An Intrusion Detection System Using a Machine Learning Approach in IOT-based Smart Cities

Liloja[1*] and Dr.P. Ranjana[2]

[1*]Research Scholar, Department of Computer Science, Hindustan Institute of Technology and Science, Chennai, Tamil Nadu, India. lilojabasheer@gmail.com

[2]Professor and HOD (MCA), Hindustan Institute of Technology and Science, Chennai, Tamil Nadu, India. pranjana@hindustanuniv.ac.in

Received: September 12, 2022; Accepted: November 19, 2022; Published: February 28, 2023

Abstract

For a long time, the digitization of all aspects of life in current cultures is seen as a procured gain. In any way, the computerized world is noticeably flawed and numerous risks and dangers are present as in the terrestrial land. People's daily life has changed due to the quick and advanced level of improvement in smart cities. The most important problem that needs to be looked upon is citizens' life, security and privacy issues. The use of Deep Learning (DL), a subcategory of Machine learning (ML) has excelled in the field of smart cities. So, the following stages in this paper bring an effective intrusion detection system using deep learning. a) Data collection from standard datasets such as GPRS, CIDDS001, as well as UNSW-NB15 contains various types of attacks, these will be given for b) Preprocessing, for eliminating anomalies using missing value removal, and normalization techniques. Then from those data, quintessential features are extracted using Autoencoder (AE) and then from those several features, d) feature selection for selecting and mostly removing timestamps from attack dataset using Random Forest (RF) and finally for e) prediction with help of Restricted Boltzmann Network (RBN). Experiment evaluation states that proposed model (RF-RBN) performed better over various state-of-art models under various measures (accuracy:0.95, sensitivity:0.96, specificity:0.97, detection rate:0.95).

Keywords: Deep Learning, Intrusion Detection System, IOT, Machine Learning, Restricted Boltzmann Network, Smart Cities.

1 Introduction

A smart city raises the standard of living for its citizens by maximising the potential of its organisational structure. Cities can use smart city technologies to monitor their growth and to immediately communicate with their municipal infrastructure and social networks. ICT is used to improve urban services, which also aids in cost and resource reduction and improves citizen-government interactions. Smart city elements were created to control urban traffic and provide speedy responses.it is more prepared to handle problems than a city where relationships with its residents are mostly "transactional" [1]. According to Rudolf Giffinger, an expert in research analysis on urban and regional growth at Vienna University of Technology, smart cities have six essential characteristics. Both regional and

*Corresponding author: Research Scholar, Department of Computer Science, Hindustan Institute of Technology and Science, Chennai, Tamil Nadu, India.

neoclassical theories of development and urban expansion are related to these characteristics. The notion of regional competitiveness, transportation economics, information and communication technology, environmental assets, and social capital are a few of these themes. Raising the level of standard of living of citizens while ensuring environmental preservation is the key challenge smart cities face. One use of the internet of things is the "Smart City," which is a financially sound urban development strategy that provides a high standard of living for its citizens. The creation of smart cities uses various technologies [3].

Deep Learning algorithms are customized for many applications including data classification and Intrusion Detection (ID). As a Binary classifier, ID in IoT networks is categorized as being either under attack or belonging to the usual class using a trained classification model. The ultimate objective is to lower false alarm rates while raising accuracy. [4]. Information discovery is discussed using data mining. Security-related difficulties in smart cities are shown in Figure 1.

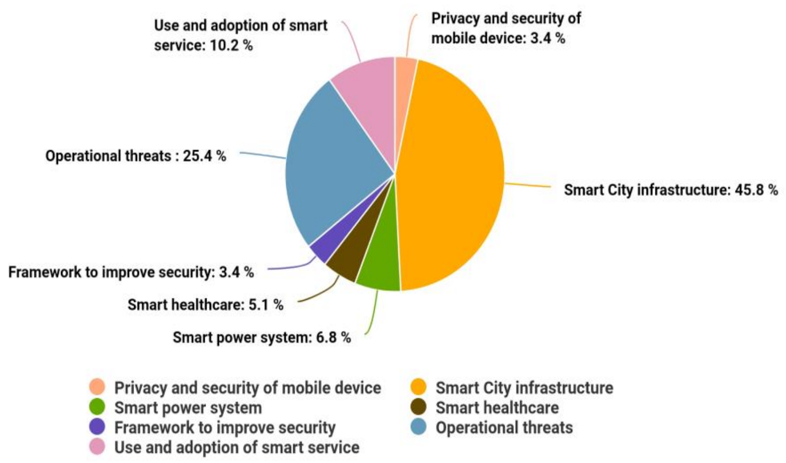


Figure 1: Security-related Possible Issues Occurring in Various Smart Sectors

Different methods that use machine learning and DL for detection and prevent invasions have been developed in the past. Many of them paid less attention to the pre-processing stage, particularly feature selection. So, the specific algorithm's classification accuracy is directly impacted [5-7]. The length of time required for training the models is additionally lengthened by the premature pre-processing phase. The present neural network algorithm's back-propagation method lengthens training time as well [8–10].

1.1 Research Objectives

The following are the objectives considered for intrusion detection in smart cities which are focused on in this paper:

- Develop an effective Intrusion detection in smart cities using a deep learning model.
- Bring an integration model of machine learning and deep learning technique.
- Here the use of 3 standard datasets such as GPRS, CIDDS001 and UNSW-NB15 to improve the model performance and reduce the data Imbalance problem.
- With help of a stack of RBM networks, Prediction of attacks in smart cities is possible.

2 Methodology

Figure 2 represents the overall architecture of the proposed framework. The different stages are a) Data Collection from the popular data repository such as GPRS, CIDDS001, and UNSW-NB15 in which each dataset contains certain categories of possible attack classes which could cause issues in IoT based smart cities. Once these data are collected, they will undergo b) Preprocessing stage where the normalization technique is used. When working with attributes of multiple scales, normalization is required. Attribute of smaller values may be diluted by the values of other attributes, which have values on a greater scale. Once these data are normalized, they will undergo a c) feature extraction stage were using an autoencoder, quintessential features are extracted. The autoencoder technique makes use of 120 input neurons. We repeatedly converted X into 100, 80, 60, 40, 22, 20, 15, 10, and 8 neurons in the hidden layer to get the best accuracy score. The output will then be feature extracted from the inputs. Then from those extracted features, feature reduction, also known as d) feature selection for selecting required features for even more effective performance is done. Finally, e) the Prediction stage, with the help of a stack of Restricted Boltzmann Machine (RBM) will be used for predicting the classes. Two stochastic network layers make up the RBM's hidden and visible layers. Data can travel between levels because the links between them are symmetric and bidirectional.

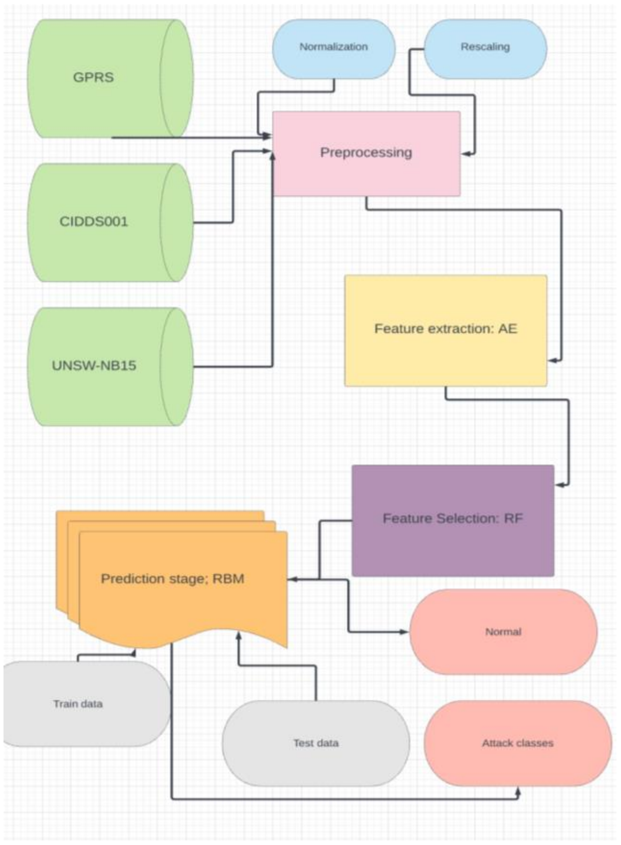


Figure 2: Proposed Framework Architecture

2.1 Data Collection

Here data are collected from the popular repository as GPRS, CIDDS001, as well as UNSW-NB15. Each is described below.

GPRS dataset: The hidden and visible layers, both of which are stochastic networks, make up the RBM. Due to connections between layer neurons and neurons in other layers, the network is referred to as being constrained. Data flow is made possible by the bidirectional and symmetric connections between layers.

CIDDS001 dataset: A labelled flow-based dataset called CIDDS-001 ("CIDDS-001," 2017) [17,18] is available (Ring, Wunderlich, Grudl, Landes, & Hotho, 2017). It was created primarily for AD-based NIDS evaluation purposes. The dataset includes traffic from External Servers and OpenStack. 13 features and one class attribute make up CIDDS-001. This analytical investigation used a total of 11 features. Since they provide more details about actual attacks, the study ignored the features of Attack ID and Attack Description. As a result, these characteristics had little impact on the analysis. For analysis, data on 172,839 instances from OpenStack Server and about 153,026 instances from other servers were gathered. The dataset's instances were classified into the normal, attacker, victim, suspicious, and unknown classes. The CIDDS-001's description may be found in Table 1

UNSW-NB15 dataset: A dataset for network intrusions is UNSW-NB15 [19,20]. There are nine different assaults in it, including worms, backdoors, DoS attacks, and fuzzes. The collection also includes unprocessed network packets. The testing set features 82,332 records from the attack and normal categories, compared to 175 341 records in the training set. A total of 49 features with the class designation are produced using twelve techniques, the Argus and Bro-IDS tools, and other factors. Figure 3 displays the UNSW-NB15 dataset diagram in its entirety.

2.2 Preprocessing

For many activities linked to data mining and anomaly intrusion detection, data preparation is crucial. Most anomaly detection algorithms that learn the statistical characteristics of variables acquired from audit data must first prepare their data, and data normalisation is a crucial stage in this process. Data normalisation tries to proportionally scale the values of each continuous characteristic into a range so that none of the continuous variables can significantly affect the other continuous attributes. In the phase of data preprocessing, attribute normalisation is the main topic of this section. In this study, four more approaches for attribute normalisation are used in addition to the original attributes.

2.2.1 Normalization

2.2.1.1 Mean Range [0,1]

It is simple to convert an attribute into a range of values between [0,1] if we know its maximum and minimum values.

$$x_i = \frac{v_i - min(v_i)}{max(v_i) - min(v_i)} \qquad (1)$$

Where v_i is the attribute's actual value, and the attribute's maximum and lowest are calculated over all other values. Normally, if the maximum and minimum are equal, x_i is set to 0.

2.2.1.2 Statistical Standardization

Data from any Normal distribution should be transformed into a conventional Normal distribution with a mean of zero and a unit variance using statistical normalisation. The definition of statistical normalisation is:

$$x_i = \frac{v_i - \mu}{\sigma} \qquad (2)$$

where σ is the standard deviation and $v_i - \mu$ is the mean of n values for a specific attribute: = 1/n €ni=1 vi.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i - \mu)} \qquad (3)$$

However, when statistical normalisation is utilised, the data set should follow a Normal distribution. More specifically, the central limit theorem specifies that there should be a large number of samples (n). The attribute's value is not statistically normalised into the range [0,1]. Instead, 99.9% of the attribute's samples fall inside the range [-3, 3].

2.2.1.3 Ordinal Normalization

Ranking an attribute's continuous value before normalising the rank into [0,1] is known as ordinal normalisation. The ordinal normalisation is defined as:

$$x_i = \frac{r-1}{max(r)-1} \qquad (4)$$

Where r is the rank of a particular value in an attribute.

Evidently, ordinal normalising also place an attribute's values in the [0,1] range. In this study, if several attribute values are the same, the rank is not raised. For example, the following rank after 18 is 16 if certain values are ordered as…,15,15,15.

2.2.1.4 Frequency Normalization

By taking into account the ratio of a value to the attribute's total value, frequency normalisation seeks to normalise an attribute. It's described as:

$$x_i = \frac{v_i}{\sum_i v_i} \qquad (5)$$

Frequency normalization also scales an attribute into [0,1].

2.2.2 Feature Scaling

To cope with local optima and skewness towards specific features, feature scaling is a crucial step. Additionally, faster training is made possible for the ML-based IDS. Applying standard scaling causes the values to be replaced by their Z-scores. A data point's Z-score indicates how far away from the mean value the data point is. Zi, the z-score, is determined by the following equation:

$$z_i = \frac{x_i - x_{mean}}{x_{std}} \qquad (6)$$

in which x_i stands for the value of each feature, x_{mean} for the feature's average value, and x_{std} for the standard deviation.

2.3 Feature Extraction

The input and output layers of a special type of multilayer perceptron known as an autoencoder (Figure 3) have the same number of neurons.

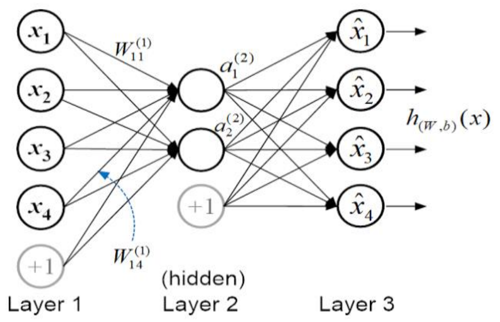


Figure 3: Autoencoder Network

Encoder and decoder are two components of the autoencoder's architecture that are trained at each subsequent layer. Each layer of a deep autoencoder takes input from the one before it. Specifically, the output is reconstructed from the compressed representation that the autoencoder has been trained to transform the raw input into.

$$H(x) = \sigma(W \cdot l(x) + b) \qquad (7)$$

In this case, the learning function is sigmoid. The terms "weight" and "bias" are respectively W and b. With the function described in layer (5).

$$R(x) = a(W \cdot H(x) + b') \qquad (8)$$

Where, from the latent representation H(x), R(x) is the expected output of the input I(x). To lower the network's reconstruction error, the autoencoders' weights are optimised. The computation for the reconstruction error is:

$$(I(x), R(x)) = \| k(x) - R(x) \|^2 \qquad (9)$$

Additionally, the reconstruction error for binary values using the cross-entropy measure (7):

$$L(I(x), R(x)) = -\sum_{k=1} [I(x)_k log\, R(x)_k + (1I(x)_k) log(1 - R(x)r_k)] \ (10)$$

2.4 Feature Selection

A Random Forest (RF) is made up of a number of classification or regression trees that have not been pruned. Random forest generates a large number of classification trees, and each tree is constructed using a distinct bootstrap sample selected from the initial data and a tree classification algorithm. After the forest has been built, a new object that needs to be identified is placed on each tree. Each tree casts a vote, symbolising the choice that establishes the class of the item. The class that best supports the object is chosen by the forest. The random forests algorithm is as follows (for both classification and regression):

1) To build ntree bootstrap samples, create n samples from the training set.
2) Create a regression tree, with adjustment for each of the bootstrap samples. The tree is allowed to grow to its maximum capacity without any pruning. When the number of predictors, mtry, equals p, bagging can be viewed as a particular instance of random forests.
3) To forecast new data, combine the ntree tree

The error rate can be assessed in two different ways. The dataset must first be split into training and test halves. The forest can be constructed using the training part, and the error rate can be calculated using the test part. Utilizing the Out-of-Bag (OOB) error estimate is an alternative strategy. We do not need to separate the training data in order to get it because the RF approach calculates OOB errors during the training phase. Both methods have been employed in our research to assess the mistake rate.]. Features are picked at random as the forest grows.

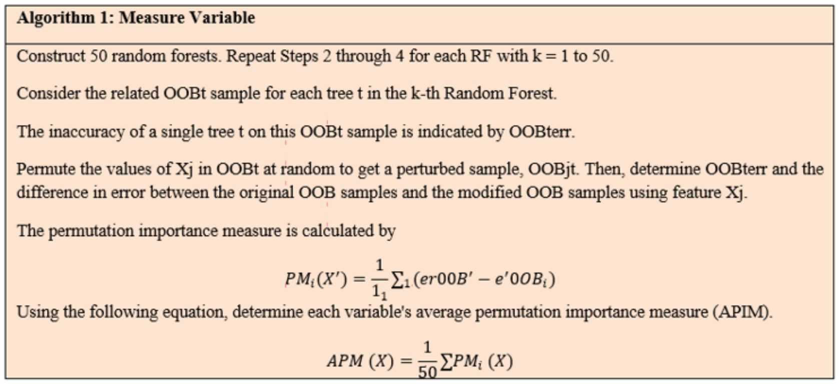
The main tuning parameter is the number of characteristics used to separate each node for each tree (mtry). This parameter has to be improved for random forests to perform better. Only choose a sufficient number of trees to ensure that the OOB error has been stabilised. The descriptor's significance or intrinsic proximity can be reached with more than 500 trees, however typically 500 are enough. RF does not use a halting criteria, instead penalising "too many" trees by wasting computational resources. The size of the trees that are grown is somewhat influenced by this parameter. In Random Forest, the default value for classification is 1, while the default value for regression is 5, ensuring that trees reach their full potential.

2.4.1 Variable Selection

The high dimensionality of many pattern recognition applications has made feature selection strategies urgently necessary. There are many different purposes for feature selection in this discipline, but the two most significant ones are:

1) To prevent overfitting and enhance model performance; and
2) To gain a more thorough understanding of the underlying mechanisms that produced the data. For the majority of life science challenges, the interpretability of machine learning models is seen as being as crucial to prediction accuracy.

In contrast to most other classifiers, Random Forest does feature selection right away when building a classification rule. The total or average relevance value for a feature in a forest is determined by adding up the importance values of all the trees in the forest. The most common variable significance measure employed in RF is probably the Permutation Importance Measure (PIM). Not every training sample is used by the RF algorithm when building a single tree. To determine how accurately the forest can be identified, use the remaining samples that are Out of Bag (OOB). The classification accuracy of the intact OOB samples and the OOB samples with the specific feature permuted should be compared in order to determine the relative significance of each feature in the tree. As shown in Algorithms 1 and 2, we employed the PIM in this work to evaluate the relevance of a subset of features in comparison to all features.


Algorithm 1: Measure Variable

Construct 50 random forests. Repeat Steps 2 through 4 for each RF with k = 1 to 50.

Consider the related OOBt sample for each tree t in the k-th Random Forest.

The inaccuracy of a single tree t on this OOBt sample is indicated by OOBterr.

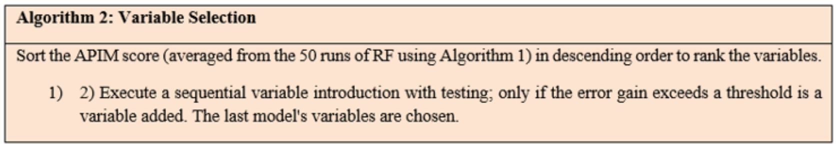
Permute the values of Xj in OOBt at random to get a perturbed sample, OOBjt. Then, determine OOBterr and the difference in error between the original OOB samples and the modified OOB samples using feature Xj.

The permutation importance measure is calculated by

$$PM_i(X') = \frac{1}{1_1}\Sigma_1(er00B' - e'0OB_i)$$

Using the following equation, determine each variable's average permutation importance measure (APIM).

$$APM\ (X) = \frac{1}{50}\Sigma PM_i\ (X)$$


Algorithm 2: Variable Selection

Sort the APIM score (averaged from the 50 runs of RF using Algorithm 1) in descending order to rank the variables.

1) 2) Execute a sequential variable introduction with testing; only if the error gain exceeds a threshold is a variable added. The last model's variables are chosen.

2.5 Classification

Two stochastic network layers make up the RBM's hidden and visible layers. While the hidden layer attempts to feature learning from the visible layer with a focus on exhibiting the probabilistic data distribution, the visible layer represents the collected data. Since layer neurons connect to neurons in other layers, the network is said to be limited. Data transport is made possible by the bidirectional and symmetric connections between layers. Figure 4 shows the RBM, where the visible layer's neuron counts are shown as m and the hidden layer's counts as n, respectively. The weight matrix is represented by the letters w, and the bias vectors by the letters a and b. To create the probability distribution on observable features, RBM uses hidden layer variables. Each input pattern feature results in the production of one visible unit, which is made up of other visible units and is reliant on observational components. Each input pattern feature has one visible unit in the primary layer, which is composed of visible units and depends on observational components. The observational elements, which are relationships between features, are modelled by the dependencies of hidden units. The entropy function sometimes referred to as an energy-based model, is used to represent the probability distribution of the variables h and v.

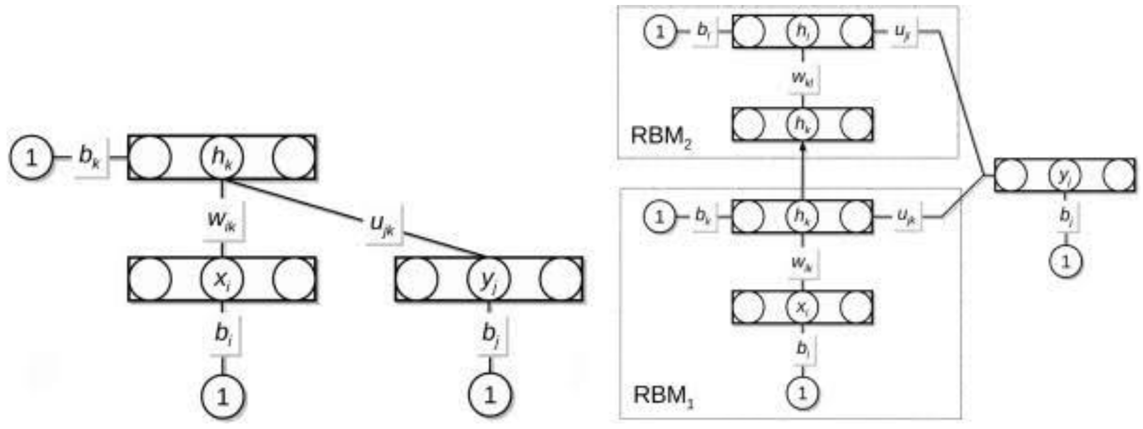


Figure 4: RBM Network for IDS

The function is defined in vectorial format and extensive format by the aforementioned equations:

$$E(v,h) = -h^T W v - a^T v - b^{T_h} E(v,h) = -\sum_{i=1}^{m} v_j h_j w_j - \sum_{k=1}^{m} a_i v_i - \sum_{i=1}^{n} b h_j \quad (11)$$

It is likely to assign probabilities using the entropy function in the hidden and visible layers for each pair of neurons in the network, resulting in the probabilistic distribution shown below.

$$p(v,h) = \frac{e^{-E(vh)}}{\sum_{nh} e^{-E(vh)}} \quad (12)$$

The total of all the vector probabilities from the hidden layer is supplied as the vector probability from v as the visible layer.

$$p(v) = \frac{\sum_h e^{-E(vh)}}{\sum_{\gamma h} e^{-E(vh)}}. \quad (13)$$

The actions are autonomous because RBMs do not include a connection between nearby neurons in the same layer. This provides the calculations for conditional probabilities that are,

$$p(h \mid v) = \prod\nolimits_p \left(h_j \mid v\right)$$
$$p(h \mid v) = \prod p(v_{|} \mid h). \quad (14)$$

The original RBM versions were created to address problems with binary data. Using the proper distribution, (11) and (10), which may be constructed as

$$p\left(h_j = \| \, v\right) = sigm\left(b_j + \sum_{i=1}^{m} v_i w_j\right)$$

$$p\left(v_j = 1hh\right) = sigm\left(a_j + \textstyle\sum_{j=1}^{n} h_i w_j\right). \tag{15}$$

The probabilities of applying the entropy function in the hidden and visible layers are shown by the aforementioned formulae. Let's say the sigmoid function relates to sigma (x). The use of RBM for binary data restricts problem-solving capability. The RBM form is used, allowing subsequent data to improve problem accuracy. RBM of the Gauss type is the most well-known variety. This RBM changes the visible layer probability distribution to a Gaussian distribution. Gaussian-Bernoulli RBM is the term used to describe this disparity. According to a description of the Gaussian-Bernoulli RBM probability distribution,

$$p\left(h_j = 1 \mid v\right) = sigm\left(b_j + \frac{1}{a^2}\textstyle\sum_{i=1}^{m} v_i w_{jj}\right)$$
$$p\left(v_j = 1 \mid h\right) = N\left(a_i + \textstyle\sum_{j=1}^{n} h_j w_{ij}, \sigma^2\right). \tag{16}$$

The RBM training includes reducing the negative log-likelihood provided.

$$\Delta w_j = e\,\frac{\partial log_g(V)}{\partial w_{jj}} = e h_j h_m \tag{17}$$

In order to indicate the desired data rates and model appropriately, the learning rate is denoted by the letters e, d, and m. Whereas (14) and (15) provide expectations for continuous data, (10) and (11) provide expectations for binary data.

3 Performance Analysis

The model is implemented over hardware specifications like Ryzen 5/7 series CPU, NV GPU, 1TB HDD and Windows 11 OS, software specifications like PyTorch, an open-source python library for building deep learning models and Google Collaboratory, an open-source Google environment for developing deep learning model. Experimental evaluations are carried over various models like SVM, VGG16, VGG19, NB, Alexnet, Resnet50 and Googlenet over measures like accuracy, sensitivity, specificity, recall, precision, F1-score, detection rate, TPR, FPR and computation time.

4 Conclusion

To improve the city as a whole, physical, information technology, social and business infrastructures must all be integrated. In order to anticipate danger and recognize network intrusions as well as attacks, machine learning approaches require data training. Threats to cyber networks are frequently evolving faster than the ability of cyber defenders to create and put into use novel signatures to recognize them. There are several possibilities for applying neural network-based deep learning in cyber security applications to precisely identify malware variants when combined with recent developments in machine learning algorithm growth. The paper also describes how to apply DL tactics to a wide spectrum of cyber security threats that target networks, application software, host systems, and information. This paper aids other research specialists to dig deeper and understand various approaches and try to integrate them with other models to bring even more efficiency.

References

[1] Ullah, Z., Al-Turjman, F., Mostarda, L., & Gagliardi, R. (2020). Applications of artificial intelligence and machine learning in smart cities. *Computer Communications*, *154*, 313-323.

[2] Vimal, S., Suresh, A., Subbulakshmi, P., Pradeepa, S., & Kaliappan, M. (2020). Edge computing-based intrusion detection system for smart cities development using IoT in urban

areas. *In Internet of things in smart Technologies for Sustainable Urban Development*, 219-237. Springer, Cham.

[3] Otoum, Y., Liu, D., & Nayak, A. (2022). DL-IDS: a deep learning–based intrusion detection framework for securing IoT. *Transactions on Emerging Telecommunications Technologies*, *33*(3).

[4] Gupta, S.K., Tripathi, M., & Grover, J. (2022). Hybrid optimization and deep learningbased intrusion detection system. *Computers and Electrical Engineering*, *100*.

[5] Kumar, P., Kumar, R., Srivastava, G., Gupta, G.P., Tripathi, R., Gadekallu, T.R., & Xiong, N.N. (2021). PPSF: a privacy-preserving and secure framework using blockchain-based machine-learning for IoT-driven smart cities. *IEEE Transactions on Network Science and Engineering*, *8*(3), 2326-2341.

[6] Maniriho, P., Niyigaba, E., Bizimana, Z., Twiringiyimana, V., Mahoro, L.J., & Ahmad, T. (2020). Anomaly-based intrusion detection approach for iot networks using machine learning. *In IEEE International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, 303-308.

[7] Alsoufi, M.A., Razak, S., Siraj, M.M., Nafea, I., Ghaleb, F.A., Saeed, F., & Nasser, M. (2021). Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review. *Applied Sciences*, *11*(18), 8383.

[8] Reddy, D.K., Behera, H.S., Nayak, J., Vijayakumar, P., Naik, B., & Singh, P.K. (2021). Deep neural networkbased anomaly detection in Internet of Things network traffic tracking for the applications of future smart cities. *Transactions on Emerging Telecommunications Technologies*, *32*(7).

[9] Singh, S., Fernandes, S.V., Padmanabha, V., & Rubini, P.E. (2021). Mcids-multi classifier intrusion detection system for iot cyberattack using deep learning algorithm. *In IEEE Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, 354-360.

[10] Rashid, M.M., Kamruzzaman, J., Hassan, M.M., Imam, T., & Gordon, S. (2020). Cyberattacks detection in iot-based smart city applications using machine learning techniques. *International journal of environmental research and public health*, *17*(24), 9347.

[11] KarsligEl, M.E., Yavuz, A.G., Güvensan, M.A., Hanifi, K., & Bank, H. (2017). Network intrusion detection using machine learning anomaly detection algorithms. *In IEEE 25th Signal Processing and Communications Applications Conference (SIU)*, 1-4.

[12] Singh, R., Kumar, H., & Singla, R.K. (2015). An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications*, *42*(22), 8609-8624.

[13] Wang, H., Gu, J., & Wang, S. (2017). An effective intrusion detection framework based on SVM with feature augmentation. *Knowledge-Based Systems*, *136*, 130-139.

[14] Aloqaily, M., Otoum, S., Al Ridhawi, I., & Jararweh, Y. (2019). An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Networks*, *90*.

[15] Rahman, M.A., Asyhari, A.T., Leong, L.S., Satrya, G.B., Tao, M.H., & Zolkipli, M.F. (2020). Scalable machine learning-based intrusion detection system for IoT-enabled smart cities. *Sustainable Cities and Society*, *61*.

[16] Primartha, R., & Tama, B.A. (2017). Anomaly detection using random forest: A performance revisited. *In IEEE International conference on data and software engineering (ICoDSE)*, 1-6.

[17] Verma, A., & Ranga, V. (2018). On evaluation of network intrusion detection systems: Statistical analysis of CIDDS-001 dataset using machine learning techniques. *Pertanika Journal of Science & Technology*, *26*(3), 1307-1332.

[18] Carneiro, J., Oliveira, N., Sousa, N., Maia, E., & Praça, I. (2021). Machine learning for network-based intrusion detection systems: an analysis of the CIDDS-001 dataset. In *International Symposium on Distributed Computing and Artificial Intelligence*, 148-158. Springer, Cham.

[19] Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *In IEEE military communications and information systems conference (MilCIS)*, 1-6.

[20] Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, *25*(1-3), 18-31.

[21] Manfredi, S., Ceccato, M., Sciarretta, G., & Ranise, S. (2022). Empirical Validation on the Usability of Security Reports for Patching TLS Misconfigurations: User-and Case-Studies on Actionable Mitigations. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 13*(1), 56-86.