Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.034277
Article

Tech Science Press

A Lightweight Deep Autoencoder Scheme for Cyberattack Detection in the Internet of Things

Maha Sabir[1], Jawad Ahmad[2,*] and Daniyal Alghazzawi[1]

[1]Information Systems Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, 80200, Saudi Arabia
[2]School of Computing, Edinburgh Napier University, Edinburgh EH10 5DY, UK
*Corresponding Author: Jawad Ahmad. Email: J.Ahmad@napier.ac.uk
Received: 12 July 2022; Accepted: 22 September 2022

Abstract: The Internet of things (IoT) is an emerging paradigm that integrates devices and services to collect real-time data from surroundings and process the information at a very high speed to make a decision. Despite several advantages, the resource-constrained and heterogeneous nature of IoT networks makes them a favorite target for cybercriminals. A single successful attempt of network intrusion can compromise the complete IoT network which can lead to unauthorized access to the valuable information of consumers and industries. To overcome the security challenges of IoT networks, this article proposes a lightweight deep autoencoder (DAE) based cyberattack detection framework. The proposed approach learns the normal and anomalous data patterns to identify the various types of network intrusions. The most significant feature of the proposed technique is its lower complexity which is attained by reducing the number of operations. To optimally train the proposed DAE, a range of hyperparameters was determined through extensive experiments that ensure higher attack detection accuracy. The efficacy of the suggested framework is evaluated via two standard and open-source datasets. The proposed DAE achieved the accuracies of 98.86%, and 98.26% for NSL-KDD, 99.32%, and 98.79% for the UNSW-NB15 dataset in binary class and multi-class scenarios. The performance of the suggested attack detection framework is also compared with several state-of-the-art intrusion detection schemes. Experimental outcomes proved the promising performance of the proposed scheme for cyberattack detection in IoT networks.

Keywords: Autoencoder; cybersecurity; deep learning; intrusion detection; IoT

1 Introduction

The IoT is most commonly referred to as the interconnection of smart sensors and devices with the internet. The IoT could be established with multiple types of devices, including environmental sensors to consumer electronics or wearable devices. The IoT has been incorporated into a vast variety of applications such as healthcare, industry, transportation, agriculture, smart buildings, etc. [1]. The interconnection of smart devices through IoT networks allows the exchange of valuable information and

data among themselves. However, this exchange of information could be intercepted by the attackers and intruders, which consequently compromises the security and privacy of all devices and the complete IoT architecture [2]. The intrusion detection system (IDS) can be useful in ensuring the security of IoT networks. The IDS can detect several malicious activities, such as security violations and unauthorized access to valuable information in the IoT network [3,4].

In this article, a lightweight deep autoencoder (DAE) scheme is presented for cyberattack detection in IoT networks. The suggested technique learns the normal and malicious patterns of the data to successfully identify the anomalous behavior of the network. One of the primary objectives for developing the suggested DAE framework is to minimize the model's overall complexity by reducing the number of operations. It reduces the computational cost along with the energy consumption and makes this IDS more feasible to be implemented in resource-constrained IoT networks. The proposed DAE-based IDS has been evaluated in both binary-class and multi-class scenarios by using two standard and open-source datasets such as NSL-KDD, and UNSW-NB15.

The rest of the article is organized as follows. Section 2 comprises the latest research related to the IDSs for IoT. A detailed research methodology of the proposed framework is presented and discussed in Section 3. Section 4 presents the implementation procedure and a detailed discussion of the obtained results. Finally, Section 5 concludes the outcomes of this research.

2 Literature Review

This section presents an overview of some of the latest research on DL-based intrusion detection algorithms for IoT networks. The discussed studies have been categorized according to the proposed DL schemes, utilized datasets, and performance metrics.

Parra et al. [5] introduced a cloud-based DL approach for cyberattack detection in the IoT. The suggested scheme contains an integration of two deep learning schemes, including a CNN and LSTM network. The performance of the suggested technique was analyzed through the N-BaIoT dataset. The experimental outcomes demonstrate that the proposed scheme efficiently detected phishing and botnet attacks with higher accuracies. Shone et al. [6] introduced a nonsymmetric deep autoencoder (NDAE) for unsupervised feature learning. Researchers presented a novel DL-based intrusion detection model using stacked NDAE. The proposed model was implemented in a graphics processing unit (GPU)-enabled TensorFlow and performance was evaluated using NSL-KDD and KDDCup99 datasets. Awotunde et al. [7] introduced a deep feed-forward neural network (DFNN) with rule-based features selection method for cyberattack detection in the IIoT. The proposed architecture verifies the collected information of packets. The effectiveness of the suggested technique was analyzed using the two standard IoT security datasets. In another work, Attota et al. [8] presented a multiple-view federated learning-based IDS (MV-FLID). Researchers utilized the grey wolf optimization technique (GWO) for feature selection. The proposed model is trained and evaluated by using a lightweight MQTT protocol dataset.

Qaddoura et al. [9] designed an FNN-LSTM-based hybrid IDS for IoT networks. The designed hybrid scheme utilizes the smote oversampling method to equivalent the samples of each class and evaluates the suggested design through the IoTID20 dataset. Hassan et al. [10] proposed a reliable cyberattack detection scheme to improve the trustworthiness of an IIoT network. Researchers utilized an ensemble learning technique based on the combination of a random subspace (RS) with a random tree (RT) for cyberattack detection. The suggested scheme was tested over 15 datasets on the SCADA networks. Experimental findings indicated the superior performance of the proposed technique over conventional attack detection models. Li et al. [11] designed a bidirectional long and short-term memory (LSTM) network for cyberattack detection in the IIoT. In the proposed scheme, sequence and stage feature layers are introduced that facilitate the model to learn corresponding attack intervals from historical data. As

compared to some related works the proposed scheme demonstrated the lower false positive and false negative rates. Despite considerable work spent on annotating IoT traffic data, the number of labeled records remains very small, increasing the challenge of detecting assaults and intrusions. Growing IoT networks are becoming more vulnerable to various types of cyberattacks. Luo et al. [12] introduced a web attack detection system (WADS). Researchers proposed three DL-based models to detect the cyberattacks separately, and an ensemble classifier has been used for the final decision obtained from the combined results of all three DL models. The real-world and open-source security datasets have been utilized for performance evaluation. The experimental results proved the efficacy of the proposed framework to detect several web attacks with higher accuracy and lower false alarm rates.

Based on the aforementioned discussion, several researchers have done a great job in the development of ML/Dl-based intrusion detection schemes for IoT. However, there is still room for improvement in the development of a lightweight and advanced attack detection scheme that can improve attack detection accuracy, reduce the computational cost, and be highly compatible with the resource-constrained nature of the IoT networks. The proposed DAE presents significant advantages over existing schemes, such as compact design, and optimal hyperparameters selection. Furthermore, it reduces the computational complexity and energy requirements that tend to be helpful in the integration of IDS with resource-constrained networks.

3 Research Methodology

This section presents a detailed description of the proposed scheme including utilized datasets, the deep autoencoder (DAE) design, and the performance assessment parameters.

3.1 Datasets Description

The proposed scheme is evaluated through two publicly available IoT security datasets including NSL-KDD, and UNSW-NB15. In the following, a short description of each dataset is presented.

3.1.1 NSL-KDD

This is one of the most commonly used datasets for the benchmarking of modern-day internet traffic. This dataset contains 42 features per record, of which 41 features are considered input features, and the last feature is the label to determine whether it is a normal or malicious value. Furthermore, it contains 4 different classes of cyberattacks including Probe, Remote to Local (R2L), Denial of Service (DoS), and User to Root (U2R) [13].

3.1.2 UNSW-NB15

This is considered a new generation dataset and it was firstly published in 2015. This dataset has a total of 49 features and a variety of normal and malicious values with the class label of a total of 257,673 samples. It contains 164673 malicious and 93000 normal samples [14,15]. Features of this dataset are categorized into 6 groups which are named “basic”, “time”, “flow”, “content”, “additional generated”, and “labeled” features. Further, UNSW-NB15 has 9 different classes of modern attacks which include analysis, backdoor, Dos, exploits, fuzzers, generic, reconnaissance, shellcode, and Worms.

3.2 The Proposed Framework

The workflow of the suggested framework is depicted in Fig. 1. The main operation consists of three stages that include, data pre-processing, the mathematical model of the DAE, and performance assessment parameters. In the following sub-sections, details of each stage have been described briefly.

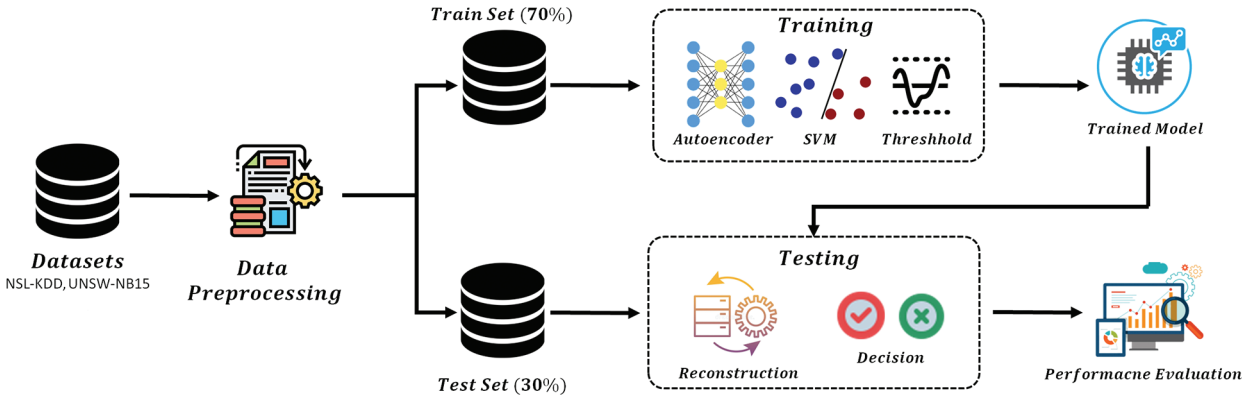

Figure 1: Block diagram of the proposed architecture

3.2.1 Data Pre-processing

It is one of the most important stages of each ML/DL model. It transforms the data into the most compatible form for input of any neural network. In our study, four different datasets have been utilized, hence, different procedures have been adopted for the pre-processing of each dataset.

a) Pre-processing of NSL-KDD: In the NSL-KDD dataset, data contains some text values. Therefore, we need to transform the nominal features into numeric values. At this stage, categorical features are transformed into numerical features by using one-hot encoding. Since the dataset is very large and there is a large variation between values, data normalization is also required for better performance. We'll use mean normalization here. It makes the values of each feature in the data have zero-mean and unit variance. We utilized "min-max scaling" for the normalization process. The detailed class distribution of the NSL-KDD dataset is presented in Table 1.

Table 1: Class distribution of the NSL-KDD dataset

Classes	Total samples	Training	Testing
Normal	77054	53938	23116
DoS	53387	37371	16016
Probe	14077	9854	4223
R2L	3880	2716	1164
U2R	119	83	36

b) Pre-processing of UNSW-NB15: This dataset contains 10 classes and 49 features. In preprocessing phase, we used two approaches data conversion and data normalization. In the first stage, the data conversion technique transforms all the categorical data into numerical data. In the second stage, the large variance of all features is reduced by using the "min-max scaling" technique. This technique removes all the invalid samples and scales the large values in the range of zero to one. A detailed distribution of the UNSW-NB15 dataset is presented in Table 2.

Table 2: Class distribution of the UNSW-NB15 dataset

Classes	Total samples	Training	Testing
Normal	93000	65100	27900
Analysis	2677	1874	803
Backdoor	2329	1630	699
DoS	16353	11447	4906
Exploits	44525	31168	13358
Fuzzers	24246	16972	7274
Generic	58871	41210	17661
Reconnaissance	13987	9791	4196
Shellcode	1511	1058	453
Worms	174	122	52

3.2.2 Mathematical Model of the Proposed Attack Detection Scheme

The proposed attack detection scheme contains three main stages including the feature extraction stage, feature selection stage, and classification stage. The first two stages perform the dimensionality reduction operation that increases the computational efficiency of the model to make it highly compatible with resource-constrained IoT devices. A DAE is used for the extraction of optimal features with mutual information (MI) and a support vector machine (SVM) is incorporated with a gradient descent (GD) algorithm to perform the detection process.

The basic design of the proposed DAE is shown in Fig. 2. The DAE is an unsupervised neural network that uses a backpropagation algorithm to learn from unlabeled information. The input and output values of DAE are the same that try to lean the hypothesis function.

$$h_{W,b}(r) \approx r \tag{1}$$

The DAE contains an encoder and decoder. The main function of the encoder is to compress the incoming information in low-dimensional representation. On the other hand, the decoder performs the reconstruction of data from the low-dimensional representation. In the encoding process, the input vectors are transformed into an abstract vector and the input data space's dimensionality is also reduced.

To accurately perform the encoding and decoding operations, multiple constraints are involved in the neural network. The selection of the hidden neurons less than the input features and some useful representation can be discovered during reconstruction operation. As a result, if there are any correlations among the features, the DAE will be able to discover them.

The constraint shown in Eq. (2) is applied to hidden neurons in the encoder that compresses the input data representation and performs feature extraction. Here $\hat{\delta}_z$ in Eq. (3) represents the average activation and $a_z(r)$ represent the activation of hidden neuron z. The neuron z is considered to be in an active or inactive state if the activation of a neuron is 1 and 0 respectively. The variable δ indicates the sparsity parameter and is usually set near zero to ensure the inactive state of neurons most of the time.

$$\hat{\delta}_z = \delta \tag{2}$$

$$\hat{\delta}_z = \frac{1}{m} \sum_{y=1}^{m} [a_z(r)] \tag{3}$$

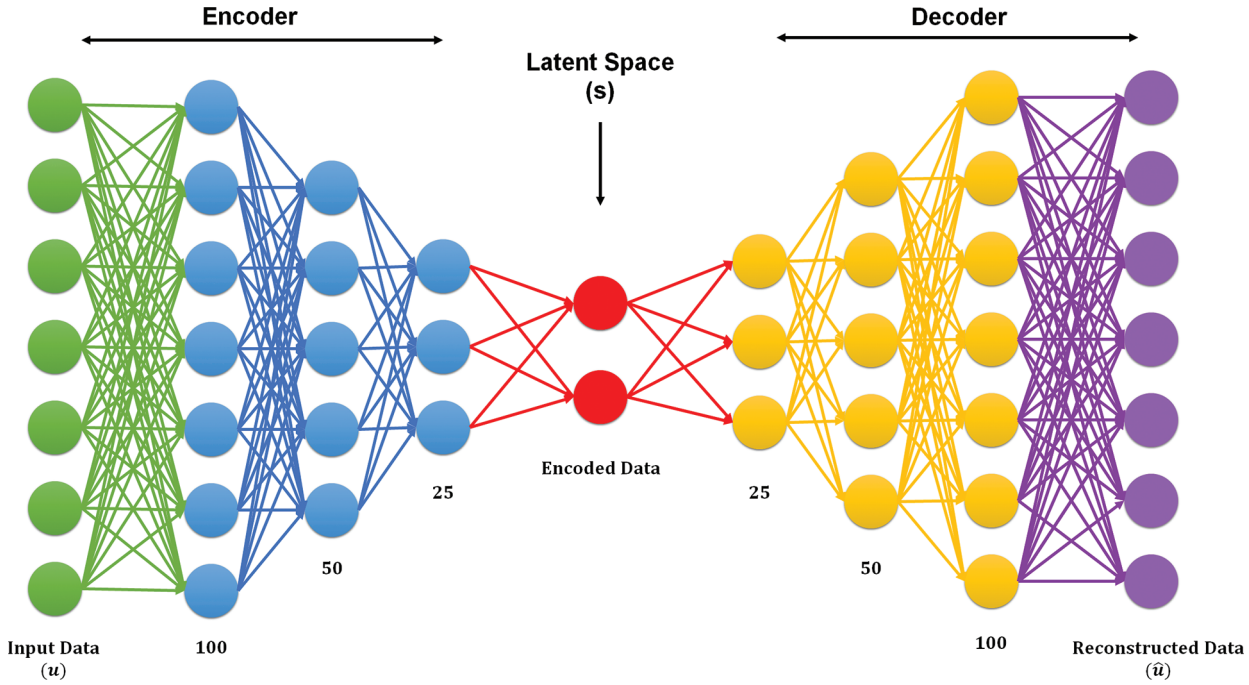

Figure 2: The proposed Deep Autoencoder (DAE) for cyberattack detection

The mean squared error (MSE) shown in Eq. (4) specifies the cost function of DAE.

$$MSE = \frac{1}{m}\sum_{y=1}^{m}\left\|\frac{1}{2}h_{W,b}\left(r^{(y)}\right) - s^{(y)}\right\|^2 \tag{4}$$

L2 regulation shown in Eq. (5) is added to the cost function that prevents overfitting by decreasing the weights $W_{zy}^{(l)}$ among neuron y in layer l and neuron z in layer $l+1$:

$$\Omega_{L2Reg} = \frac{1}{2}\sum_{l=1}^{L-1}\sum_{y=1}^{n}\sum_{z=1}^{k}\left(W_{yz}^{(l)}\right)^2 \tag{5}$$

Here L represents the total layers in a neural network. The other parameters n and k indicate the number of neurons in layers l and $l+1$ respectively.

Additionally, a sparsity regularization is added to a cost function as shown in Eq. (6). It penalizes $\hat{\delta}_j$ for deviating from δ using the Kullback-Leibler (KL) divergence [16]. KL is an indicator of the difference between two different distributions. This function will be zero if Eq. (2) is satisfied or can have a higher value if $\hat{\delta}_j$ diverges from δ. The minimization of this term enables the $\hat{\delta}_j$ to be close to δ. Here S_2 represents the hidden neurons within the encoder.

$$\Omega_{Sparsity} = \sum_{z=1}^{s_2} KL(\delta \left|\hat{\delta}_j\right|$$

$$\Omega_{Sparsity} = \sum_{z=1}^{s_2} \delta\, log\frac{\delta}{\hat{\delta}_z} + (1-\delta)\, log\frac{1-\delta}{1-\hat{\delta}_z} \tag{6}$$

The cost function consists of the sum of L_2 regularization, MSE, and a sparsity regularization term. Here φ and ϑ regulate the strength of L_2 regularization and sparsity respectively.

$$\xi_{Sparse}\ (W,\ b) = MSE + \varphi * \Omega_{SL2Reg} + \vartheta * \Omega_{Sparsity} \tag{7}$$

The proposed attack detection scheme performs the feature selection operation through DAE. It facilitates obtaining the most optimal features and removing the irrelevant features to reduce the computational complexity and increase the attack detection performance. In the proposed scheme mutual information (MI) is incorporated with optimal feature selection.

MI is a measure of the mutual dependency among two random variables. It describes the level of information of one random variable about another. In other words, it denotes the reduction in uncertainty of one random variable as a result of information about another. As stated in Eq. (8), MI is related to the concept of entropy ψ, which is the anticipated information content of a random variable R:

$$\psi(R) = -\sum\nolimits_i \mu(r_y) log P(r_y) \tag{8}$$

Here, μ represents the probability of occurrence of an event with index y. The entropy of two random variables R and S with values r_y and r_z can be defined as shown in Eq. (9)

$$\psi(R \mid S) = -\sum\nolimits_{y,\,z} \mu(r_y,\ \ s_y) log \frac{\mu(r_y,\ \ s_y)}{\mu(s_y)} \tag{9}$$

Here $(r_y,\ \ s_y)$ represents the joint probability distribution. Then, MI of two discrete variables R and S can be described as

$$\begin{aligned} I(R;\ S) &= \psi(R) - \psi(R|S) \\ &= \psi(R) - \psi(R) - \psi(R,\ S) \\ &= \sum\nolimits_{y,\,z} \mu(r_y,\ \ s_z) log \frac{\mu(r_y,\ \ s_y)}{\mu(r_y) P(s_y)} \end{aligned} \tag{10}$$

Here, $\psi(R,\ S)$ indicates the joint entropy. The bigger MI value reduces the uncertainty in a variable is lower value can increase uncertainty.

The proposed scheme classifies the data into two classes attack and normal using SVM. To perform this operation, a linear SVM with GD is used as the optimizer. Linear SVM is a supervised ML technique that is used to solve two-class binary classification problems. Several hyperplanes can split the classes, thus a technique for determining the optimal one is necessary. SVM seeks the best decision boundary by maximizing the margin between the boundary and the nearest data occurrences. Support vectors are the nearest data occurrences that establish the greatest margin.

Providing training data of n instances $(r_1,\ \ s_1),\ \ldots,\ (r_n,\ \ s_n)$, where s_i is the real class of input data r_y $(y = 1,\ \ldots,\ n\ )$ and either 1 or -1, the decision boundary is defined as

$$f(r_y) = W^T r_y + b = 0 \tag{11}$$

Here, w and b represent the weight vector and bias respectively.

To prevent data instances from lying on the wrong side, the following constraints are enforced for each y:

$$if\ s_y = 1, \quad w^T r_y + b \geq 1 \tag{12}$$

$$if\ s_y = -1, \quad w^T r_y + b \leq -1 \tag{13}$$

Eqs. (12) and (13) can be combined as

$$s_y\left(w^T r_y + b\right) \geq 1 \quad for\ all\ 1 \leq y \leq n \tag{14}$$

SVM may address non-linearly issues by employing the kernel technique, which translates the original information into higher dimensional space. One possible issue is that SVM may need a lengthy training period. Despite producing high-performance outcomes, SVM training periods are frequently excessively long in contrast to alternative classifiers. However, a linear variant of SVM was used in this work, which shortened training time while getting equivalent results.

SVM optimizes using hinge loss as its loss function. The hinge loss can be defined with an output $s_y = \pm 1$ as

$$\max\left(0,\ \ 1 - \ yf\left(r_y\right)\right) \tag{15}$$

$$c\left(r,\ \ s,\ \ f\left(r_y\right)\right) = 1 - s_y f\left(r_y\right) \tag{16}$$

$$c\left(r,\ s,\ \ f\left(r_y\right)\right) = \begin{cases} 0, & s_y f\left(r_y\right) \geq 1 \\ 1 - s_y f\left(r_y\right), & otherwise \end{cases} \tag{17}$$

The objective function $\varepsilon(w)$ presented in Eq. (18) is made up of two terms: the regularization term and the loss term. Due to the convexity of the hinge loss function, ML convex optimizers can be employed. The goal function should be minimized for optimization:

$$Minimize\ \varepsilon(w) = \frac{\varphi}{2}\|w^2\| + \frac{1}{n}\sum\nolimits_{y=1}^{n} \max\left(0,\ \ 1 - s_y f\left(r_y\right)\right) \tag{18}$$

GD uses repeated stages to update parameters in the gradient's direction. GD requires derivatives concerning b and w. However, because the hinge loss is not differentiable, the following sub-gradient should be utilized for w and $f\left(r_y\right)$:

$$\frac{\partial}{\partial W}\max\left(0,\ \ 1 - s_y f\left(r_y\right)\right) = \begin{cases} 0, & s_y f\left(r_y\right) \geq 1 \\ -s_y f\left(r_y\right) & otherwise \end{cases} \tag{19}$$

3.2.3 Performance Evaluation Metrics

To analyze the proposed DAE, several performance assessment metrics are defined. All of these parameters are described in the following.

a) Accuracy: It is the most commonly used performance indicator that presents a ratio of accurately predicted observations to the total number of observations.

$$Accuracy = \frac{T_{pos} + T_{neg}}{T_{pos} + F_{pos} + F_{neg} + T_{neg}}$$

b) Precision: It is defined as the proportion of accurately anticipated positive observations to total expected positive observations.

$$Precision = \frac{T_{pos}}{T_{pos} + F_{pos}}$$

c) Recall: It is the proportion of accurately predicted positive observations to all positive observations in the class.

$$Recall = \frac{T_{pos}}{T_{pos} + F_{neg}}$$

d) F1 Score: Averaging precision and recall yields this score. As a result, this score takes into consideration both false positives and false negatives. While F1 is not as intuitive as accuracy, it is sometimes more useful, especially when the class distribution is uneven.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

4 Implementation and Performance Analysis

This study explores the proposed framework's implementation details and evaluates the effectiveness of the proposed DAE through extensive experimentation.

4.1 Simulation Platform

The simulations and performance analysis of the suggested DAE are performed on a Dell Precision 7550 Data Science computer. This workstation contains an Intel Xeon W-10855M processor and 32 GB DDR4 2933 MHz ECC Memory. An NVIDIA Quadro RTX 5000 w/16 GB graphic card ensures the smooth operations of DAE. The main algorithm of the proposed DAE is written in "Anaconda Navigator" using Python script.

4.2 Selection of Hyperparameters

In all experiments, the main structure of the DAE is fixed. By conducting extensive experiments, we selected the optimal hyperparameters of the proposed model to ensure the best performance for all the datasets. The utilized hyperparameters are learning rate, batch size, no of epochs, and latent space. All the selected hyperparameters are depicted in Table 3.

Table 3: Utilized hyperparameters for training and performance evaluation

Hyperparameters	Datasets	
	NSL-KDD	UNSW-NB15
Learning rate	0.001, 0.01, 0.10	0.005, 0.075, 0.150
Batch size	32, 64, 128	64, 128, 256
No of epochs	100	100
Latent space	12	14

4.2.1 Learning Rate

This parameter defines how much the model should change in response to the expected error when the model weights are updated. The selection of learning rate is tricky since a number that is too little may result in a protracted training process that becomes stuck, while a value that is too large may result in learning an inefficient set of weights too rapidly or in an unstable training process.

4.2.2 Batch Size

The parameter demonstrates how many samples must be processed before the internal model parameters are updated.

4.2.3 No of Epochs

This parameter indicates how many times the learning algorithm will traverse the training dataset. Each epoch is an opportunity for each sample in the training dataset to change the internal model parameters.

4.2.4 Latent Space

Latent space is a compressed data format in which related data points are relatively close together. Latent space may be used to discover features of the data and to develop simpler representations of data for analysis.

4.3 Performance Analysis

All datasets have been split into the training and testing datasets with 70/30 percent respectively. The detailed distribution of all the datasets is already presented in Tables 1 and 2. In the following, we discuss the performance of the DAE for each dataset.

4.3.1 Performance Evaluation with NSL-KDD

The simulations for the NSL-KDD dataset have been conducted with a range of learning rates of 0.001, 0.01, and 0.10 on 32, 64, and 128 batch sizes. The latent space is fixed as 12 and all the simulations are executed for 100 epochs. The efficiency of the suggested DAE is evaluated for both binary class and multi-class scenarios using the NSL-KDD dataset.

a) Binary-class Performance Assessment: In the first batch of the experiments, the effectiveness of the suggested scheme was evaluated for the NSL-KDD dataset in a binary class scenario. Experiments were conducted by using three learning rates 0.001, 0.01, and 0.10 and batch sizes 32, 64, and 128. Performance scores of the proposed DAE for batch size 32 are presented in bar graphs in Fig. 3. The suggested scheme attained the highest accuracy of 98.86% at a learning rate of 0.001. In the second stage, all experiments were repeated for batch size 64 using the same learning rates. Performance scores of the proposed DAE for batch size 64 are presented in bar graphs in Fig. 4. The suggested DAE attained the highest accuracy of 98.68% at the learning rate of 0.002. In the third stage, all experiments were repeated for batch size 128 using the same learning rates. Performance scores of the proposed DAE for batch size 128 are presented in bar graphs in Fig. 5. The suggested scheme attained the highest accuracy of 98.51% at a learning rate of 0.001.

b) Multiclass Performance Evaluation: In the second batch of experiments, the effectiveness of the suggested scheme is evaluated for the NSL-KDD dataset in the multiclass classification scenario. Experiments are conducted by using three learning rates 0.001, 0.01, and 0.10 and batch sizes 32, 64, and 128. As previously explained for the binary class experiments, at the first stage a batch size of 32 was selected. Performance scores of the suggested DAE for the batch size of 32 are presented in bar graphs in Fig. 6. The suggested scheme attained the highest accuracy of 98.14%, at a learning rate of 0.001. In the second stage, all experiments were repeated for the batch size of 64 using the same learning rates. Performance scores of DAE for batch size 64 are presented in bar graphs of Fig. 7. The suggested DAE attained an accuracy of 98.08% at the learning rate of

0.001. In the third stage, all experiments were repeated for a batch size of 128 using the same learning rates. Performance scores of the suggested technique for batch size 128 are presented in bar graphs in Fig. 8. The suggested scheme attained the highest accuracy of 98.26% at a learning rate of 0.001.

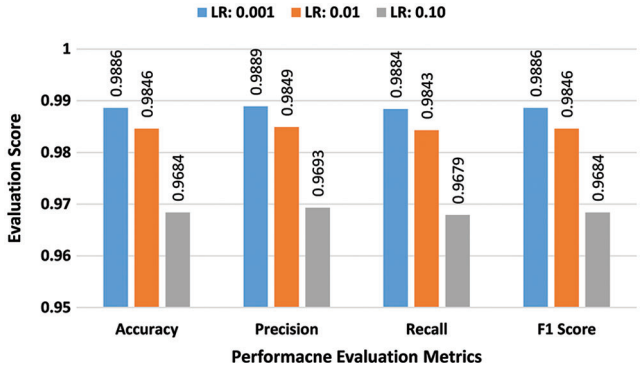

Figure 3: Binary class evaluation with NSL-KDD for batch size 32

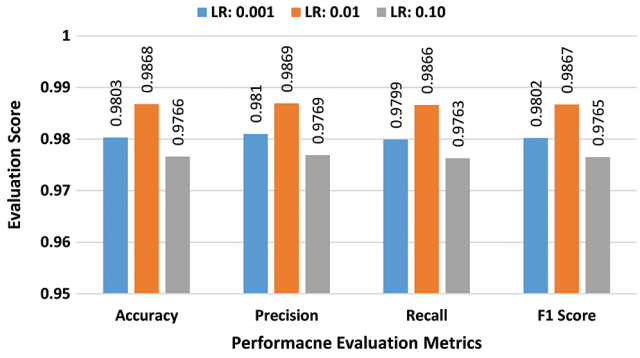

Figure 4: Binary class evaluation with NSL-KDD for batch size 64

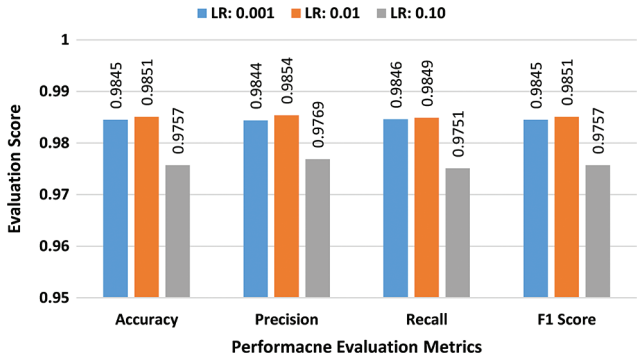

Figure 5: Binary class evaluation with NSL-KDD for batch size 64

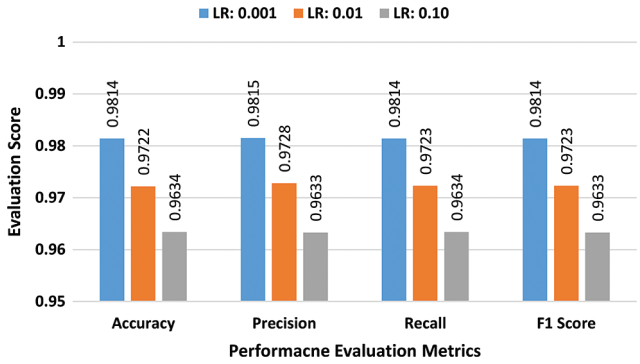

Figure 6: Multiclass evaluation with NSL-KDD for batch size 32

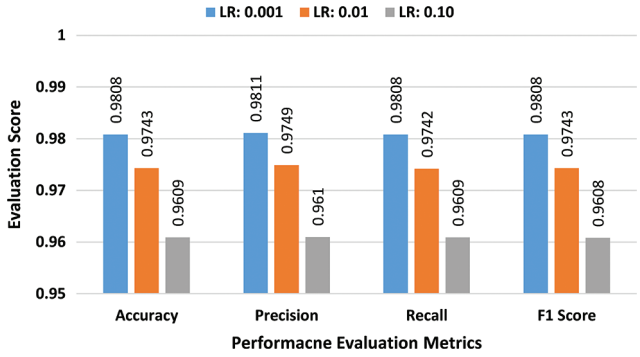

Figure 7: Multiclass evaluation with NSL-KDD for batch size 64

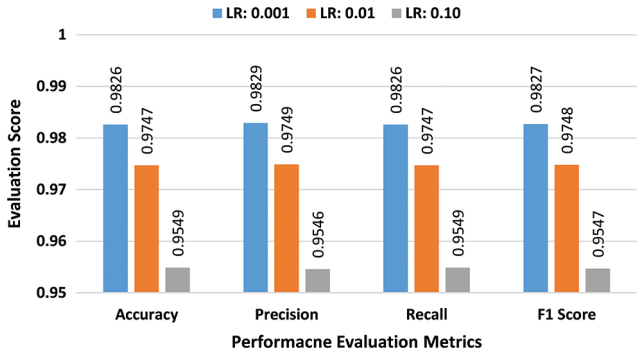

Figure 8: Multiclass evaluation with NSL-KDD for batch size 128

4.3.2 Performance Evaluation with UNSW-NB15

The simulations for the UNSW-NB15 dataset have been conducted with a range of learning rates of 0.005, 0.075, and 0.150 on 64, 128, and 256 batch sizes. The latent space is fixed as 14 and all the simulations are executed for 100 epochs. The effectiveness of the suggested scheme is analyzed in both binary class and multi-class scenarios using the UNSW-NB15 dataset.

a) Binary-class Performance Assessment: In the third batch of experiments, the performance of the suggested scheme is analyzed for the UNSW-NB15 dataset in the binary class scenario. Experiments were conducted by using three learning rates 0.005, 0.075, and 0.150, and batch sizes 64, 128, and 256. In the first stage of experiments, the performance of the proposed DAE

was evaluated for batch size 64. Performance scores for batch size 64 are presented in bar graphs in Fig. 9. The suggested scheme attained the highest accuracy of 99.07%, at a learning rate of 0.075. In the second stage, all the experiments were repeated for batch size 128 using the same learning rates. Performance scores of DAE for batch size 128 are presented in bar graphs in Fig. 10. The suggested scheme attained the highest accuracy of 99.32% at the learning rate of 0.075. In the third stage, all the experiments were repeated for batch size 256 using the same learning rates. Performance scores of DAE for batch size 256 are presented in bar graphs in Fig. 11. The suggested scheme attained the highest accuracy of 99.21% at the learning rate of 0.075.

b) *Multiclass Performance Evaluation:* In the fourth batch of experiments, the effectiveness of the suggested scheme is evaluated for the UNSW-NB15 dataset in a multiclass classification scenario. Experiments are conducted by using three learning rates 0.005, 0.075, and 0.150, and batch sizes 64, 128, and 256. As previously implemented for the binary class classification, at the first stage a batch size of 64 was selected. Performance scores of the proposed DAE for batch size 64 are presented in bar graphs in Fig. 12. The suggested scheme attained the highest accuracy of 98.82%, at a learning rate of 0.005. In the second stage, all the experiments were repeated for batch size 128 using the same learning rates. Performance scores of DAE for batch size 128 are presented in bar graphs in Fig. 13. The suggested DAE attained the highest accuracy of 98.79% at the learning rate of 0.005. In the third stage, all experiments were repeated for batch size 256 using the same learning rates. Performance scores of DAE for batch size 256 are presented in bar graphs in Fig. 14. The suggested scheme attained the highest accuracy of 98.66% at a learning rate of 0.005.

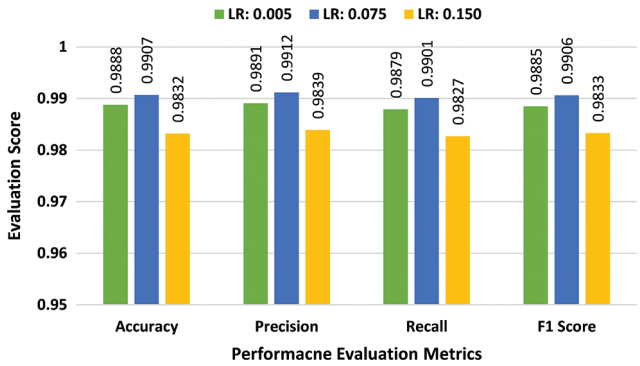

Figure 9: Binary class evaluation with UNSW-NB15 for batch size 64

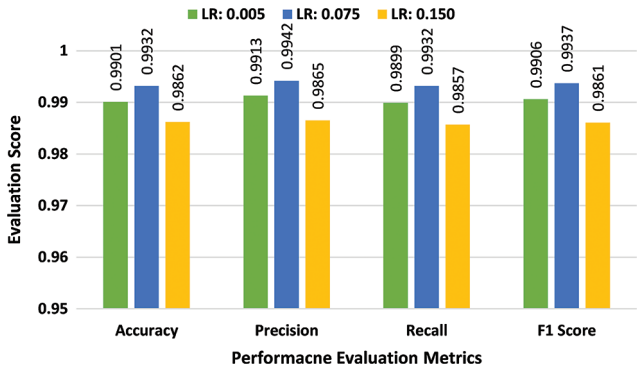

Figure 10: Binary class evaluation with UNSW-NB15 for batch size 128

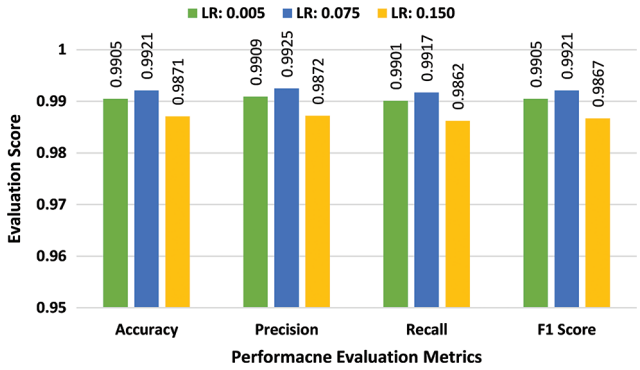

Figure 11: Binary class evaluation with UNSW-NB15 for batch size 256

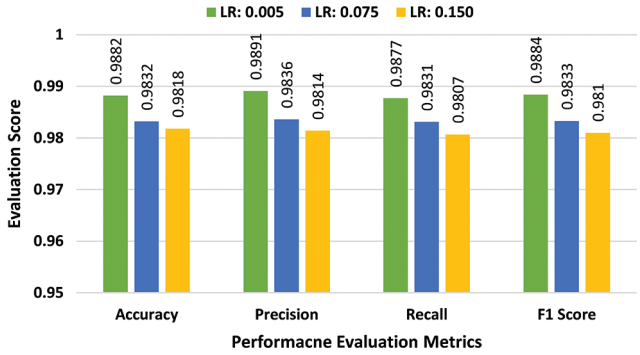

Figure 12: Multiclass evaluation with UNSW-NB15 for batch size 32

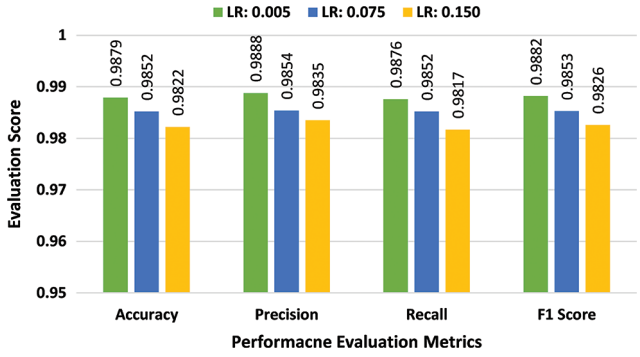

Figure 13: Multiclass evaluation with UNSW-NB15 for batch size 64

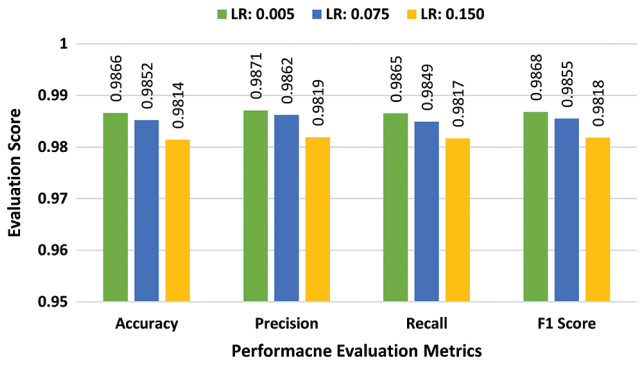

Figure 14: Multiclass evaluation with UNSW-NB15 for batch size 128

4.3.3 Performance Comparison with the State-of-the-art

To further investigate the efficiency and robustness of the proposed scheme, the performance is also compared to the related works. A brief performance comparison is presented in Table 4. This comparison is organized based on the utilized DL algorithm, datasets, hyperparameter selection, and attack detection accuracy. Most of the researchers utilized time-intensive deep learning algorithms which are not suitable for deployment in resource-constrained IoT networks. Second, only a few studies focused on the selection of suitable hyperparameters for the optimal training of their schemes. Third, most of the studies presented their evaluation for binary class scenarios and multiclass evaluation is missing. The performance of the proposed scheme is analyzed in both binary and multiclass scenarios and it attained higher attack detection accuracies as compared to the several state-of-the-art IDSs.

Table 4: Performance comparison with the state-of-the-art IDSs

Reference	Proposed scheme	Utilized dataset	Hyperparameters slection	Accuracy	
				Binary class	Multiclass
[5]	LSTM	N_BaIoT	No	97.74%	Not evaluated
[6]	NDAE	KDD Cup '99 and NSL-KDD	No	Not evaluated	97.85%, 80.58
[7]	DFFNN	NSL-KDD, UNSW-NB15	Yes	99.0%, 98.90%	93.64%, 91.22%
[8]	MV-FLID	MQTT dataset	No	98.0%	Not evaluated
[9]	SLFN	IoTID20	No	86.20%	Not evaluated
[10]	RSRT	SCADA dataset	Yes	96.78%	Not evaluated
[11]	B-MLSTM	CTU-13, gas-water, AWID	No	95.01%, 93.41% 97.58%	Not evaluated
Proposed scheme	DAE	NSL-KDD, UNSW-NB15	Yes	98.96%, 99.32%	98.26%, 98.82%

5 Conclusion

This article proposed a novel DAE-based framework for cyberattack detection in IoT networks. The most significant feature of the proposed design is its lower computational complexity which makes it resource efficient cybersecurity framework for IoT networks. To achieve the optimum performance of the DAE, a range of suitable hyperparameters were determined to train the neural network. These parameters include learning rate, batch size latent space, and no of epochs. Extensive experiments are conducted to analyze the efficacy of the suggested scheme using two standard security datasets including NSL-KDD, and UNSW-NB15. The performance was evaluated through several assessment parameters such as accuracy, precision, recall, and F1 score in both binary class and multiclass scenarios. Experimental results proved that the suggested scheme attained higher attack detection accuracy and other scores for both datasets. A single board computing platform can be incorporated as a hardware accelerator to improve the speed and performance of proposed attack detection for future endeavors.

Funding Statement: The Deanship of Scientific Research (DSR) at King Abdulaziz University (KAU), Jeddah, Saudi Arabia has funded this project, under Grant No. (IFPDP-279-22).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. P. Dharshini, D. Gopalakrishnan, C. K. Shankar and R. Ramya, "A survey on IoT applications in smart cities," *Immersive Technology in Smart Cities*, pp. 179–204, 2022.

[2] S. Latif, Z. Huma, S. S. Jamal, F. Ahmed, J. Ahmad *et al.,* "Intrusion detection framework for the internet of things using a dense random neural network," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6435–6444, 2022.

[3] A. Churcher, R. Ullah, J. Ahmad, S. U. Rehman, F. Masood *et al.,* "An experimental analysis of attack classification using machine learning in IoT networks," *Sensors*, vol. 21, no. 2, pp. 446–478, 2021.

[4] S. M. Tahsien, H. Karimipour and P. Spachos, "Machine learning based solutions for security of internet of things (IoT): A survey," *Journal of Network and Computer Applications*, vol. 161, pp. 102630–102651, 2020.

[5] G. D. L. T. Parra, P. Rad, K. K. R. Choo and N. Beebe, "Detecting internet of things attacks using distributed deep learning," *Journal of Network and Computer Applications*, vol. 163, pp. 102662–102675, 2020.

[6] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[7] J. B. Awotunde, C. Chakraborty and A. E. Adeniyi, "Intrusion detection in industrial internet of things network-based on deep learning model with rule-based feature selection," *Wireless Communications and Mobile Computing*, 2021.

[8] D. C. Attota, V. Mothukuri, R. M. Parizi and S. Pouriyeh, "An ensemble multi-view federated learning intrusion detection for iot," *IEEE Access*, vol. 9, pp. 117734–117745, 2021.

[9] R. Qaddoura, M. Al-Zoubi, H. Faris and I. Almomani, "A multi-layer classification approach for intrusion detection in iot networks based on deep learning," *Sensors*, vol. 21, no. 9, pp. 2987–3008, 2021.

[10] M. M. Hassan, A. Gumaei, S. Huda and A. Almogren, "Increasing the trustworthiness in the industrial IoT networks through a reliable cyberattack detection model," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6154–6162, 2020.

[11] X. Li, M. Xu, P. Vijayakumar, N. Kumar and X. Liu, "Detection of low-frequency and multi-stage attacks in industrial internet of things," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8820–8831, 2020.

[12] C. Luo, Z. Tan, G. Min, J. Gan, W. Shi *et al.,* "A novel web attack detection system for internet of things via ensemble classification," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5810–5818, 2020.

[13] T. Su, H. Sun, J. Zhu, S. Wang and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020.

[14] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey and R. T. Goswami, "An integrated rule based intrusion detection system: Analysis on UNSW-NB15 data set and the real time online dataset," *Cluster Computing*, vol. 23, pp. 1397–1418, 2020.

[15] J. Ahmad, S. A. Shah, S. Latif, F. Ahmed, Z. Zou *et al.,* "DRaNN_PSO: A deep random neural network with particle swarm optimization for intrusion detection in the industrial internet of things," *Journal of King Saud University-Computer and Information Sciences*, 2022.

[16] S. J. Lee, P. D. Yoo, A. T. Asyhari, Y. Jhi, L. Chermak *et al.,* "IMPACT: Impersonation attack detection via edge computing using deep autoencoder and feature abstraction," *IEEE Access*, vol. 8, pp. 65520–65529, 2020.