A deep learning approach combining autoencoder with supervised classifiers for IoT anomaly detection

Nguyen Huu Noi[1], Doan Van Hoa[2*], Tran Nguyen Ngoc[1*]

[1]Military Technical Academy, Hanoi, Vietnam;
[2]Academy of Military Science and Technology, Hanoi, Vietnam.
[*]Corresponding author: doanvanhoa@gmail.com, ngoctn@lqdtu.edu.vn
Received 2 Oct. 2023; Revised 11 Dec. 2023; Accepted 12 Dec. 2023; Published 30 Dec. 2023.
DOI: https://doi.org/10.54939/1859-1043.j.mst.CSCE7.2023.98-110

ABSTRACT

Anomaly detection for Internet of things (IoT) networks is a challenging issue due to the huge number of devices that connect to each other and generate huge amounts of data. In this study, we propose a model combining Autoencoder (AE) with classification algorithms to build an end-to-end architecture for processing, feature extraction and data classification. Autoencoder is used to extract valuable hidden features of the original data, while supervised learning algorithms such as Softmax, Random Forest, Decision Trees, XGBoost, etc. are used for training and testing on AE's encoder output data. We then test our recommended models on nine recent devices in the NBaIoT dataset and evaluate their performance. According to the experimental results, the proposed model greatly improves the performance of IoT anomaly detection methods.

Keywords: IoT; Autoencoder; Anomaly detection; Supervised learning.

1. INTRODUCTION

IoT devices and services are growing in popularity in the current era, but IoT security is facing many difficulties. IoT attacks are becoming more frequent and sophisticated, affecting different layers and components of an IoT system, such as devices, gateways, servers, and networks. One of the remarkable attacks was Mirai, a distinctive type of botnet that exploits vulnerabilities in IoT devices to execute large-scale Distributed Denial of Service (DdoS) attacks [1].

Malware detection has benefited from machine learning methods, which are more robust and effective [2]. Many different machine learning algorithms have been used to perform collection, processing and analysis of IoT data, thereby detecting anomalies in IoT device systems. Malware detection has also made significant progress in recent times, as evidenced by the results of detecting and classifying IoT malware [3]. The improvement of computing power in recent times has also increased the efficiency of machine learning methods in processing large amounts of data generated by IoT devices.

The analysis of IoT malware can use static, dynamic, or hybrid methods. Ganesh et al. [4] used machine learning techniques to analyse and detect Mirai botnet attacks in IoT systems. The dataset N-BaIoT dataset [5] is using for training and testing through artificial neural networks (ANN). Li et al. [6] applied ensemble learning to detect malicious code in the cloud computing environment. Carrilo et al. [7] investigated the forensic and reverse engineering aspects of malware characterization. They also employed clustering techniques to discover new malware.

There are different algorithms are used in unsupervised and semi-supervised learning manner [8]. Deep Belief Networks (DBNs) and deep AutoEncoders (AEs) have been employed to extract useful features from the original data for improved

detection/classification methods [9]. AEs are commonly used to learn features in both semi-supervised and unsupervised learning. AEs have shown great performance in detecting anomalies in cybersecurity by modeling one class of data, typically the normal data. The hidden representation of AEs can also capture multiple classes of data for clustering methods in unsupervised learning, such as in [10]. Nguyen et al. [11, 12] proposed a hybrid model of AE and Self-organizing maps (SOM) to detect IoT malware. The original model AESOM [11] and its improved version DAESOM [12] work well with original IoT data, mapping the inputs to groups and detecting anomalies. Those models achieved good results for the IoT dataset.

This study presents a model that combines AEs and supervised classification algorithms for anomaly detection. AEs can transform input data into a feature space that captures more meaningful features of normal behavior, while classifiers can distinguish data types accurately. Classifiers that use lower dimensional data, such as the latent representation of AE (which has more powerful features), may outperform classifiers that use the original input data.

Our main contributions in this article are presented as follows:

- We propose a model that combines Autoencoder with supervised classification algorithms to form a complete framework for IoT anomaly detection. The model uses AE's ability to learn latent representation. Its output data then will be passed through the classification algorithms that perform the training;

- We perform testing on various datasets to evaluate the performance of the referral model. The obtained results show that the model combines AE with classifiers when only using the classifier on the original datasets.

The rest of this article consists of five parts: Part 2 presents the background of the research. In part 3, we present our proposed model and algorithms. Experimental setup is reported in part 4. In part 5, we present and discuss the experimental results. Finally, in part 6, we summarize our main achievement and suggest some ideas for future research.

2. BACKGROUND

2.1. Autoencoder

2.1.1. Traditional AE

An AE is a neural network that consists of two components: an encoder and a decoder. The task of the encoder is to transform the original data into a latent representation, while a decoder reconstructs the input data from the lower-dimensional representation. The basic architecture of AE is presented in the second phase in figure 1. The layer h defines a code for representing the input [13].

Given data x without a label, the encoder function f and the decoder function s; the AE equations are constructed as follows.

$$\begin{aligned} z &= f(x) = a_e(wx + b) \\ \hat{x} &= g(z) = g\big(f(x)\big) = a_d(w'.f(x) + b') \end{aligned} \tag{1}$$

where $\hat{x}$ is reconstruction of x , a_e is the encoder activation function, a_d is the decoder activation function.

The loss function of the reconstruction process can be optimized by minimizing the mean squared error loss between x and $\hat{x}$.

$$\begin{aligned} L_{AE}(x,\hat{x}) &= |x-\hat{x}|^2 = |x - a_d(w'.f(x)+b')|^2 \\ &= |x - a_d(w'.a_e(wx+b)+b')|^2 \end{aligned} \quad (2)$$

2.1.2. Denoising AE

The DAE representation [13] performs like traditional AE, but it does not reconstruct directly the input from the latent representation. First, the corrupted version $\tilde{x}$ of the original input x is created by adding the noise x_{noise} to x. The noise is the additive isotropic Gaussian noise and it is drawn by the formula: $x_{noise} = \mathcal{N}(0, \sigma_{noise})$. The value of σ_{noise} is 1.0 in experiments. Next, the real input version $\tilde{x}$ is computed by appending noise to x by the formula: $\tilde{x} = x + x_{noise}$.

Similar to AE, optimizing the DAE is to minimize the reconstruction loss. It is the difference between the original input x and the output $\hat{x}$. It is represented in the following equations:

$$\begin{aligned} L_{DAE}(x,\hat{x}) &= |x-\hat{x}|^2 = |x - a_d(w'.a_e(w\tilde{x}+b)+b')|^2 \\ &= |x - a_d(w'.a_e(w(x+x_{noise})+b)+b')|^2 \end{aligned} \quad (3)$$

where the output $\hat{x}$ is the reconstruction of $\tilde{x}$, while a_e and a_d are the activation functions of the encoder and decoder, respectively.

2.1.3. Sparse AE

A SParse AE [13] is a type of autoencoder that uses sparsity to create an information bottleneck. In SPAE the loss function is designed to penalize the activations within a layer. The sparsity constraint can be enforced with L_1 regularization or a Kullback–Leibler (KL) divergence between the expected average neuron activation and an ideal distribution p.

SPAE has a sparsity penalty $\Omega(h)$ on the code layer h, besides the reconstruction error, as part of its training criterion:

$$L_{spae} = L\left(x, g\big(f(x)\big)\right) + \Omega(h) \quad (4)$$

where $g(h)$ is the decoder output and typically we have $h = f(x)$, the encoder output.

SPAE is often used to learn features for another task such as classification. The penalty $\Omega(h)$ is a regularizer term added to the code at hidden layer.

2.1.4. Variational AE

The VAE [13] is a type of model that can infer hidden variables from data using learned functions and can be optimized using only gradients. It draws a sample z from the code distribution $p_\theta(z)$ (θ are the model parameters) to generate samples for the model. The generator network $g(z)$, which can compute gradients, then transforms the sample. At the final step, x is randomly drawn from a distribution $p_\theta\big(x; g(z)\big) = p_\theta(x|z)$.

The main idea of VAE is that they can be trained by increasing the variational lower bound $L(q)$ that corresponds to each data point x:

$$\begin{aligned} L(q) &= E_{z\sim q(z|x)} \log\, p_\theta\,(z,x) + H\big(q(z|x)\big) \\ &= E_{z\sim q(z|x)} \log p_\theta(x|z) - D_{KL}\,(q(z\,|\,x)||p_\theta\,(z)) \le \log p_\theta(x) \end{aligned} \quad (5)$$

In Eq. (5), the first term is the the reconstruction log-likelihood found in other AEs. The second term is the entropy of the approximate posterior. It tries to make the approximate posterior distribution $q(z|x)$ and the model prior $p_\theta(z)$ approach each other.

2.1.5. Shrink AE

Shrink AE [14] is a type of Autoencoder, which adds a regularizer to the reconstruction loss. The regularizer is designed to penalize normal datapoints to lie closely to the origin. The loss function of SAE is presented as follows:

$$\begin{aligned} L_{SAE}(x, \hat{x}, z) = & \ |x - \hat{x}|^2 \ + \ \lambda|z|^2 \\ & = \ |x - a_d(w'.a_e(w(x \ + \ x_{noise}) + b) + b')|^2 \ + \lambda|a_e(w\tilde{x} \ + b)|^2 \end{aligned} \quad (6)$$

where $\hat{x}$ and z are the reconstruction and the latent vector of the x respectively. The first term is the reconstruction error, $|x - \hat{x}|^2$, and the second term $\lambda|z|^2$ is the shrink regularizer. The parameter λ controls the trade-off between the two terms in the loss function.

2.1.6. Dirac delta VAE

A Dirac delta VAE [14] is is a type of VAE that uses a discrete latent variable with a Dirac delta prior distribution. However, a discrete latent variable poses a challenge for backpropagation, which is the algorithm used to update the network parameters based on the gradient of the loss function. A DVAE solves this problem by using a Dirac delta prior distribution over the discrete latent variable, which is a distribution that assigns a probability of one to a single point and zero to all other points. A Dirac delta prior distribution can be seen as a limit of a Gaussian distribution with a very small variance. The loss function of DVAE is presented as follow:

$$\begin{aligned} L_{DVAE}(\theta, \phi, x^i) = & \frac{1}{n}\sum_{1}^{n} \left\|x^i - \hat{x}^i\right\|^2 \\ & + \lambda \frac{1}{2\alpha}\sum_{j=1}^{J}\left[\left(\sigma_j^i\right)^2 + \left(\mu_j^i\right)^2 - \alpha - \alpha\log\alpha - \alpha\log\left(\left(\sigma_j^i\right)^2\right)\right] \end{aligned} \quad (7)$$

where λ is a trade-off hyperparameter between loss components, α is a constant ($\alpha \ll 1.0$), σ is a variance of the distribution.

2.2. Classification Algorithms

2.2.1. Softmax

Softmax (SM) is a mathematical function used in machine learning to map input values to probabilities. It is a generalization of the logistic function and can be used for multiclass classification. Softmax converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. It is commonly used as an activation function in neural network models for multi-class classification tasks. The output of the softmax function represents the probability distributions of a list of potential outcomes.

2.2.2. Random Forest

The random forest (RF) algorithm is a powerful classifier in machine learning. Although it is a classification algorithm, it can also be used for regression problems. The

algorithm works by building a "forest" of decision trees, each of which is trained on a data sample with the characteristics of that data sample. The decision-making algorithm is based on the voting of the trees.

A random forest algorithm can produce accurate and stable results without much hyperparameter tuning, and it can handle both categorical and numerical features. It can also measure the importance of each feature and deal with missing values.

2.2.3. K-nearest Neighbors

The k-nearest Neighbor (KNN) is a supervised learning method that can be used for classification and regression problems. It is based on the idea that similar data points tend to be close to each other in a feature space, and that the label or value of a query point can be inferred from its nearest neighbors.

To use KNN, there are two things need to be specified: the distance metric and the number of neighbors. The distance metric is a function that measures how far two data points are from each other. There are different ways to define the distance, such as Euclidean distance, Manhattan distance, or cosine similarity. The number of neighbors, denoted by k, is a positive integer that determines how many closest data points are considered for the prediction. For example, if k = 3, then the algorithm will look at the three nearest neighbors of the query point and use them to make a decision.

2.2.4. Support Vector Machine

The support vector machine (SVM) algorithm is a supervised machine learning technique that can be used for classification, regression, or anomaly detection tasks. It works by finding the optimal hyperplane that separates the data points into different classes or predicts their values. The hyperplane is a decision boundary that maximizes the margin between the closest data points of different classes, which are called support vectors. SVM can handle linearly and non-linearly separable data by using different kernels, which are functions that transform the data into a higher-dimensional space where they can be more easily separated. SVM is a powerful and flexible algorithm that can achieve high accuracy and robustness in many applications, such as image recognition, text analysis, spam detection, face detection, and network security.

2.2.5. AdaBoost

An AdaBoost (ADA) classifier is a machine learning technique that can improve the accuracy of a base classifier by combining multiple weak classifiers. A weak classifier is a classifier that performs slightly better than random guessing, while a strong classifier is a classifier that has a high accuracy and low error rate. An AdaBoost classifier works by iteratively adding weak classifiers to the ensemble and adjusting their weights based on their performance on the training data. The final prediction is made by taking a weighted majority vote of the ensemble members.

3. METHODOLOGY

3.1. Model Architecture

This section presents the architecture design of our proposed model for detecting IoT malware. It is a combination of AE and classifiers, consisting of three phases for data preprocessing, model training and model testing as shown in figure 1.

Phase 1: Preprocessing of input data. With machine learning models, the processing of input data is crucial to whether the model works effectively. At this stage we use different techniques to process data such as: cleaning, missing handling, normalization and standardization.

Phase 2: We use AE to build a new feature space from the original data. The new feature space has a lower dimensionality and is expected to have stronger features that represent the properties of the data well. The classification algorithms will use the hidden layer returned by AE to train the model.

Phase 3: The trained models are deployed later for classifying the coming data. The model will categorize the test instance into two classes: benign or attack.

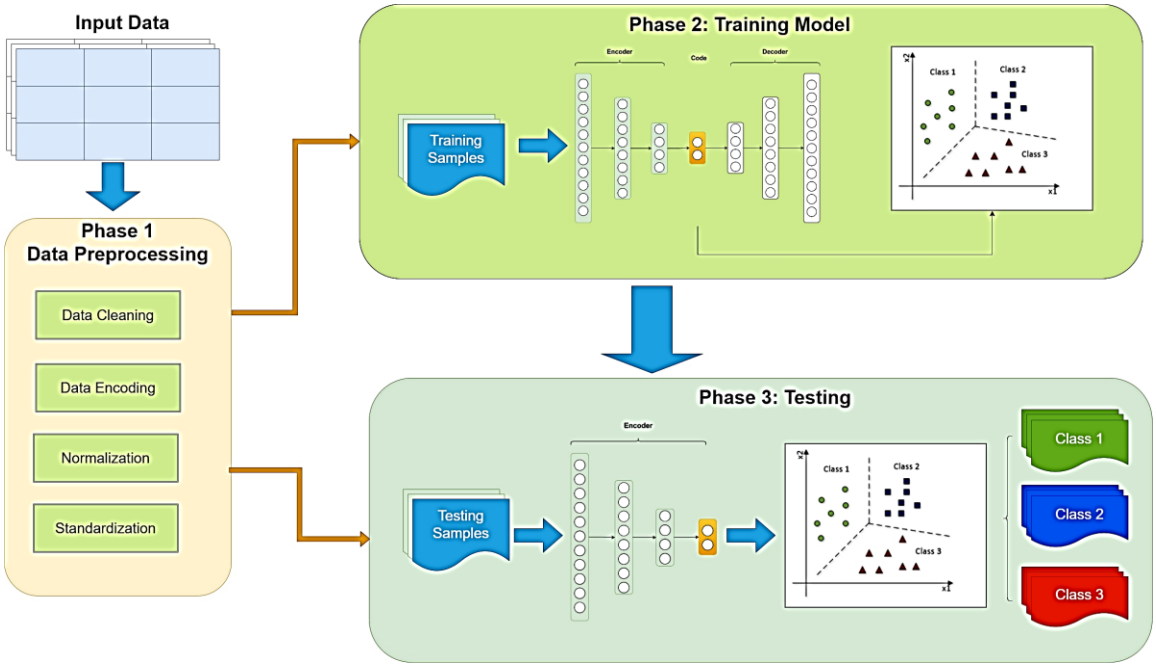

Figure 1. *Model Architecture.*

3.2. Latent Representation

An AE learns a latent representation from input data, which usually contains unlabelled data from normal and abnormal data. The latent space for feature representation has a lower dimensionality and reveals more significant features. The latent space is used for the next traning phase. We use a standard architecture for AEs with 5 layers in both the encoder and decoder. The AE architecture is shown in figure 1. We keep the same number of layers as in the study [11], but we change the number of neurons in each layer to suit the IoT malware data. The AE is trained in an semi-supervised way. The dataset used for training contains only normal data. The hidden layer (bottleneck) has a size of 29 as described in [11].

Autoencoder learns to minimize the reconstruction loss, which is the difference between the input and the output, such as the least-squares error:

$$\min L = \sum_{i=1}^{n} |x_i - \hat{x_i}|^2 = \sum_{i=1}^{n} |x_i - g(f(x_i))|^2 \quad (8)$$

where x_i is the input sample, $f(x_i)$ is the encoded output, $f(\cdot)$ is the encoder, and $\widehat{x_i}$ is the decoded output of the decoder $g(\cdot)$ that reconstructs the input.

We use autoencoders (AE) to learn latent representations of the original data, which can extract useful features and reduce data dimensionality before feeding them to classification models. In our previous studies [11, 12], we analyzed the data and found that benign data and attack data tend to be distant from each other. Benign data are usually close to the origin (after scaling origin data to the range [-1.0, 1.0]), while attack data are farther away. Based on this observation, we use only benign data to train the AE models. Different AE models additionally use different techniques (denoising, penalty, adversarial sample generation) to help the AE model have the best possible set of parameters after training. The goal of this training phase is to separate attack data from benign data and push benign data as near to the origin as possible.

3.3. Anomaly Classification

IoT anomaly detection can be benefited by machine learning, such as deep learning, to analyze data from IoT devices and identify abnormal patterns or behaviors [15]. These devices include sensors, smart home devices, wearable devices, and more. IoT anomaly detection can help in various domains, such as fault detection, health monitoring, security, and energy saving. By spotting anomalies, IoT anomaly detection can help prevent or fix potential problems or risks in IoT systems.

In this phase, we train anomaly classifiers using the encoder's output data. The training data consists of normal and attack data with equal proportions. We also balance the data among different types of attacks by randomly sampling them. The problem becomes a binary classification problem, as the data has only two labels: normal and abnormal. The algorithms such as SM, RF, KNN, SVM and ADA are used to classify this type of data.

4. EXPERIMENTS

In this section, we present the description of datasets, experimental settings, and evalu-ation metrics for our model in detecting IoT attacks. The IoT dataset with nine device types was used, as described in the subsection below.

4.1. Dataset

The dataset N-BaIoT[1] was first presented by Y. Meidan et al. [5]. It has data samples from nine different IoT devices. These devices belong to four categories: doorbell (Danmini and Ennio), thermostat (Ecobee), monitor (Philips B120N10 baby monitor), and camera/webcam (Provision PT-737E security camera, Provision PT-838 security camera, Samsung SNH-1011-N Webcam, SimpleHome XCS7-1002-WHT security camera, Simple Home XCS7-1003-WHT security camera). Each sample in the dataset has 115 features, which were produced by Kitsune [16]. More details about the dataset is shown in table 1.

[1] https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT

Table 1. *Dataset description.*

ID	Device Name	Type	Benign	Gafgyt	Mirai
D1	Danmini Doorbell	Doorbell	49548	652100	316650
D2	Ecobee Thermostat	Thermostat	13113	512133	310630
D3	Ennio Doorbell	Doorbell	39100	316400	
D4	Philips B120N10 Baby Monitor	Monitor	175240	312273	610714
D5	Provision PT 737E Security Camera	Camera	62154	330096	436010
D6	Provision PT 838 Security Camera	Camera	98514	309040	429337
D7	Samsung SNH 1011 N Webcam	Webcam	52150	323072	
D8	SimpleHome XCS7 1002 WHT Security Camera	Camera	46585	303223	513248
D9	SimpleHome XCS7 1003 WHT Security Camera	Camera	19528	316438	514860

4.2. Parameters Setting

The AE in the model is designed with five layers with the sizes of hidden layers as described in [12]. We set the ratio between the size of the encoded layer and the original feature space as 0.25. Thus, the size of the bottleneck (encoded layer) is 29 (0.25*115). Finally, the sizes of other hidden layers are set as {115, 86, 58, 38, 29} (the encoded layer) of the input layer's size, respectively.

The activation function used is ReLU. The number of epochs is 500, the batch size is 64 and the learning rate is 1e-3. Through training and testing, the original dataset is split into two parts: one for training and one for testing with 80% samples used for training, 20% samples used for testing. For classifiers, the default hyperparameters are triggered as defined in sciki-learn[2] library.

4.3. Evaluation Metrics

4.3.1. AUC

To evaluate the effectiveness of proposed methods on different scenarios, we utilize the precision and the AUC (Area Under the Curve).

First, we calculate the true positive rate (TPR) and false positive rate (FPR) using the following formulas.

$$TPR = \frac{TP}{TP + FN};\ FPR = \frac{FP}{FP + TN} \tag{9}$$

The TPR is plotted against FPR to get the Receiver Operating Characteristic curve (ROC curve) at different thresholds. Lowering the threshold makes more items positive, which increases both FP and TP values. The AUC is the whole area under the ROC curve. AUC is a measure of performance for all possible thresholds. If the AUC is around 0.5, the model does not discriminate well, 0.7 to 0.8 is acceptable, 0.8 to 0.9 is excellent and more than 0.9 is outstanding.

4.3.2. FAR and MDR

The two metrics are False Alarm Rate (FAR) and Miss Detection Rate (MDR) are also used to evaluate the effectiveness of the proposed model. FAR measures the proportion of negative samples that are incorrectly classified as positive. MDR measures the proportion of positive samples that are incorrectly classified as negative.

[2] https://scikit-learn.org/

$$FAR = \frac{FP}{FP + TN};\ MDR = \frac{FN}{FN + TP} \quad (10)$$

5. RESULTS AND DISCUSSION

In this section, we present and discuss the results of the experiment process. Experiments were implemented in Python using the Keras, scikit-learn, and Tensorflow frameworks. We conducted the experiments on a computer with Ubuntu 22.04 LTS, an Intel(R) Core i5 11400H CPU, 24 GB of RAM, and a Geforce 3050 GPU.

5.1. Anomaly Detection

We report the compact version of test results in table 2.

Table 2. *Anomaly classification for unknown attack.*

Method	Data	AUC					Data	AUC				
		SM	RF	KNN	SVM	ADA		SM	RF	KNN	SVM	ADA
w/o AE	D1	0.997	0.314	0.817	0.851	0.866	D6	0.997	0.314	0.817	0.851	0.851
AE		0.943	0.997	0.945	0.957	**1.000**		0.966	0.987	0.987	0.999	0.936
DAE		0.845	0.500	0.949	0.279	0.800		0.991	**1.000**	0.993	0.999	0.925
SPAE		0.857	0.993	0.999	0.997	0.993		**1.000**	**1.000**	0.998	0.999	0.940
VAE		0.887	0.850	0.827	0.917	0.891		0.907	0.799	0.894	0.964	0.933
SAE		**1.000**	**1.000**	0.998	**0.999**	0.986		0.944	**1.000**	**1.000**	**1.000**	**1.000**
DVAE		**1.000**	**1.000**	**1.000**	**0.999**	**1.000**		0.942	**1.000**	**1.000**	**1.000**	0.926
w/o AE	D2	0.997	0.314	0.817	0.851	0.950	D8	0.997	0.314	0.817	0.851	**1.000**
AE		0.939	0.999	0.945	0.999	**1.000**		0.998	0.993	0.978	0.997	**1.000**
DAE		**1.000**	**1.000**	**1.000**	**1.000**	0.937		0.294	0.447	0.442	0.896	0.498
SPAE		0.988	0.999	0.998	0.994	0.994		0.832	0.560	**1.000**	0.800	0.999
VAE		0.743	0.617	0.776	0.833	0.759		0.876	0.892	0.722	0.846	0.812
SAE		0.997	0.998	**1.000**	0.950	**1.000**		**1.000**	**1.000**	0.999	0.999	0.995
DVAE		**1.000**	**1.000**	**1.000**	**1.000**	**1.000**		0.975	0.999	**0.999**	**0.999**	**1.000**
w/o AE	D4	0.997	0.314	0.817	0.851	0.817	D9	0.997	0.314	0.817	0.851	**1.000**
AE		0.890	0.997	0.998	0.999	0.975		0.992	0.979	0.955	**0.999**	**1.000**
DAE		0.800	0.992	0.975	0.999	0.970		0.995	**1.000**	0.993	0.952	**1.000**
SPAE		**1.000**	0.997	0.999	0.999	**1.000**		**1.000**	**1.000**	**1.000**	0.800	0.990
VAE		0.793	0.730	0.810	0.884	0.804		0.865	0.903	0.804	0.918	0.876
SAE		**1.000**	**1.000**	**1.000**	**1.000**	**1.000**		**1.000**	**1.000**	**1.000**	**0.999**	**1.000**
DVAE		**1.000**	0.900	**1.000**	**1.000**	0.987		**1.000**	0.980	**1.000**	**0.999**	**1.000**
w/o AE	D5	0.997	0.314	0.817	0.851	**1.000**						
AE		0.982	0.988	0.947	**0.999**	0.998						
DAE		0.996	**1.000**	0.991	0.990	**1.000**						
SPAE		0.881	0.974	0.962	**0.999**	0.999						
VAE		0.923	0.910	0.853	0.957	0.947						
SAE		0.966	**1.000**	0.975	**0.999**	0.990						
DVAE		**1.000**	0.995	**0.999**	0.998	**1.000**						

The full experiment process contains seven datasets as described in table 1 (the D3

and D7 were excluded from the report table as they do not contain the Mirai data), six representation models based on AE and nine classification algorithms. We also test the anomaly detection algorithms on the original dataset to evaluate the performance of the representation models. Evaluation results are based on the AUC and F1-score criteria, as described in section 4.3. For each dataset, we work on seven different models: keeping the original data structure (without using AE-based representation), AE, DAE, SPAE, VAE, SAE and DVAE. The classification algorithms include SM, RF, KNN, SVM, and ADA. All results are presented on matrix view to better show the collected results.

We perform many different experiments with the different settings and metrics to evaluate the effectiveness of the proposed model, as well as to find out the effective algorithms for IoT network anomaly detection. In table 2, we present the evaluation results according to the AUC criterion for the Gafgyt dataset. In this table, only the SM, RF, KNN, SVM and ADA algorithms are included.

Let's analyze the obtained results in the table 2. With the device D1, DVAE appears to be the superior performing method as the outputs of this method give good classification results with all classification algorithms. With an accuracy of up to three decimal places, DVAE+(SM, RF, KNN, SVM, ADA) all give a result 1.0. Ranked second among these methods is SAE, when it gives results as good as DVAE with the SM, RF, SVM algorithms. Only AE gives results as good as DVAE with the ADA classifier. Considering the D2 dataset, DVAE still gives the best results, followed by DAE (good on the SM, RF, KNN and SVM algorithms). With the D4 dataset, SAE ranked first with all classification results being 1.0. Second is DVAE, followed by SPAE. With the D5 dataset, the data seem to be differentiated so the classification algorithms on the outputs of the representation methods have no precision. However, DVAE is still at the top with three better cases, followed by DAE. On the D6, D8 and D9 datasets, DVAE as well as SAE are both methods that give the best results than the other methods. In common, the two representation methods SAE and DVAE give quite good results on the Gafgyt dataset according to the AUC criterion. Specifically, DVAE is a bit better than SAE because DVAE tries to force the data to each cluster center, so the data after representation are separated into different clusters (benign and attack). SAE also tries to do the same thing, however SAE forces the benign data closer to the cluster center than when forcing the attack data. Recall that we use both the benign and the attack data for the training process.

The general conclusion with the experimental process of anomaly detection using a combination of representation methods and classification algorithms is that the model that combines SAE/DVAE with classification algorithms gives good results compared to other methods. Also from experimental results, we also note that ADA gives more stable results than the remaining classification algorithms (not only on SAE/DVAE representation methods but also when run independently or in combination with the other representation methods).

5.2. Confusion assessment

The Gafgyt and Mirai botnets are two types of malware that infect IoT devices and launch DDoS attacks. Gafgyt is a simpler version of an IRC model and it mainly uses SYN, UDP and ACK Flooding attacks. Mirai is a more advanced and dangerous malware that can target devices with different architectures and perform different types

of DDoS attacks based on the TCP, UDP or HTTP protocols. Both Gafgyt and Mirai can create multiple DDoS attacks, generating different network traffic patterns and feature values for the infected devices.

This experiment aims to test the stability and efficiency of the AE-based representation methods when combined with the classification models. These models are trained and tested on one botnet or another. We consider two scenarios: (1) train and test on the same attack data and (2) train on one attack data, such as Gafgyt, and test on another attack data, such as Mirai.

Table 3. *Confusion Assesment.*

Method	**Gafgyt/Mirai**			**Mirai/Gafgyt**		
	AUC	**FAR**	**MDR**	**AUC**	**FAR**	**MDR**
w/o AE	0.997	0.040	0.150	0.989	0.100	0.400
AE	0.943	0.003	0.111	0.802	0.060	0.397
DAE	0.845	0.030	0.110	0.802	0.060	0.090
SPAE	0.857	0.030	0.050	0.765	0.073	0.300
VAE	0.887	0.025	0.080	0.880	0.098	0.141
SAE	**1.000**	**0.010**	**0.040**	**0.999**	**0.050**	0.097
DVAE	**1.000**	**0.010**	**0.040**	0.989	0.060	**0.070**

In this test, AUC, FAR, MDR metrics are used to evaluate the effectiveness of the model. The device used for evaluation is D1. All representation methods are used (w/o AE, AE, DAE, SPAE, VAE, SAE, DVAE). The classification algorithm used is Softmax. In table 3, we present the results for cross-validation process. The model is trained with one type of attacks and tested with another type. When training with Gafgyt and testing with Mirai, SAE and DVAE showed similar results on all criteria sets. When the model was trained with Mirai and evaluated with Gafgyt, SAE gave better results than the remaining models according to the AUC and FAR criteria, and DVAE gave better results according to the MDR criterion.

The overall evaluation according to this experiment shows that the SAE+SM model has higher anomaly detection rates and lower false positive rates than the other competing models.

6. CONCLUSIONS

In this paper, we present a novel hybrid model for detecting IoT attacks on NBaIoT dataset, which combines an Autoencoder network with a supervised classification algorithm. Our main contribution is to integrate these methods into an end-to-end system that can identify anomalies and classify attack data. We evaluate our model on different datasets and devices, and demonstrate its ability to detect not only known attacks, but also unknown ones. We find that the best performance is achieved by using a traditional AE with the Softmax algorithm. Moreover, our method is suitable for real-time systems, as it has low model loading and classification time.

In the future, our method can be extended in the following way. First, we can enhance the hidden layer representation technique to capture the latent features of the data. Second, we can test our model on various datasets to further validate its effectiveness. For instance, we can apply our method to anomaly detection problem in computer networks.

REFERENCES

[1]. C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, *"DDoS in the IoT: Mirai and other botnets,"* Computer (Long Beach Calif), vol. 50, no. 7, pp. 80–84, (2017), DOI: 10.1109/MC.2017.201.

[2]. M. Asam et al., *"IoT malware detection architecture using a novel channel boosted and squeezed CNN,"* Scientific Reports 2022 12:1, vol. 12, no. 1, pp. 1–12, (2022), DOI: 10.1038/s41598-022-18936-9.

[3]. S. Li, Q. Zhang, X. Wu, W. Han, and Z. Tian, *"Attribution classification method of APT malware in IoT using machine learning techniques,"* Secur. Commun. Netw., (2021), DOI: 10.1155/2021/9396141.

[4]. T. G. Palla and S. Tayeb, *"Intelligent Mirai Malware Detection in IoT Devices,"* IEEE World AI IoT Congress, AIIoT 2021, pp. 420–426, (2021), DOI: 10.1109/AIIOT52608.2021.9454215.

[5]. Y. Meidan et al., *"N-baiot—network-based detection of iot botnet attacks using deep autoencoders,"* IEEE Pervasive Comput, vol. 17, no. 3, pp. 12–22, (2018).

[6]. S. Li, Y. Li, W. Han, X. Du, M. Guizani, and Z. Tian, *"Malicious mining code detection based on ensemble learning in cloud computing environment,"* Simul. Model. Pract. Theory, vol. 113, p. 102391, (2021), doi: 10.1016/j.simpat.2021.102391.

[7]. J. Carrillo-Mondéjar, J. L. Martínez, and G. Suarez-Tangil, *"Characterizing Linux-based malware: Findings and recent trends,"* Futur. Gen. Comput. Syst., vol. 110, pp. 267–281, (2020), doi: 10.1016/j.future.2020.04.031.

[8]. G. Pang, C. Shen, L. Cao, A. H.-A. computing surveys (CSUR), and undefined 2021, *"Deep learning for anomaly detection: A review,"* dl.acm.org, vol. 54, no. 2, (2020), DOI: 10.1145/3439950.

[9]. L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, *"Deep Transfer Learning for IoT Attack Detection,"* IEEE Access, vol. 8, pp. 107335–107344, (2020), DOI: 10.1109/ACCESS.2020.3000476.

[10].L. Vu, V. L. Cao, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Learning Latent Representation for IoT Anomaly Detection," IEEE Trans Cybern, pp. 1–14, (2020), doi: 10.1109/tcyb.2020.3013416.

[11].H. N. Nguyen, V. C. Nguyen, N. N. Tran, and V. L. Cao, *"Feature Representation of AutoEncoders for Unsupervised IoT Malware Detection,"* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 13076 LNCS, pp. 272–290, (2021), DOI: 10.1007/978-3-030-91387-8_18/COVER.

[12].H. N. Nguyen, N. N. Tran, T. H. Hoang, and V. L. Cao, *"Denoising Latent Representation with SOMs for Unsupervised IoT Malware Detection,"* SN Computer Science 2022 3:6, vol. 3, no. 6, pp. 1–15, (2022), DOI: 10.1007/S42979-022-01344-1.

[13].I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *"Deep learning"*, vol. 1, no. 2. MIT press Cambridge, (2016).

[14].V. L. Cao, M. Nicolau, and J. McDermott, *"Learning Neural Representations for Network Anomaly Detection,"* IEEE Trans Cybern, vol. 49, no. 8, pp. 3074–3087, (2019), DOI: 10.1109/TCYB.2018.2838668.

[15].A. Chatterjee and B. S. Ahmed, *"IoT anomaly detection methods and applications: A survey,"* Internet of Things, vol. 19, p. 100568, (2022), doi: 10.1016/J.IOT.2022.100568.

[16].Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, *"Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection,"* (2018), DOI: 10.14722/ndss.2018.23204.

TÓM TẮT

Phương pháp tiếp cận học sâu kết hợp Autoencoder với các thuật toán phân loại giám sát cho phát hiện bất thường mạng IoT

Phát hiện bất thường cho mạng IoT đang là một vấn đề thách thức do số lượng rất lớn các thiết bị kết nối với nhau và tạo ra một lượng dữ liệu khổng lồ. Trong nghiên cứu này, chúng tôi đề xuất một mô hình kết hợp giữa Autoencoder với các thuật toán phân loại để xây dựng một kiến trúc đầu cuối cho việc xử lý, trích xuất đặc trưng và phân loại dữ liệu. Autoencoder được sử dụng để trích xuất những đặc trưng ẩn, có giá trị từ dữ liệu ban đầu, trong khi các thuật toán học có giám sát như Softmax, Random Forest, K-nearest Neighbors, Support Vector Machine, AdaBoost được sử dụng để huấn luyện và kiểm tra trên dữ liệu đầu ra của AE. Sau đó, chúng tôi kiểm tra các mô hình đề xuất trên chín thiết bị gần đây trong bộ dữ liệu N-BaIoT và đánh giá hiệu suất của chúng. Theo kết quả thực nghiệm, mô hình đề xuất cải thiện đáng kể hiệu suất của các phương pháp phát hiện bất thường IoT.

Từ khóa: IoT; Autoencoder; Phát hiện bất thường; Học giám sát.