Computers, Materials & Continua
DOI:10.32604/cmc.2021.016074
Article

Tech Science Press

DeepIoT.IDS: Hybrid Deep Learning for Enhancing IoT Network Intrusion Detection

Ziadoon K. Maseer[1], Robiah Yusof[1], Salama A. Mostafa[2,*], Nazrulazhar Bahaman[1], Omar Musa[3] and Bander Ali Saleh Al-rimy[4]

[1]Faculty of Information and Communication Technology, Universiti Teknikal Malaysia, Melaka, 76100, Malaysia
[2]Center of Intelligent and Autonomous Systems, Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Johor, 86400, Malaysia
[3]Faculty of Business and Technology, UNITAR International University, Selangor, 47301, Malaysia
[4]School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Johor, 81310, Malaysia
[*]Corresponding Author: Salama A. Mostafa. Email: salama@uthm.edu.my
Received: 21 December 2020; Accepted: 07 May 2021

Abstract: With an increasing number of services connected to the internet, including cloud computing and Internet of Things (IoT) systems, the prevention of cyberattacks has become more challenging due to the high dimensionality of the network traffic data and access points. Recently, researchers have suggested deep learning (DL) algorithms to define intrusion features through training empirical data and learning anomaly patterns of attacks. However, due to the high dynamics and imbalanced nature of the data, the existing DL classifiers are not completely effective at distinguishing between abnormal and normal behavior line connections for modern networks. Therefore, it is important to design a self-adaptive model for an intrusion detection system (IDS) to improve the detection of attacks. Consequently, in this paper, a novel hybrid weighted deep belief network (HW-DBN) algorithm is proposed for building an efficient and reliable IDS (DeepIoT.IDS) model to detect existing and novel cyberattacks. The HW-DBN algorithm integrates an improved Gaussian–Bernoulli restricted Boltzmann machine (Deep GB-RBM) feature learning operator with a weighted deep neural networks (WDNN) classifier. The CICIDS2017 dataset is selected to evaluate the DeepIoT.IDS model as it contains multiple types of attacks, complex data patterns, noise values, and imbalanced classes. We have compared the performance of the DeepIoT.IDS model with three recent models. The results show the DeepIoT.IDS model outperforms the three other models by achieving a higher detection accuracy of 99.38% and 99.99% for web attack and bot attack scenarios, respectively. Furthermore, it can detect the occurrence of low-frequency attacks that are undetectable by other models.

Keywords: Cyberattacks; internet of things; intrusion detection system; deep learning neural network; supervised and unsupervised deep learning

1 Introduction

Many services are found online using the internet, and this trend is increasing over time. The internet helps millions of automotive Internet of Things (IoT) endpoints in providing provide services. The new technology replacing traditional networks is the IoT networks, such as those of industrial machines, smart energy grids, building automation, and many personal assistance devices [1,2], as shown in Fig. 1. An IoT represents a complex and dynamic network that connects endpoints of devices to provide various services. The diversity and volume of data transmitted over the networks raise many security issues. The risk and occurrences of attacks are increasing tremendously as a result of these revolutionary technologies [3]. Therefore, there is a need to build or develop an effective method to protect data confidentiality and network resource availability from zero-day attacks. Furthermore, it is essential to sustain the progress of these technologies [4]. Many organizations have limited network defense resources. They use traditional security methods and tools such as anti-spam, antivirus, firewalls, etc., to protect their networks. However, these methods and tools are still vulnerable to sophisticated and novel attacks [5].

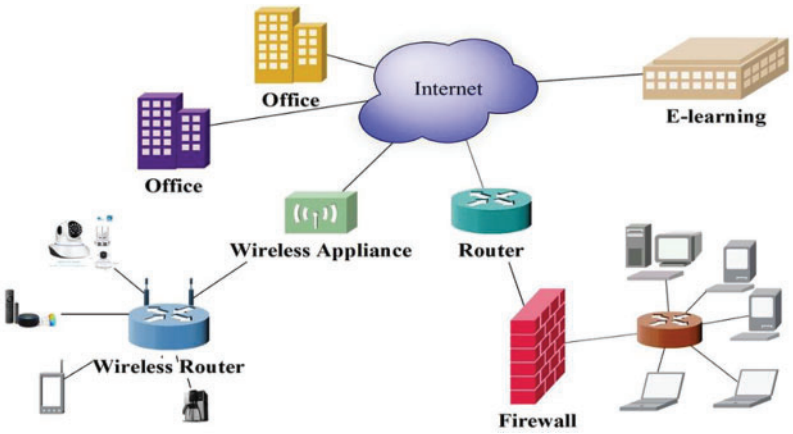


Figure 1: IoT infrastructure of a smart city

The 2020 Unit 42 IoT Threat Report is a survey of over 1.2 million IoT devices in the healthcare information technology industries in the United States. The report shows that these organizations are unsafe as their sensitive data is exposed to threats. The results of the survey show that 83% of devices used for medical imaging are running on outdated and unsecured operating systems [3]. The most common types of cyberattacks in such organizations are ransomware, botnet, password, and other attacks, as shown in Fig. 2.

Network intrusion detection systems (IDS) are essential tools for monitoring a network, tracking malicious activities, and identifying intrusions. They provide powerful defense systems against various threats and cyberattacks. IDS models are classified according to their detection mechanisms into anomaly-based and signature-based approaches [6]. The anomaly-based IDS models rely upon distinguishing abnormal behaviors from normal behaviors (deviations) to detect intrusions; these have been found to be effective against polymorphic or unknown attacks [7]. Signature-based IDS models rely on a set of pre-defined patterns of malicious activities and attacks. The large volume and diversity of network traffic as IoT devices, social media applications, and platforms continue to increase [8,9], meaning it is impractical to rely merely on identifying pre-defined attack patterns to detect intrusion of networks. The anomaly-based IDS

models are mainly utilized for identifying unknown attacks, such as when pre-defined patterns are absent for IoT networks [10]. It has recently become clear that the most powerful feature of deep learning (DL) algorithms is the extraction of significant features from input data to enhance the learning of classification models in different computer science domains [11]. Consequently, DL algorithms are proposed for the IDS models to predict attacks; these algorithms outperform conventional machine learning (ML) algorithms to detect and identify attacks. For instance, Tama et al. [12] used web attacks to measure anomaly IDS model effectiveness based on ML classifiers in protecting IoT and web applications. Web attacks such as Structured Query Language (SQL) injection and cross-site scripting (XSS) cause data loss, and these attacks are not found in KDD Cup 9 and UNSW-NB15 datasets. In addition to the state-of-the-art methods, supervised learning (SL) entails an enormous amount of labeled attacks to maintain accurate performance, which is unaffordable in many real-world attack cases.

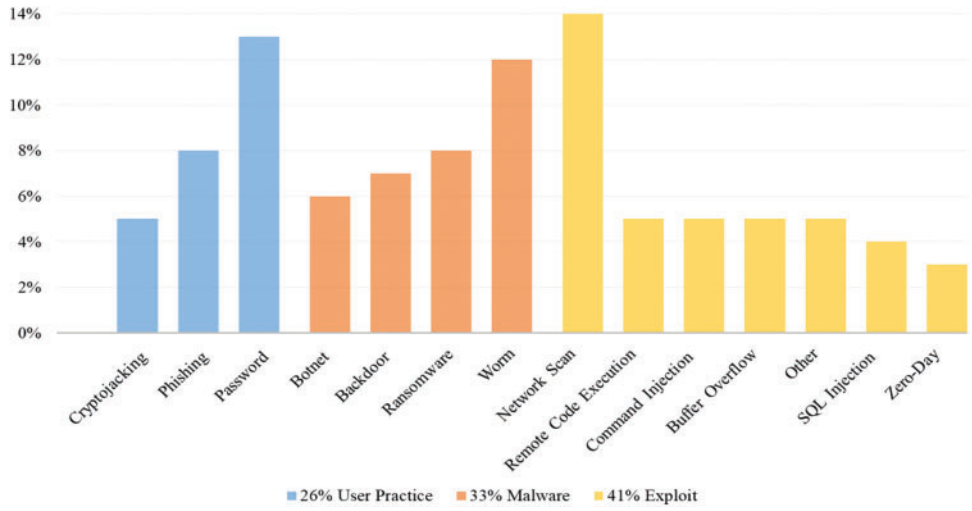


Figure 2: 2020 unit 42 IoT threat report

One of the outstanding algorithms that researchers propose for building IDS models is the restricted Boltzmann machine (RBM) algorithm [13]. This algorithm's key success is the ability to identify new patterns of attack that have never before been witnessed [14,15]. A deep belief network (DBN) algorithm consists of two phases: the pre-training phase that utilizes RBMs to generate high-level features from the network traffic and the classification phase that consists of two hidden backpropagation layers to identify attacks. However, the performance of IDS using the conventional DBN algorithm still faces several challenges in detecting attacks. The first challenge is the complexity of data traffic in the network traffic, which has a high dimensionality of features, noise, and corrupted data, making it difficult for the DBN algorithm to find correlations between them in a realistic dataset. The second challenge is an imbalanced-class issue in which less frequent attacks are invisible within the training phase. It subsequently causes overfitting towards the higher frequent classes. As a result, a flawed anomaly-based IDS model would not detect skewed attacks and would have a high false alarm rate [16,17]. Finally, a reliable evaluation procedure is a challenge in which the major studies have used only the accuracy rate to evaluate anomaly-based IDS models' performance. Relying solely on the accuracy rate is inefficient at showing an

anomaly-based IDS model's effectiveness in detecting attacks. For example, a reliable evaluation may consider the percentage of true positive and true negative alarms for each class [18–20].

Consequently, this work proposes an efficient and reliable IoT-network intrusion detection (DeepIoT.IDS) model based on a novel hybrid weighted deep belief network (HW-DBN) algorithm. The HW-DBN algorithm consists of a deep Gaussian–Bernoulli type of RBM (deep GB-RBM) and weighted deep neural network (WDNN) algorithms. The HW-DBN algorithm comprises an initial unsupervised stage and is responsible for generating traffic features; then, the final supervised stage is responsible for classifying the traffic into normal or abnormal classes. We stack a series of deep GB-RBMs to construct feature learning to perform learning on the original network traffic data in the pre-training stage. Next, we use the feature weighted WDNN to optimize deep features and remove noise and redundant features, thereby improving the classification performance. The main contributions of this paper to the cybersecurity domain are as follows:

(i) Enhancing an unsupervised DL (UDL) deep GB-RBM based on the GB-RBM algorithm to improve the extraction of network data features.
(ii) Proposing a novel HW-DBN algorithm based on a DBN algorithm by integrating the deep GB-RBM with the WDNN to form an improved and continuous prediction hybrid DL (HDL) algorithm. The WDNN algorithm learns high-dimensional unlabeled data along with standard labeled data.
(iii) Modeling an efficient and reliable IoT-network intrusion detection system (DeepIoT.IDS) based on a proposed HW-DBN algorithm for anomaly-based IDS.
(iv) Demonstrating that the DeepIoT.IDS model is efficient and reliable in detecting sophisticated attacks by testing it using the CICIDS2017 dataset. The CICIDS2017 manifests a benchmark dataset for complex IoT environments.

This paper is organized into six sections in which the related work of different types of ML and DL neural networks is presented in Section 2. The research background of RBM and DBN algorithms is illustrated in Section 3. The methods of this work are described in Section 4, including the improved RBM algorithm, the WDNN algorithm, the proposed HW-DBN algorithm, and the DeepIoT.IDS model. The implementation of the proposed DeepIoT.IDS model, the evaluation criteria, and the testing environment are used to check and evaluate the performance of the proposed model, and these are scrutinized in Section 5. Lastly, the conclusion and future work directions of this research are highlighted in Section 6.

2 Related Work

The anomaly-based IDS epitomizes one of the best methods to detect significant attacks in a network. The IDS uses the ML algorithms as one of its essential components for performing classification tasks. Examples of these IDS models include artificial neural networks (ANN), decision trees (DT), expectation-maximization (EM), k-nearest neighbors (KNN), linear regression (LR), naïve Bayes (NB), random forests (RF), logistic regression (LR), and support vector machine (SVM) [21]. Researchers evaluate the ML-IDS models' performance using different parameters to define these models' capability, including true positive and true negative rates. For instance, Janarthanan and Zargari [22] used feature selection and an RF algorithm in IDS to detect and classify intrusions and attacks on the UNSW-NB15 dataset. The RF algorithm achieved a false alarm rate (FAR) of 4.4% and an accuracy of 81.61%. Nawir et al. [23] used an averaged one-dependence estimator (AODE) algorithm to propose a multi-class classification model. They tested

this model on the UNSW-NB15 dataset and reported a FAR of 6.57% and an accuracy of 83.47%. However, for the current IDS, it is clear that relying only on traditional machine ML algorithms to detect anomalies of advanced attacks that have unexpected patterns is no longer enough. Therefore, there is a need for strong methods to protect IoT networks from different types of intrusion.

Many studies have attempted to analyze and compare anomaly-based ML-and DL-IDS models [24,25]. Most of the studies conclude that the DL is better for detecting attacks and reducing false alarms. For instance, Khan et al. [26] introduced self-taught learning (STL) in a DL-IDS to exploit unlabeled data. They applied the model to the NSL-KDD dataset, and the experimental results show that DL-IDS outperforms previous studies. Similarly, Yin et al. [27] proposed a DL recurrent neural network (RNN) algorithm for IDS. They applied the DL-RNN algorithm to the NSL-KDD dataset. The results showed that the algorithm outperformed several ML algorithms. A DL approach can be categorized into supervised DL (SDL), unsupervised DL (UDL), and hybrid DL (HDL) learning algorithms [28]. The most powerful learning approach is SDL, but in IDS, it exposes misclassifying anomalies due to a lack of prior knowledge or less represented features of novel attacks, a low convergence speed, and the ease of falling into a local optimum [29]. The studies of Ghanem [30] and Hajisalem et al. [31] used metaheuristic techniques to improve SDL as the DNN algorithm; however, the anomaly-based IDS model requires more time and resources for training. In contrast, the UDL approach includes different algorithms, such as restricted Boltzmann machines (RBMs), deep belief networks (DBNs), and auto-encoders. These algorithms have the self-training ability from unlabeled raw data to accomplish different tasks and provide less presented data to be classified [32]. The HDL approach represents the recent trend in building advanced IDS models as it has variations of UDL and SDL training algorithms. The UDL is used for feature learning or detection to detect the significant features, and SDL is used to build an anomaly-based IDS model based on extracted features. An autoencoder algorithm (AE) is a particular type of feedforward neural network and UL approach; its input data is equal to the output. The AE compresses the input into a lower-dimensional code and then constructs the output [33]. The RBM algorithm is considered one of the best algorithms for online detection and classification of new data samples [34]. The RBM differs from other autoencoders by the two biases in the visible and hidden layers. The first bias helps the RBM learn the reconstructions on the backward pass. The second bias helps the RBM to yield activations on the forward pass.

Fiore et al. [35] presented a discriminative RBM (DRBM) algorithm to compress the features that are not in the network packet payloads and produce new features. The IDS provides new features for a soft-max binary classification algorithm to train and predict normal and DoS attacks. Similarly, Erfani et al. [36] proposed combining one-class linear SVM with DBNs to construct an intrusion detection model. They applied the model on several benchmark datasets, and the results showed improvement in the IDS performance. Alrawashdeh et al. [37] proposed a DL combination of RBMs and LR algorithms in constructing a DBN for an IDS. This combination produced a DBN architecture of four hidden layers for feature reduction and one hidden layer for classification. They included a fine-tuning phase to update the DBN weights while training the network and prior to the classification task made by the LR. They implemented this type of DBN on the KDD99 dataset and attained a FAR of 0.5% and an accuracy of 97.9%. As with those studies, a conventional DBN algorithm is proposed for constructing the detection model. The learning process used in the DBN algorithm is a traditional gradient-based process using update rules, which are obtained by taking the partial derivative of the log-likelihood function to adjust the weights of a model and build an anomaly-based IDS model. This learning

represents the traditional way to train the DBN algorithm, which is not efficient in real traffic networks with a variety of values. Moreover, testing and validation are unreliable and do not yet reflect an anomaly-based IDS's actual performance because the dataset includes outdated attack scenarios, as with KDD99 [38].

In a nutshell, most DL/ML algorithms are proposed as anomaly-based IDS models, which discover anomalies by examining data patterns to distinguish data lines and determine normal and abnormal activities. However, some of the existing studies have lacked reliable and efficient validation as they use low-quality datasets. The datasets might suffer from several weaknesses and deficiencies, including outdated attack scenarios and versions that may not be a perfect representative of the existing network attacks, ignoring the complexity of IoT-like networks. Moreover, the conventional HDL algorithms do not provide sufficient training ability to detect modern attacks in realistic datasets. In contrast with those works, we propose a new model for intrusion detection called DeepIoT.IDS. The IDS model is based on a deep GB-RBM algorithm as feature learning of our raw data and then combined with a weighted DNN (WDNN) algorithm for classification to construct a novel HW-DBN algorithm. The HW-DBN algorithm is based on an improved RBMs (Deep GB-RBM) algorithm that is combined with a Weighted DNN (WDNN) algorithm to construct a novel DeepIoT.IDS model.

3 Research Background

3.1 Bernoulli–Bernoulli Restricted Boltzmann Machine (BB-RBM) Algorithm

The basic RBM algorithm represents a neural network with directional connections of multiple hidden layers. A basic structure of the RBM is the v_i, which represents the visible node of the input layer, and h_j, which represents the hidden node of the output layer. Meanwhile, a_i denotes the biases of the input layer, b_j denotes the biases of the output layer, and W_{ij} denotes the weights between the nodes of the two layers. The RBM increases its generation probability by adjusting the training weights between the nodes, as shown in Fig. 3a. A deep restricted Boltzmann machine (deep RBM) consists of several RBMs in which each RBM output is fed to the following RBM, as shown in Fig. 3b [39]. The energy function is defined as:

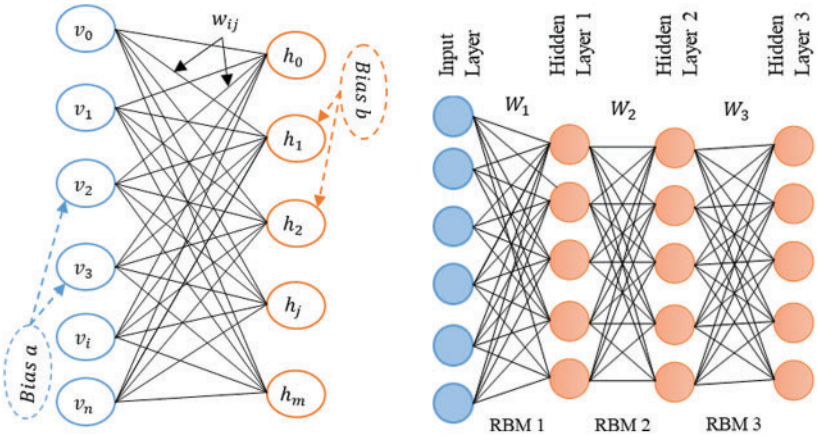
$$E(v, h) = -\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}h_j v_i - \sum_{i=1}^{n} v_i b_i - \sum_{j=1}^{m} c_j h_j \tag{1}$$

where m is the number of visible nodes, n is the number of hidden nodes, and the values' range of visible and hidden nodes is between $\{0, 1\}$. The conditional probabilities $p(h_i = 1 \mid v)$ and $p(v_i = 1 \mid h)$, are:

$$p(h_i = 1 \mid v) = \sigma\left(a_i + \sum_{i}^{m} w_{ij} v_i\right) \tag{2}$$

Eq. (2) shows the probabilities of a hidden layer using sigmoid function σ and the probabilities of a visible layer using sigmoid function σ, as defined in Eq. (3):

$$p(v_i = 1 \mid h) = \sigma\left(b_i + \sum_{j=1}^{n} h_j w_{ij}\right) \tag{3}$$



(a) (b)

Figure 3: The improvement of deep RBMs (a) Basic structure of RBM (b) Deep RBMs

We can use the Gibbs sampling algorithm [40] to sample the nodes of hidden and visible layers; each iteration of Gibbs sampling consists of two steps: (1) sample a new hidden vector $h(h)$ based on the conditional probability $p(h \mid \boldsymbol{v}(t-1))$ and (2) sample a new visible vector $v(t)$ based on the conditional probability $p(v \mid h(t))$. The log-likelihood loss is used to train the RBM algorithm, and the derivative with respect to $\theta = \{W, b, c\}$ takes the following form:

$$\frac{\partial}{\partial w_{ij}} logl(\theta \mid v) = p(h_i = 1 \mid v)\, v_i - \sum_v p(v)\, p(h_i = 1 \mid v)\, v_i \tag{4}$$

Similarly, the log-likelihood derivatives for the bias b_i of the i_{th} visible node and the bias c_j of the j_{th} hidden node are, respectively, given by:

$$\frac{\partial}{\partial b_i} \log l(\theta \mid v) = v_i - \sum p(v)\, v_i \tag{5}$$

$$\frac{\partial}{\partial c_i} \log l(\theta \mid v) = p\left(h_j = 1 \mid v\right) - \sum p(v)\, p\left(h_j = 1 \mid v\right) \tag{6}$$

The training of the RBM algorithm differs from the training of regular neural networks. The learning process used a contrastive divergence (CD) algorithm [41] to estimate the log-likelihood gradient derivatives of an RBM using Gibbs sampling. In CD learning, a Markov chain is run only for k steps (typically, $k = 1, 2, \ldots, 10$). The Gibbs chain is initialized with one of the training vectors, $\boldsymbol{v}(0)$. During each Gibbs sampling step t, $\boldsymbol{h}(\boldsymbol{t})$ is sampled from the conditional probability $p(h \mid v(t-1))$, followed by the sampling of $\boldsymbol{v}(\boldsymbol{t})$ based on $p(v \mid h(t))$. The k steps of Gibbs sampling yield $v(k)$ and $h(k)$, which are subsequently used for estimating the log-likelihood derivatives.

3.2 Conventional Deep Belief Network (DBN) Algorithm

A deep belief network (DBN) algorithm is a greedy layer-wise technique that learns features from unlabeled data in a UL manner [42]. The DBN consists of a stack of RBMs; any RBM's output forms the next RBM input. It uses a small labeled dataset to associate patterns that the

RBMs learn to form classes. The pre-training of the DBN provides a way to develop neural networks while only training shallow networks. In cybersecurity, DBN [43] can be used for classification tasks by adding backpropagated two layers classifier to the end of RBMs to classify learned features, as shown in Fig. 4a. The training process of the DBN is the same as the learning process of the RBM, in which the layers of the DBN are learning individually until they reach optimal parameters, then start extracting important features. Lastly, the backpropagated two layers are trained based on significant features in the SL manner.

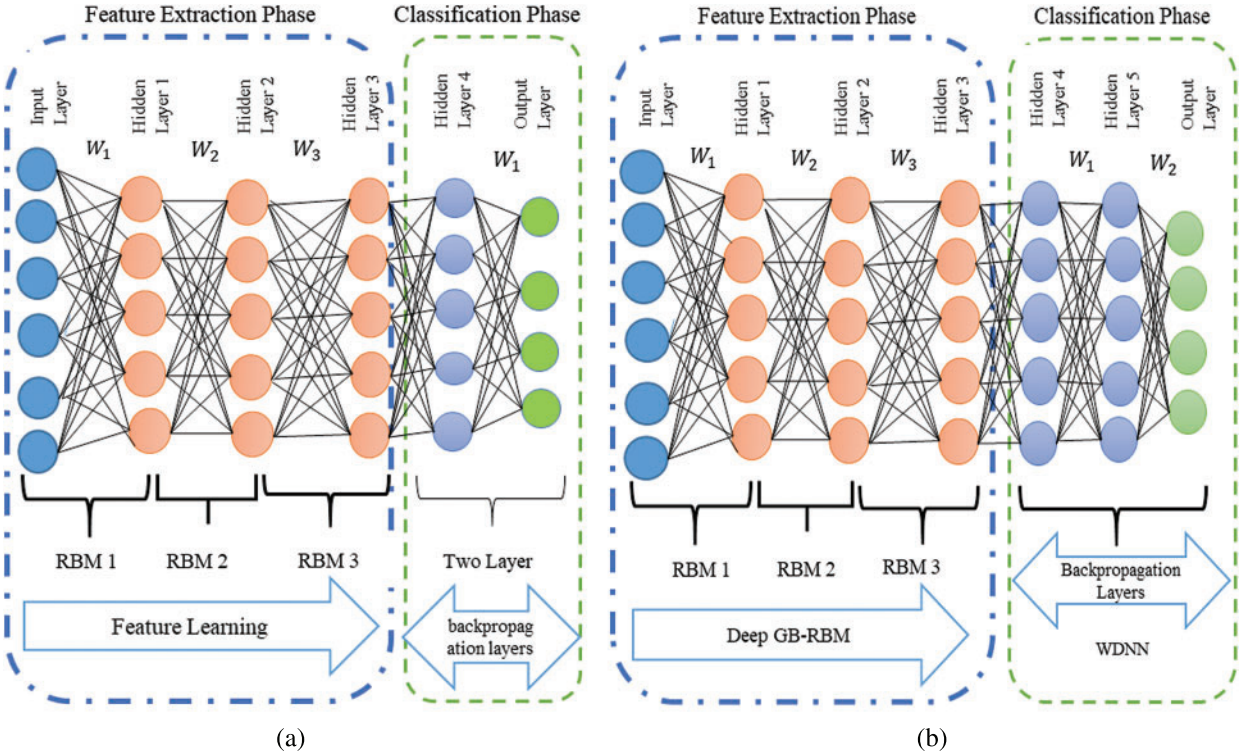


Figure 4: The DBN *vs*. the HW-DBN (a) CDBN algorithm (b) HW-DBN algorithm

4 Research Methods

This section describes the main components of this work and presents the theoretical concepts behind the novel DeepIoT.IDS model, which detects intrusions by classifying network states into normal and several abnormal types of attack. This paper proposes a few improvements to the conventional training methods for CDBN to overcome the existing difficulties, based on the GB-RBM algorithm for extracting important features in the pre-training stage and DNN for the classification task.

4.1 Deep Gaussian–Bernoulli Restricted Boltzmann Machine (GB-RBM) Algorithm

Different types of RBM algorithms have a similar structure but different parameters, such as the Gaussian–Bernoulli RBM (GB-RBM) algorithm. RBM defines each neuron's state as binary numbers, limiting their application in domain cybersecurity. One common approach to address this difficulty is to substitute the visible binary neurons with the Gaussian ones. The GB-RBM

has several real values of the visible nodes v_m and binary values of the hidden nodes h_n [36,44]. We define the deep GB-RBM based on the same idea of the BB-RBM algorithm by using the energy function as follows:

$$E(v, h) = -\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} h_j \frac{v_i}{\sigma_i} - \sum_{i=1}^{m} \frac{(v_i - b_i)}{2\sigma^2} - \sum_{j=1}^{n} c_j h_j \tag{7}$$

where in (7), σ^2 is the variance associated with the Gaussian visible unit, a_i and b_i represent the biases of the visible and hidden nodes, w_{ij} represents the weights that connect the visible and hidden nodes, σ_i represents the standard deviation that is associated with the Gaussian visible nodes v_i, and m and n specify the number of visible and hidden nodes, respectively. The conditional probability of the nodes is as below:

$$p(v_i = v \mid h) = N\left(v \mid b_i + \sum_j h_j w_{ij}, \sigma_i^2\right) \tag{8}$$

where $N(x \mid \mu, \sigma 2)$ is a probability density of normal distribution with a mean μ and a standard deviation σ, and:

$$p(h_i = 1 \mid v) = \sigma\left(a_i + \sum_i w_{ij} \frac{v_i}{\sigma_i^2}\right) \tag{9}$$

where $\sigma(x)$ is the logistic sigmoid function, i.e., $\sigma(x) = 1/(1 + e^{-x})$.

The free energy $f(v)$ is an energy function that assigns low energies to the variables with correct values and higher energies to the remaining variables with incorrect values. The $f(v)$ of an RBM with binary units further simplifies to:

$$f(v) = -\grave{b}v - \sum_i log\left(1 + e^{(b_i + W_i v)}\right) \tag{10}$$

The improved RBMs are trained based on the contrastive divergence (CD) algorithm. The main aim of the CD algorithm is to find optimal parameters θ according to the procedures of Gibbs sampling, loss function, weights modification, and new weights calculation as follows:

Gibbs sampling: The first part of the training is called Gibbs sampling. Given an input vector v, we use $p(h \mid v)$ for the prediction of the hidden values h. Knowing the hidden values, we use $(v \mid h)$:

$$h_i \approx p(h_i = 1 \mid v) = \sigma\left(a_i + \sum_i w_{ij} \frac{v_i}{\sigma_i^2}\right) \tag{11}$$

This process is repeated for k iterations to predict new input values v. Subsequently, new input values v_k are obtained from the original input values v_o.

$$v_i \approx p(v_i = v \mid h) = N\left(v \mid b_i + \sum_j h_j w_{ij}, \sigma_i^2\right) \tag{12}$$

Loss function: We compute the loss function using mean squared error by squaring the difference of distribution probability of data $<\blacksquare>_d$ and expecting the distribution probability of the model $<\blacksquare>_m$ as in the following Eq. (12):

$$l_i = \frac{1}{n}\sum_{i=1}^{n}(\langle\blacksquare\rangle_d - \langle\blacksquare\rangle_m)^2 \tag{13}$$

Gradient descent computing: Using the updated matrix of θ, we calculate the new θ with Adam's algorithm given by Adam (θ, v, m, $lr(\eta)$, t), where v and m are the variance and mean of the last step of the parameters, respectively. We assume the constants, which are $beta1 = 0.9$, $beta2 = 0.999$, $eps\_stable\,(\varepsilon) = 1e-8$, and $\eta = 0.0001$. Then we have gradients (g) of weights and biases and $g = \frac{\partial p(x)}{\partial\theta} \approx \frac{\partial f\,(x)}{\partial\theta} - \frac{\partial f\,(\hat{x})}{\partial\theta}$, $m_t = \beta_1 m_{t-1} + (1-\beta_1)\times g_t$ and $v_t = \beta_2 v_{t-1} + (1-\beta_2)\times g_t^2$.

From the above, we can calculate the $\hat{m}_t$ and $\hat{v}_t$ by using $\hat{m}_t = \frac{v_t}{(1.-beta1^t)}$ and $\hat{v}_t = \frac{m_t}{1.-beta2^t t}$. Finally, we can update the parameters θ based on the calculated moving averages with a step size η:

$$\theta_i = \theta_{i-1} - \eta\frac{\widehat{m}_i}{\sqrt[2]{\widehat{v}_i}+\varepsilon} \tag{14}$$

In general, learning in RBMs involves minimizing the mean square error given by:

$$\Delta w_{ij} = \frac{\partial l_i}{\partial W_{ij}}\left(\frac{1}{N}(e<v_i, h_i>_d - e<v_i, h_i>_m)^2\right) \tag{15}$$

and

$$\Delta a_{ij} = \frac{\partial l_i}{\partial a_j}\left(\frac{1}{N}\left(<h_j>_d - <h_j>_m\right)^2\right) \tag{16}$$

and

$$\Delta b_{ij} = \frac{\partial l_i}{\partial b_j}\left(\frac{1}{N}(<b_i>_d - <b_i>_m)^2\right) \tag{17}$$

The $<\blacksquare>_{\boldsymbol{d}}$ denotes the expected values of the data and $<\blacksquare>_{\boldsymbol{m}}$ denotes the expected values of the model. The expectation for the binary data is performed by using Eq. (10), whereas the expectation for the expected data is performed by using Eq. (11). Eq. (12) calculates the error, and Eqs. (14)–(16) minimize the error using gradient descent.

New weights calculation: The Adam algorithm calculates the new weights, given by Eq. (17) using the updated matrix:

$$\theta_{new} = \theta_{old} - \eta\frac{\widehat{m}_i}{\sqrt[2]{\widehat{v}_i}+\varepsilon} \tag{18}$$

We implement many steps with the above forms to get the optimal parameters $\theta = \{W, a, b\}$; this minimizes the loss to the lowest point and ensures that the cost function is at the optimal value (minima).

4.2 Weighted Deep Neural Network (WDNN) Algorithm

A weighted deep neural network (WDNN) is made up of densely connected or fully connected neural layers based on a cross-entropy loss function [45]. Specifically, weight cross-entropy loss (WCEL) is used for solving imbalanced data in cybersecurity. It can distribute labels for enhancing the training of the classification task. The structure of a WDNN consists of three layers. The first layer is training data, which is extracted from the last layer of the RBM. The second layer is a hidden layer with a rectified linear activation (ReLU) function and 78 neurons. Lastly, the output layer has neurons that are equal to the labels of the dataset and softmax activation function. The loss is calculated by using the WCEL function, and Eqs. (18) and (19) are used to calculate the weights w_k for each label of training data. Generally, the weights of positive classes are higher than negative classes. During the training of the model, the parameters (w, b) are updated for the low-frequency classes more than the high-frequency classes. It is mathematically represented as:

$$w_{k=0} = \frac{\left(\frac{1}{len(normal)}\right) \times (len(data))}{K} \tag{19}$$

and

$$w_{k=1}^{K} = \frac{\left(\frac{1}{len(positive)}\right) \times len\,(data)}{K} \tag{20}$$

where $Z_j = xW + b_i$, $Y_j = ReLU(Z_j)$ and $P(y = j \mid x) = softmax(Y)$.

Then weight-categorical cross-entropy loss is calculated using Eq. (21):

$$loss_{weight} = -\frac{1}{M} \sum_{k=1}^{K} \sum_{m=1}^{M} w_k \times y_m^k \times log\,(P\,(\gamma)) \tag{21}$$

where M represents the number of training examples, K represents the number of classes, w_k represents the weight for a class k, y_m^k represents the target label for training example m for the class k, x_m represents the input for training example m, and h_θ represents the model with neural network weights.

4.3 Hybrid Weighted Deep Belief Network (HW-DBN) Algorithm

The conventional DBN has a multi-layer RBM for feature extraction and a backpropagation SL algorithm for classification, as shown in Fig. 4a. Subsequently, we propose a novel hybrid weighted deep belief network (HW-DBN) algorithm for building an anomaly-based IDS model. The HW-DBN integrates the deep GB-RBM algorithm as a UDL approach with the WDNN classifier that has a backpropagation algorithm as an SDL approach.

The HW-DBN has the advantage of classifying based on extracted features fed from the proposed deep GB-RBM algorithm. We pre-train the first and second layers of the HW-DBN algorithm during the pre-training phase and then extract the features from the last layer and feed them into the training algorithm at the classification stage of the WDNN to further minimize

the prediction error. The algorithm HW-DBN comprises three hidden layers of the deep GB-RBM for the UDL technique with two hidden layers of WDNN backpropagation for the SDL technique at the other end, as shown in Fig. 4b. Hence, there are two core training techniques performed in the training phase of the HW-DBN algorithm. Firstly, we pre-train, individually, each layer of the deep GB-RBM algorithm using UDL and CD algorithms. We then use the initial stage's extracted features as an input feature of the SDL technique for the training process of the classification phase. In the subsequent decision stage, we collectively train the two layers of the WDNN algorithm. Then the HW-DBN algorithm employs the WDNN algorithm to decide on normal and multi-class attack traffic. Algorithm 1 outlines the main steps of the HW-DBN algorithm of the DeepIoT.IDS model:

Algorithm 1: The HW-DBN algorithm

Construct GB-RBM;
Set input Data: S_i is a sample of Features (X) = $\{x_1, x_2, \ldots, x_n\}$; label: Y; Epoch = N; threshold = 0.99999; Bach size (m) is samples number of data;
WHILE condition ≠ threshold **DO**:
    WHILE ($len(X)$) **Do**:
        FOR $(l_i) \in$ X **DO** UDL *to each layer* (i) :
            FOR $S^n_{i=0}$ to X_m **DO** training:
                Compute hidden units(h_i) as in Eq. (10);
                Compute visible units(v_i) as in Eq. (11);
                Compute loss function(err_i) as in Eq. (12);
                Compute gradient to the parameters:
                Calculate Δw_{ij} as in Eq. (14); calculate Δb_{ij} as in Eq. (15);
                Calculate Δc_{ij} as in Eq. (16);
                Update the parameters(θ) using Adam algorithm:
                w_{new}; b_{new}; c_{new}; as in Eq. (17)
            END-FOR
        END-FOR
    END-WHILE /*****Deep GB-RBM*****/
    X_{new} = Extracted features using Deep GB-RBM;
    FOR $(S_m) \in X_{new}$ **DO** *SL to each layer* (i):
        FOR $S^n_{i=0}$ to S_m **DO** backpropagation training:
            $Y_{predicted} = DNN\,(S_n)$;
            $loss_{weight}$ = weight cross-entropy loss ($Y_{predicted}$) as in Eq. (20);
            Compute gradient using Nadam algorithm to $loss_{weight}$;
            Update the parameters(θ):
            $w_{new} = w_{old} - \eta \Delta w$; $b_{new} = b_{old} - \eta \Delta b$;
        END-FOR
    END-FOR
END-WHILE

During the training phase, the WDNN of the HW-DBN algorithm classifies labeled and unlabeled network traffic instances to build the optimized classification model. The false positive alarms are reduced based on the weight of classes using gradient computing, which is a Nesterov-accelerated adaptive moment estimation algorithm [46]. In this way, the model is not biased and

reduces the overfitting to the majority labels with limited epochs. Moreover, it can detect skewed attacks and has a low number of false alarms.

4.4 DeepIoT.IDS Model

The DeepIoT.IDS model consists of four phases: data pre-processing, UDL pre-training, SDL classification, and evaluation. The first two phases represent the initial data-processing stage, and the last two phases represent the final decision stage. The DeepIoT.IDS model starts with the data pre-processing phase that performs statistical analysis on the CICIDS2017 dataset to discover defects, missing data, and noise. It then converts the dataset to raw data in a specific format and normalizes it by the min-max normalization method, also known as deviation standardization. We need to normalize the dataset to meet the standard conditions of the HW-DBN's input dataset. This normalization makes a linear modification to the original data and maps the result to become a value between [0, 1]. It is represented as $X_{new} = (X - Min)/(Max - Min)$ where *Max* is the maximum value, and *Min* is the minimum value of the sample data.

Then the initial stage employs the deep GB-RBM algorithm to extract network traffic features. This phase generates low presentation features from high dimension data to increase the detection rate for low-frequency attacks. In order to classify the connections of lines to different classes, the deep GB-RBM algorithm provides an abstract feature space to distinguish between attacks and normal traffics of the network. At each layer of the deep GB-RBM, the algorithm minimizes the errors arising from reconstructing the input features. The original unlabeled features are inputs to the first layer, and the output-compressed features are inputs to the second layer. Fig. 5 shows the main components of the DeepIoT.IDS model along with the four processing phases.

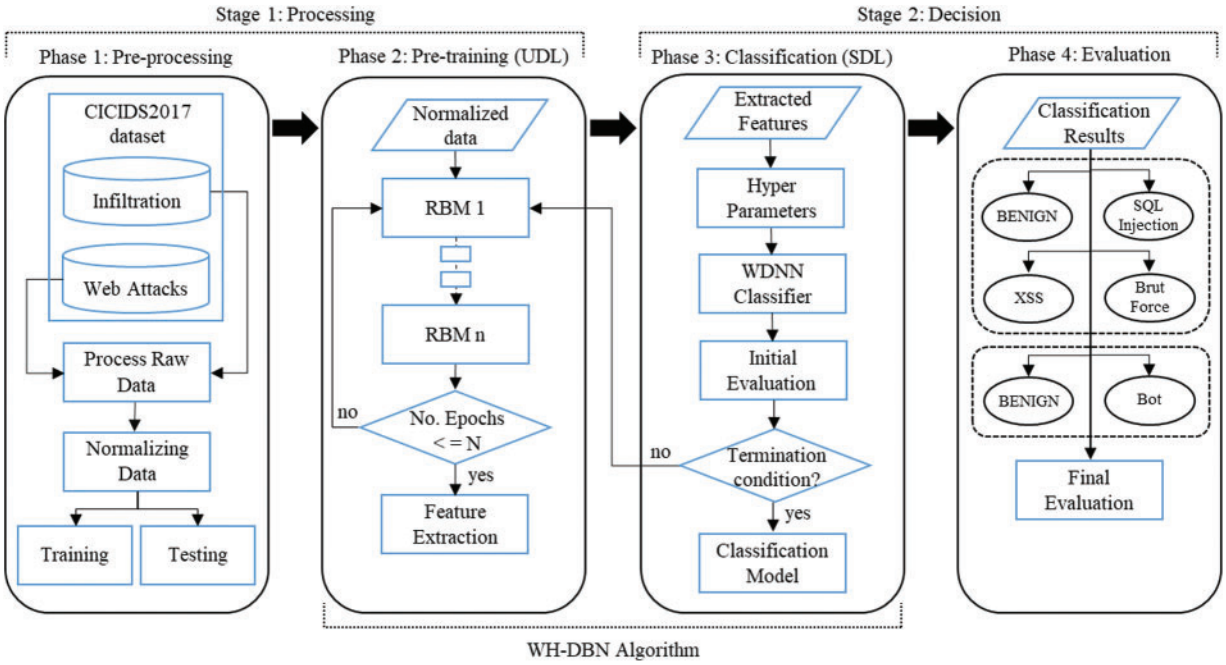


Figure 5: The DeepIoT.IDS model

The proposed DeepIoT.IDS uses the HW-DBN algorithm to build its HDL classification model. Thus, the DeepIoT.IDS model operates according to UDL and SDL techniques. The termination condition of the UDL technique is represented by the N number of epochs. In contrast, the SDL technique's termination condition is represented by exceeding the thresholds of pre-determined evaluation parameters. The evaluation phase represents a set of evaluation metrics that measure the proposed model's performance according to the obtained results during the testing of unlabeled data. These metrics include accuracy, G-mean, precision, recall, and F_Score criteria, as defined in Section 5.2.

5 Experiments

In this work, we propose a DeepIoT.IDS model that integrates a deep Gaussian–Bernoulli type of RBM (Deep GB-RBM) with a WDNN of the HW-DBN algorithm for efficient and reliable IoT network IDS. In this section, we show the implementation results and evaluation of the DeepIoT.IDS model. We evaluate the DeepIoT.IDS model based on 78 features of realistic and new benchmark datasets.

5.1 Benchmark Datasets Description

This work utilizes the CICIDS2017 benchmark dataset that represents BENIGN and modern network traffic attacks [47]. We obtained the dataset from the Canadian Institute for Cybersecurity's datasets repository.[1] We generated the dataset over two different simulation periods of five days and a total of 80 features. We then collected raw data of 51 GB using TCPDUMP and analyzed it with the CICFlowMeter tool. We distinguished this dataset from other datasets by using protocols such as email protocols, FTP, SSH, HTTP, and HTTPS. Moreover, this dataset includes most of the up-to-date common attacks, according to the 2016 McAfee report. The CICIDS2017 dataset includes web attacks and infiltration datasets, which we used to evaluate the DeepIoT.IDS model. Fig. 6 shows the distribution of the CICIDS2017 scenarios that were used to train and test the model.

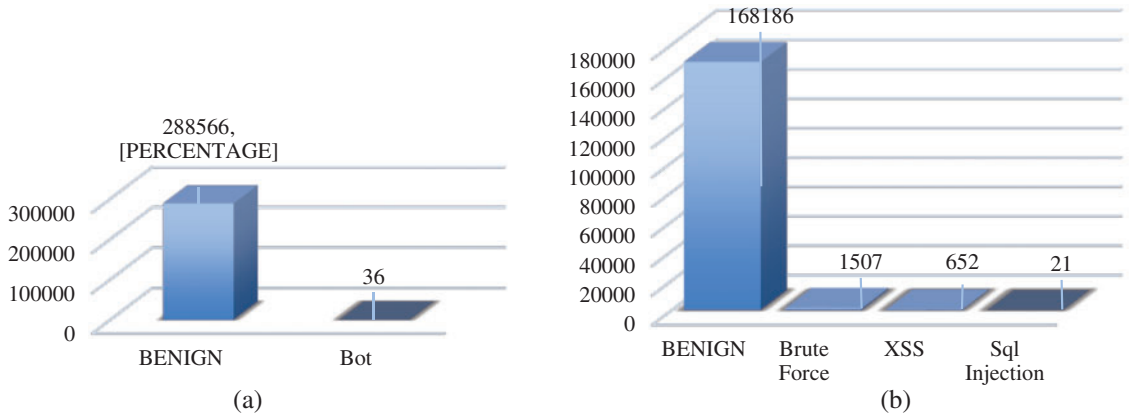


Figure 6: The distribution of web and bot attacks in the CICIDS2017 dataset (a) Bot attack scenarios (b) Web attack scenarios

[1] Canadian Institute for Cybersecurity

The BENIGN class has 168,186 samples, and other rarer classes have 2153 samples. The infiltration dataset has two classes, which are BENIGN with 288,566 samples and bot with 36 samples. Redundant instances may lead to the problem of imbalanced class distribution, which is known to bias a machine learning classifier towards the majority class [48]. However, these datasets have many issues that affect the configuration of the training model and the testing model's construction. We explain these issues as follows:

- **Huge volume of data:** As mentioned earlier, the CICIDS2017 dataset contains data of almost all recent attack labels, which makes the size of the dataset huge. The data size causes a limitation for the research in that there is a huge increase in the loading and processing overheads. The huge amount of data affects the attack detection ability of the conventional NIDS and mandates the development of advanced algorithms.
- **Missing values:** The CICIDS2017 dataset contains duplicate samples and a lot of instances of missing data. Therefore, it requires advanced pre-processing algorithms.
- **High imbalance classes:** The CICIDS2017, like most cybersecurity datasets, is extremely imbalanced. The imbalanced dataset is known to have a high imbalance ratio (IR) when comparing the majority and minority classes, as shown in Eq. (22):

$$IR = \frac{majority\ class}{minority\ classes} \tag{22}$$

Table 1: The specifications of the CICIDS2017 dataset

Dataset	Class No.	Type of class	Instance No.	IR
DDOS	2-classes	DDoS	128027	1.3102
		BENIGN	97718	
PortScan	2-classes	PortScan	158930	1.2461
		BENIGN	127537	
Bot	2-classes	BENIGN	189067	96.168
		Bot	1966	
Infiltration	2-classes	BENIGN	288566	8015.7
		Bot	36	
Web attacks	4-classes	BENIGN	168186	*
		Brute Force	1507	111.97
		XSS	652	257.95
		SQL Injection	21	800.85
Patator	3-classes	BENIGN	432074	*
		FTP-Patator	7938	44.364
		SSH-Patator	5897	73.270
DoS, Heartbleed	5-classes	BENIGN	440031	*
		DoS Hulk	231073	1.9042
		DoS GoldenEye	10293	42.750
		DoS slowloris	5796	75.919
		DoS Slowhttptest	5499	80.021
		Heartbleed	11	40002.8

The CICIDS2017 includes several imbalanced benchmark multi-class datasets, as defined in Tab. 1. The level of imbalanced attacks differs from other attacks. The higher IR indicates an extremely imbalanced dataset, e.g., infiltration and web attacks; hence, we used them for testing the DeepIoT.IDS model. The web attacks dataset has four classes: BENIGN (168 instances and 186 IR), brute force (1507 instances and 111.97 IR), XSS (652 instances and 257.95 IR), and SQL injection (21 instances and 800.85 IR). The infiltration dataset has two classes: BENIGN (288,566 instances) and bot (36 instances and 8015.7 IR).

5.2 Evaluation Metrics

We cannot visually and easily interpret most real data; therefore, we must rely upon more quantitative metrics to evaluate a model and determine which classes are easy for a model to use. This entails the use of recommended metrics for the evaluation of the DeepIoT.IDS model by utilizing performance metrics like accuracy, precision, sensitivity (recall), F_Score, training time, and prediction time, as follows [3,10]:

- Accuracy is the ratio of correct predictions for both TP and TN prediction of attacks compared with the total number of tested cases:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{23}$$

- Precision (true positive rate) is used to measure the proportion of positives that are correctly identified:

$$Precision = \frac{TP}{TP + FP} \tag{24}$$

- Recall (sensitivity) measures the number of correct classifications penalized by the number of missed entries identified:

$$Recall = \frac{TP}{TP + FN} \tag{25}$$

- F_Score is a measure to find a balance between precision and recall:

$$F\_Score = 2 * \frac{Precision + Recall}{Precision * Recall} \tag{26}$$

- Testing time describes how much time an approach has taken to predict the whole dataset as either normal or attack:

$$Testing\ \text{time} = end_{time}^{testing} - start_{time}^{testing} \tag{27}$$

We quantify true positive (TP) measures as the number of attacks appropriately determined as attacks. In contrast, we define false positive (FP) measures as the number of normal connections wrongly determined as attack connections. In addition to that, we define true negative (TN) measures as the number of normal connections accurately determined as normal. Finally, we define the false negative (FN) measures as the number of attack connections wrongly determined as normal [11]. We use the G-mean of Eq. (19) to evaluate the degree of inductive bias that considers both negative accuracy and positive accuracy. The G-mean is less sensitive to data

distributions in that its high value denotes high classification performance for the majority and minority classes. The G-mean formula is given as follows:

$$G\text{-}mean = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}} \tag{28}$$

5.3 Results and Discussion

We conducted the experimental tests of the DeepIoT.IDS models using TensorFlow 1.2 version NVIDIA 1080 on a desktop with a 2.0 GHz Intel Core i3 CPU, 4 GB RAM, and 64-bit Ubuntu16.06 operating system. The test considered a case in which the proposed model detects an anomaly in the computer network system. The test benchmarks consisted of three DL models–BB-RBM, BB-DBN, and deep AE–and our proposed DeepIoT.IDS model. We selected these other models due to their similarity to our DeepIoT.IDS model. The experimental setting included a deep GB-RBM consisting of three layers, 100 hidden neurons, and 78 visible neurons. We selected the ReLU function as the model's activation function and randomly set the initial weights of the 78 hidden neurons. We used five epochs to train the model and divided the dataset into 70% training set and 30% testing set [20]. We used the evaluation criteria of precision, recall, F_Score, and G-mean to evaluate the four models. The first experiment implemented the CICIDS2017 dataset in the web attack scenarios containing four classes (BENIGN, brute force, XSS, and SQL injection). The pre-processing of this experiment removed three of the 81 features (leaving a total of 78). Subsequently, we performed a column-wise normalization of [0, 1] interval, as described in Section 3.3. The results from running the BB-RBM, BB-DBN, deep AE, and our proposed DeepIoT.IDS models for web attack scenarios are shown in Tab. 2.

Table 2: Comparing the performance of the four models in the web attack scenarios

No.	Model	Classes	Precision	Recall	F_Score	Support
1.	BB-RBM	BENIGN	0.99	1.00	0.99	55527
		Brute Force	0.00	0.00	0.00	485
		XSS	0.00	0.00	0.00	200
		SQL Injection	0.00	0.00	0.00	9
2.	BB-DBN	BENIGN	0.99	1.00	0.99	55527
		Brute Force	0.00	0.00	0.00	485
		XSS	0.00	0.00	0.00	200
		SQL Injection	0.00	0.00	0.00	9
3.	Deep AE	BENIGN	0.99	1.00	0.99	55527
		Brute Force	0.95	0.16	0.28	485
		XSS	0.40	0.00	0.00	200
		SQL Injection	0.00	0.00	0.00	9
4.	DeepIoT.IDS	BENIGN	1.00	1.00	1.00	55527
		Brute Force	0.60	0.91	0.73	485
		XSS	0.40	0.01	0.02	200
		SQL Injection	1.00	0.56	0.67	9

As shown in Tab. 2, the proposed DeepIoT.IDS provides the best learning and testing results, which surpassed the BB-RBM, BB-DBN, and deep AE classifiers. In general, the results show that the HW-DBN algorithm of the DeepIoT.IDS model was able to learn a greater number of important features of the CICIDS2017 dataset than the other models. Consequently, the model generated confusion matrices that gave superior performance compared to those of the other models. Hence, the DeepIoT.IDS model was able to detect the four types of attack (BENIGN, brute force, XSS, and SQL injection) while other models suffered low detection rates for brute force, XSS, and SQL injection because of their poor handling of the skewed attacks distribution, noise, and high data dimensionality problems. Tab. 3 reports the results of the four models for the web attack and bot attack scenarios of the CICIDS2017 dataset. The results show that DeepIoT.IDS achieved the highest accuracy (99.38%) and the highest G-mean (99.40%) compared with the other three models. Furthermore, it performed with a lower number of epochs to achieve the required results and with the best execution time of 363.941 ms.

Table 3: Comparing the accuracy and G-mean of the four models

No.	Model	Accuracy	G-Mean	Iteration of FL	Testing time (ms)
Web attack scenarios					
1.	BB-RBM	0.9874	0.1110	100	980.07
2.	BB-DBN	0.9874	0.1110	100	2735.98
3.	Deep AE	0.9890	0.3610	100	498.37
4.	DeepIoT.IDS	0.9938	0.9940	5	363.94
Bot attack scenarios					
1.	BB-RBM	0.9874	0.0001	3	126.37
2.	BB-DBN	0.9998	0.9573	20	338.78
3.	Deep AE	0.9998	0.0112	1000	2.63
4.	DeepIoT.IDS	0.9999	0.9574	5	265.39

The bot attack scenarios only included two classes of attack, BENIGN, and bot, as shown in the evaluation phase of Fig. 6. The results in Tab. 3 show that the DeepIoT.IDS achieved the highest accuracy (99.99%) and the highest G-mean (95.74%) compared with the other three models. Furthermore, it performed with the second lowest number of epochs to get the required results but with the second highest execution time of 265.39 ms. For a better demonstration of the results of both web and bot attacks, Fig. 7 compares the accuracy and G-mean of the four models according to the web attack and bot attack test scenarios.

Subsequently, the benchmarking analysis compared the newly developed model with the existing ones, which confirmed the superiority of the DeepIoT.IDS model. A two-stage deep feature extraction and learning algorithms support the DeepIoT.IDS model to enable it to detect existing and newly discovered intrusion attacks. The model achieves higher accuracy and G-mean by constructing features from a huge amount of data for a set of refined training features. The results confirm that the model is less vulnerable to multi-type and unbalanced attacks than other DL models.

Anomaly IDSs (AIDS) in general have several limitations, including low accuracy and high false alarms. Signature-based IDS models could detect traditional intrusions, and AIDS could

detect novel attacks. Furthermore, developing a real-time detection, anomaly-based IDS for IoT networks is very demanding. This is because such an IDS would need to distinguish a normal behavior first and then detect malicious behavior to produce more realistic results. Subsequently, we summarize the limitations of the present work as follows: (i) additional classification models should be tested to provide a more comprehensive evaluation of the results, (ii) given that the proposed DeepIoT.IDS model relies on all available features. A feature reduction approach might limit the computational cost and time complexity of the training and testing processes, and (iii) other evaluation criteria and testing datasets should be considered to further the analysis of the proposed model.

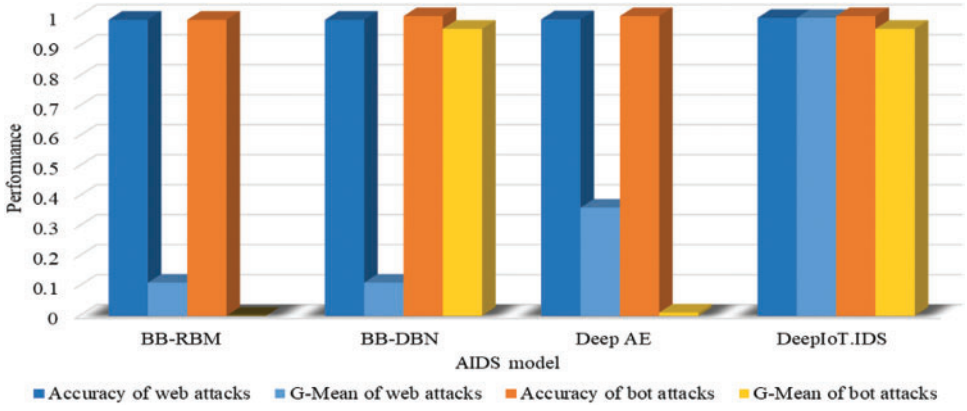


Figure 7: The relationship between accuracy and G-mean

6 Conclusion and Future Work

This paper proposes a novel hybrid weighted deep belief network (HW-DBN) algorithm that integrates a deep Gaussian–Bernoulli type of RBM (deep GB-RBM) algorithm with a weighted deep neural network (WDNN) algorithm. The HW-DBN algorithm deploys the deep GB-RBM for unsupervised deep learning (UDL) and WDNN backpropagation for supervised deep learning (SDL) to manifest a hybrid deep learning (HDL) approach. The HW-DBN algorithm is incorporated within a DeepIoT.IDS model to deal with the problem of intrusion detection in IoT networks efficiently and reliably. The DeepIoT.IDS model's operating regime covers both processing and decision stages. The processing stage includes pre-processing and pre-training phases, while the decision stage involves the classification and evaluation phases. We tested the DeepIoT.IDS model by utilizing the CICIDS2017 benchmark datasets. The CICIDS2017 dataset is suitable to represent IoT environments because it has complex anomaly patterns, novel attacks, and highly imbalanced classes. We conducted extensive experiments on web attack and bot attack scenarios to evaluate the performance of the model. The web attack scenarios comprise multi-class detection of BENIGN, brute force, XSS, and SQL injection attacks, while the bot attack scenarios comprise signal-class detection of BENIGN and bot attacks. In the first set of scenarios, the results show that the DeepIoT.IDS model achieved the highest accuracy (99.38%) and the highest G-mean (99.40%) compared with the BB-RBM DBN, GB-RBM DBN, and deep AE models. Furthermore, it performed with a lower number of epochs to get the required results and the best execution time of 363.941 ms. In the second set of scenarios, the results show that the DeepIoT.IDS model also achieved the highest accuracy (99.99%) and the highest G-mean (95.74%)

compared with the other three models. Furthermore, it performed with the second lowest number of epochs to get the required results but with the second highest execution time of 265.39 ms. The results, therefore, demonstrate the robustness and efficiency of the DeepIoT.IDS model to represent a future benchmark for constructing IDS based on deep learning algorithms in the research community of network and security. Future work may consider developing a real-time AIDS that improves the overall performance of IoT networks. Such an AIDS model would be more realistic and would need to quickly detect normal behaviors in order to trace and eliminate malicious behaviors.

Acknowledgement: This work was partially funded by the Industry Grant Scheme from Jaycorp Berhad in cooperation withUNITAR International University. The authors would like to thank INSFORNET, the Center for Advanced Computing Technology (C-ACT) at Universiti Teknikal Malaysia Melaka (UTeM), and the Center of Intelligent and Autonomous Systems (CIAS) at Universiti Tun Hussein Onn Malaysia (UTHM) for supporting this work.

Funding Statement: The authors received no specific funding for this study.

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

References

[1] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford and C. Wijenayake, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mob. Comput.*, vol. 18, no. 8, pp. 14, 2019.
[2] Ericsson, "Ericsson mobility report–On the brink of the information society," *Ericsson*, pp. 1–36, 2019.
[3] Palo Alto, *Unit 42 IoT Threat Report*. California, United States: Palo Alto, 2020.
[4] H. Tahaei, F. Afifi, A. Asemi, F. Zaki and N. B. Anuar, "The rise of traffic classification in IoT networks: A survey," *J. Netw. Comput. Appl.*, vol. 154, pp. 102538, 2020.
[5] A. Thakkar and R. Lohiya, *A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT: Recent Updates, Security Issues and Challenges*. Netherlands: Springer, 2020.
[6] C. Liang, "Intrusion detection system for the internet of things based on blockchain and multi-agent systems," *Electron*, vol. 9, no. 7, pp. 1–27, 2020.
[7] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed and W. M. Abduallah, "Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods," *IEEE Access*, vol. 7, pp. 51691–51713, 2019.
[8] M. Jiang, "Text classification based on deep belief network and softmax regression," *Neural Comput. Appl.*, vol. 29, no. 1, pp. 61–70, 2018.
[9] A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia and P. Casas, "A survey on big data for network traffic monitoring and analysis," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 3, pp. 800–813, 2019.
[10] F. Sadikin, T. Van Deursen and S. Kumar, "Internet of things: A hybrid Zigbee IoT intrusion detection system using secure and efficient data collection," *Internet of Things*, vol. 12, pp. 100306, 2020.
[11] K. Kim and M. E. Aminanto, "Deep learning in intrusion detection perspective: Overview and further challenges," *Proc. Int. Res. Conf. Eng. Technol.*, pp. 1–12, 2017.
[12] B. A. Tama, L. Nkenyereye, S. M. R. Islam and K. S. Kwak, "An enhanced anomaly detection in web traffic using a stack of classifier ensemble," *IEEE Access*, vol. 8, pp. 24120–24134, 2020.
[13] D. Papamartzivanos, F. Gomez Marmol and G. Kambourakis, "Introducing deep learning self-adaptive misuse network intrusion detection systems," *IEEE Access*, vol. 7, pp. 13546–13560, 2019.
[14] T. Aldwairi, D. Perera and M. A. Novotny, "An evaluation of the performance of restricted Boltzmann machines as a model for anomaly network intrusion detection," *Computer Networks*, vol. 144, pp. 111–119, 2018.

[15] A. Elsaeidy, K. S. Munasinghe, D. Sharma and A. Jamalipour, "Intrusion detection in smart cities using restricted Boltzmann machines," *J. Netw. Comput. Appl.*, vol. 135, no. 6, pp. 76–83, 2018.
[16] A. Ng, *Deep Learning Tutorial*. California, USA: Univ. Stanford, 2015.
[17] C. Zhang, K. C. Tan and R. Ren, "Training cost-sensitive deep belief networks on imbalance data problems," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2016, pp. 4362–4367, 2016.
[18] P. Wei, Y. Li, Z. Zhang, T. Hu, Z. Li *et al.,* "An optimization method for intrusion detection classification model based on deep belief network," *IEEE Access*, vol. 7, pp. 87593–87605, 2019.
[19] M. Z. Alom and T. M. Taha, "Network intrusion detection for cyber security using unsupervised deep learning approaches," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2017-May, pp. 3830–3837, 2017.
[20] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset," *IEEE Access*, vol. 9, pp. 22351–22370, 2021.
[21] A. Verma and V. Ranga, "On evaluation of network intrusion detection systems: Statistical analysis of CIDDS-001 dataset using machine learning techniques," *Pertanika J. Sci. Technol.*, vol. 26, no. 3, pp. 1307–1332, 2018.
[22] T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets," in *2017 IEEE 26th Int. Symp. on Industrial Electronics*, Edinburgh, UK, IEEE, pp. 1881–1886, 2017.
[23] M. Nawir, A. Amir, N. Yaakob and O. N. G. B. I. Lynn, "Multi-classification of UNSW-NB15 dataset for network anomaly detection system,," *Journal of Theoretical and Applied Information*, vol. 96, no. 15, pp. 5094–5104, 2018.
[24] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis and R. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," arXiv preprint arXiv: 1701.02145, 2017.
[25] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat *et al.,* "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
[26] F. A. Khan, A. Gumaei, A. Derhab and A. Hussain, "TSDL: A two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.
[27] C. Yin, Y. Zhu, J. Fei and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[28] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," *J. Netw. Comput. Appl.*, vol. 169, pp. 102767, 2020.
[29] G. Zhao, C. Zhang and L. Zheng, "Intrusion detection using deep belief network and probabilistic neural network," in *2017 IEEE Int. Conf. on Computational Science and Engineering and IEEE Int. Conf. on Embedded and Ubiquitous Computing*, Guangzhou, China, IEEE, vol. 1, pp. 639–642, 2017.
[30] T. F. Ghanem, "A hybrid approach for efficient anomaly detection using metaheuristic methods," *Journal of Advanced Research*, vol. 6, no. 4, pp. 609–619, 2015.
[31] V. Hajisalem and S. Babaie, "A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection," *Comput. Networks*, vol. 136, pp. 37–50, 2018.
[32] M. A. Ferrag, L. Maglaras, S. Moschoyiannis and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, pp. 102419, 2020.
[33] D. Charte, F. Charte, S. García, M. J. Del Jesus and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Inf. Fusion*, vol. 44, pp. 78–96, 2018.
[34] Y. Imamverdiyev and F. Abdullayeva, "Deep learning method for denial of service attack detection based on restricted Boltzmann machine,," *Big Data*, vol. 6, no. 2, pp. 159–169, 2018.
[35] U. Fiore, F. Palmieri, A. Castiglione and A. De Santis, "Network anomaly detection with the restricted Boltzmann machine," *Neurocomputing*, vol. 122, no. 3, pp. 13–23, 2013.
[36] S. M. Erfani, S. Rajasegarar, S. Karunasekera and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognition*, vol. 58, no. 7, pp. 121–134, 2016.

[37] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *2016 15th IEEE Int. Conf. on Machine Learning and Applications*, Anaheim, CA, USA, IEEE, pp. 195–200, 2016.

[38] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP 2018-Proc. 4th Int. Conf. Inf. Syst. Secur. Priv.*, vol. 2018, pp. 108–116, 2018.

[39] H. Yu, "A gentle tutorial on restricted Boltzmann machine and contrastive divergence," 2017.

[40] G. Casella and E. I. George, "Explaining the Gibbs sampler," vol. 3, no. 3, pp. 167–174, 2016. Stable URL: [Online]. Available: http://www.jstor.org/stable/2685208.

[41] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[42] J. Tao, Y. Liu and D. Yang, "Bearing fault diagnosis based on deep belief network and multisensor information fusion," *Shock and Vibration*, vol. 2016, no. 7, pp. 1–9, 2016.

[43] Z. Alom, V. Bontupalli and T. M. Taha, "Intrusion detection using deep belief networks," in *2015 National Aerospace and Electronics Conf.*, Dayton, Ohio, USA, IEEE, pp. 339–344, 2015.

[44] H. Shao, H. Jiang, X. Li and T. Liang, "Rolling bearing fault detection using continuous deep belief network with locally linear embedding," *Comput. Ind.*, vol. 96, no. 61, pp. 27–39, 2016.

[45] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020.

[46] S. Ruder, "An overview of gradient descent optimization algorithms," *Sebastian Ruder*, 2016. [Online]. Available: https://ruder.io/optimizing-gradient-descent/.

[47] C. Zhang, K. C. Tan, H. Li and G. S. Hong, "A cost-sensitive deep belief network for imbalanced classification," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 1, pp. 1–14, 2018.

[48] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.