDP flooding, UDPplain attacks) and Bashlite (scan, junk, UDP flooding, TCP flooding,

[1] http://packetstormsecurity.com/.
[2] https://github.com/Markus-Go/bonesi.
[3] https://kernel.ubuntu.com/~cking/stress-ng/.
[4] https://github.com/noxrepo/pox.

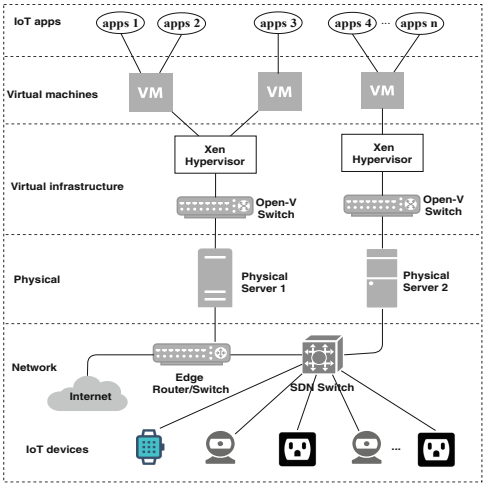


Fig. 3. Real-time testbed setup.

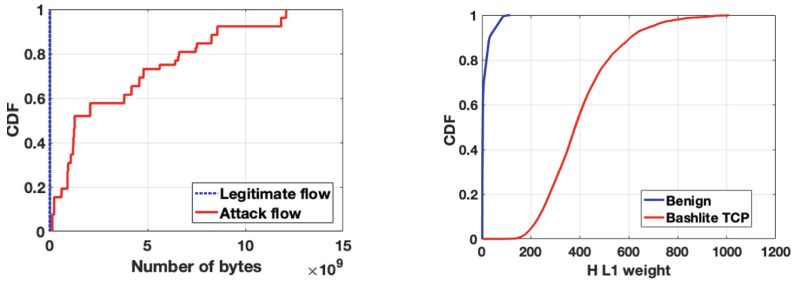
COMBO attacks), with 9 commercial IoT devices. There are 5 Bashlite, 5 Mirai, and 1 legitimate datasets, having 115 features in each of them. We created an imbalanced dataset by separating 98514 legitimate and 9850 anomalous instances with a combination of 10 different kinds of Mirai and Bashlite attacks.

4.2 Results

The proposed framework is evaluated using both real-time testbed and benchmark datasets. We present the results based on real-time testbed datasets that consider multi-level metrics such as packet, flow, and system metrics. To observe the behavior of both real-time testbed and benchmark datasets, we estimate the cumulative density for legitimate and attack instances, as shown in Fig. 4. From the Fig. 4, it is clear that the distribution of legitimate instances differs from attack instances.

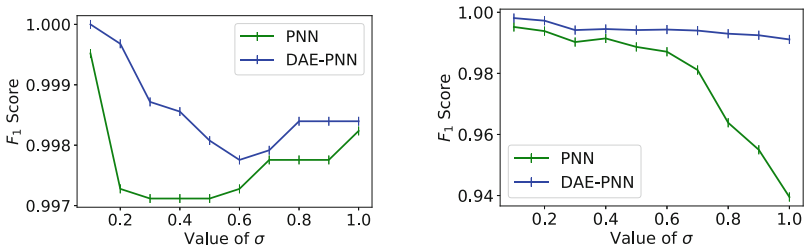
In this experiment, we used deep autoencoder with an ensemble of PNNs with the 'adadelta' optimizer and 'binary-crossentropy' as a loss function. The model was pre-trained with 100 epoch, 100 batch size, and 'relu' activation function by considering four hidden layers with the compositions 17-16-16-15 for testbed data and 115-100-50-25 for benchmark data. These compositions indicate 17 features at the first layer and finish with 15 the encoded outcome from the deep autoencoder. Regarding the testbed and benchmark datasets, we have used non-overlapping 70% for training and 30% for testing, so that the class imbalance problem remains the same in both the training and the test set.

Dependency Analysis on σ: The proposed framework analyzes the dependency of σ to identify suitable values that illustrate the maximum detection



(a) Testbed data **(b)** Benchmark data

Fig. 4. Characterization of data: cumulative density for legitimate vs. attack instances.

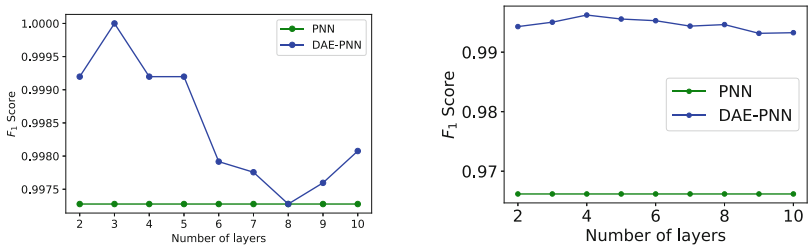


(a) Testbed data **(b)** Benchmark data

Fig. 5. Performance variation of F_1 score with respect to σ of PNN.

rate. The performance of PNN depends on the suitable values of σ. Notably, the lower values of σ provide a lower false-negative rate and higher values of σ yields a lower false-positive rate. We have heuristically identified the range of values for σ as 0.1–0.3 for real-time data and 0.1–0.3 for benchmark data, respectively. These experiments were conducted by considering imbalanced datasets. Figure 5 illustrates the performance in the real-time testbed and benchmark datasets.

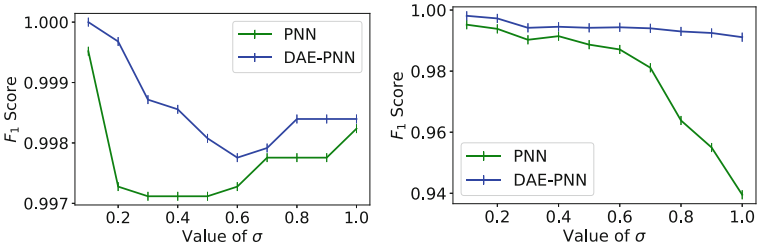
Dependency Analysis on Deep Layers: We used deep autoencoder to extract features from both real-time testbed and benchmark datasets. As we know, an increasing number of hidden layers in the autoencoder excels the performance of the model. However, finding the optimal depth of hidden layers for a specific domain and address the class imbalance problem in parallel is still tricky. Figure 6 provides the empirical investigation on the number of deep layers (3 for real-time testbed and 4 for benchmark datasets) to acquire the best performance of the model. Our model recommends that deep neural networks improve the model performance significantly, but a vast number of layers may overfit the model.



(a) Testbed **(b)** Benchmark

Fig. 6. Performance variation of F_1 score with respect to number of deep layers.

Performance on Attack Analysis in IoT: The proposed framework is evaluated with intensive experiments for multiple attack scenarios using real-time testbed and benchmark datasets. In the testbed data, we consider multi-scale attacks (i.e., DoS, DDoS) with class imbalance scenarios. Also, we found multiple attack scenarios from the benchmark dataset. Figures 7a and b illustrate the improved performance of the proposed framework for detecting anomalies in IoT.



(a) Testbed **(b)** Benchmark

Fig. 7. F_1 score variation with respect to σ using ensemble model for detecting anomalies in IoT.

4.3 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 [2], network-based IoT anomaly detection [12], and DÏoT [13]. The SoftThings [2] can detect and mitigate dynamic attacks in IoT using SDN. The system achieves 98% precision. It employs Support Vector Machine (SVM) when detecting IoT attacks at the SDN controller in a mininet simulation environment. The network-based IoT anomaly detection [12] employs the deep autoencoders to detect anomalies

in IoT. The system has been evaluated using testbed datasets and it achieves 100% detection rate. The DÏoT [13] reports an autonomous self-learning anomaly detection system for IoT. 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: (i) we detect anomalies with 99.8% detection rate for testbed and 99.9% detection rate for benchmark datasets, (ii) we address the class imbalance problem by using ensemble learning, (iii) we increase IoT device uptime.

5 Conclusion and Future Work

This work presents a novel ensemble learning model for IoT anomaly detection using SDN while addressing the class imbalance problems. Our model integrates the deep autoencoder to encode the features set and impedes them to an ensemble of PNNs for performance improvement of the baseline learning model. The proposed method is shown to perform better than the existing techniques for binary classification anomalies in IoT by employing the appealing features of SDN. We demonstrated superior performance as compared to existing methods in the real-time testbed and benchmark datasets with 99.8%.

In the future, we would like to extend our work to design and implement the deep ensemble learning model for multi-class anomaly detection in IoT. The deep ensemble learning model will consider the maximum composition of multi-scale attacks in IoT within SDN framework.

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.

References

1. Berman, D.S., Buczak, A.L., Chavis, J.S., Corbett, C.L.: A survey of deep learning methods for cyber security. Information **10**, 122 (2019)
2. Bhunia, S.S., Gurusamy, M.: Dynamic attack detection and mitigation in IoT using SDN. In: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), pp. 1–6. IEEE (2017)
3. Bhuyan, M.H., Elmroth, E.: Multi-scale low-rate DDoS attack detection using the generalized total variation metric. In: 17th IEEE International Conference on Machine Learning and Applications, Orlando, Florida, USA, 17–20 December 2018 pp. 1040–1047. IEEE SMC (2018)
4. Botta, A., Dainotti, A., Pescapè, A.: A tool for the generation of realistic network workload for emerging networking scenarios. Comput. Netw. **56**(15), 3531–3547 (2012)

5. Farris, I., Taleb, T., Khettab, Y., Song, J.: A survey on emerging and NFV security mechanisms for IoT systems. IEEE Commun. Surv. Tutor. **21**(1), 812–837 (2018)
6. Foundation, O.N.: OpenFlow switch specification. Report ONF TS-023, Open Networking Foundation (2015)
7. Frost and Sullivan: IoT Security Market Watch-Key Market Needs and Solution Providers in the IoT Landscape, June 2017. https://store.frost.com/iot-security-market-watch-key-market-needs-and-solution-providers-in-the-iot-landscape.html
8. Ghosh, A.: Big data and its utility. Consult. Ahead **10**(1), 52–68 (2016)
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, pp. 493–495. MIT Press, Cambridge (2017)
10. He, M., Alba, A.M., Basta, A., Blenk, A., Kellerer, W.: Flexibility in softwarized networks: classifications and research challenges. IEEE Commun. Surv. Tutor. **21**(3), 2600–2636 (2019)
11. McKeown, N., et al.: OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
12. Meidan, Y., et al.: N-BaIoT - network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Comput. **17**(3), 12–22 (2018)
13. Nguyen, T.D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., Sadeghi, A.R.: DIoT: a federated self-learning anomaly detection system for IoT. In: IEEE 39th International Conference on Distributed Computing Systems, Dallas, Texas, USA, 7–9 July 2019, pp. 756–767 (2019)
14. Zolanvari, M., Teixeira, M.A., Jain, R.: Effect of imbalanced datasets on security of industrial iot using machine learning. In: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 112–117, November 2018. https://doi.org/10.1109/ISI.2018.8587389