PIGNUS: A Deep Learning Model for IDS in Industrial Internet-of-Things

PLS Jayalaxmi[a], Rahul Saha[a,b], Gulshan Kumar[a,b], Mamoun Alazab[c], Mauro Conti[b], Xiaochun Cheng[d]

[a] **School of Computer Science and Engineering, Lovely Professional University Punjab, India**
[b] **Department of Mathematics, University of Padua, Italy**
[c] **College of Engineering, IT and Environment, Charles Darwin University, Australia**
[d] **Department of Computer Science, Middlesex University, United Kingdom**

Abstract

The heterogeneous nature of the Industrial Internet of Thing (IIoT) has a considerable impact on the development of an effective Intrusion Detection System (IDS). The proliferation of linked devices results in multiple inputs from industrial sensors. IDS faces challenges in analyzing the features of the traffic and identifying anonymous behavior. Due to the unavailability of a comprehensive feature mapping method, the present IDS solutions are non-usable to identify zero-day vulnerabilities.

In this paper, we introduce the first comprehensive IDS framework that combines an efficient feature-mapping technique and cascading model to solve the above-mentioned problems. We call our proposed solution *deeP learnIG model intrusioN detection in indUStrial internet-of things (PIGNUS)*. PIGNUS integrates Auto Encoders (AE) to select optimal features and Cascade Forward Back Propagation Neural Network (CFBPNN) for classification and attack detection. The cascading model uses interconnected links from the initial layer to the output layer and determines the normal and abnormal behavior patterns and produces a perfect classification. We execute a set of experiments on five popular IIoT datasets: gas pipeline, water storage tank, NSLKDD+, UNSW-NB15, and X-IIoTID. We compare PIGNUS to the state-of-the-art models in terms of accuracy, False Positive Ratio (FPR), precision, and recall. The results show that PIGNUS provides more than 95% accuracy, which is 25% better on average than the existing models. In the other parameters, PIGNUS shows 20% improved FPR, 10% better recall, and 10% better in precision. Overall, PIGNUS proves its efficiency as an IDS solution for IIoTs.

Thus, PIGNUS is an efficient solution for IIoTs.

Keywords: IoT, Industry, Security, Intrusion, Detection

1. Introduction

The Internet of Thing (IoT) paradigm's evolution elevates the digital era to a new level of pervasive and intelligent connectivity. At the same time, security is a key issue for the connected milieu. Due to computational limitations, skilled attackers can simply bypass the standard security measures and cause significant data loss. Intrusion Detection Systems (IDSs) are currently popular for identifying known attacks based on stored signatures; however, IDSs fall short in discovering zero-day threats. Human independent IDS with automatic detection and prevention can resolve the issue [1]. The growing popularity and outstanding performance of Deep Learning (DL) approaches has a significant impact on application development. Object identification, live face detection, traffic management, discovering

*Rahul Saha(Corresponding author) (email: rsahaaot@gmail.com).

threat, pattern recognition, and medical interpretations are some DL development areas. Optimistic data reduction and accurate assessment of unstructured data are the foremost benefits of DL techniques attracting Industrial IoT (IIoT) applications. The popular smart industrial sector has some security challenges which are detailed in Table 1.

Table 1: Smart Industrial Attacks

Reference	Industry	Attack
Karl Koscher et al. [2]	Telecommunications	Data theft with false messages.
James.P Farwell et al. [3]	Chemical and pharmaceutical production	Stuxnet attack on nuclear power networks - Iran 2010 targeted 60% of computers.
Nicolas Falliere et al. [4]	Plant and machinery	Stuxnet worm attack accessing PLC on Irans nuclear plant.
A.Alvaro Cardenas et al. [5]	Water supply	The destruction of a water utility pump through a SCADA System.
Felix Gomex Marmol et al. [6]	Electricity production and distribution	Abrupt change of power consumption and data theft.
David Edwards et al. [7]	Oil production and supply	Spear-Phishing and Distract attack on (Aramco oil firm) Saudi Arabia, deleted the official information.
Sujit Rokka Chhetri el al. [8]	Power supply and transportation	Jamming attacks to degrade or disable energy supply.
Fei Tao et al. [9]	Gas pipeline and distribution	Ukraine's power grid hack, stolen credentials and shut down 30 substations to access confidential information.

IIoT infrastructure is a collection of interconnected heterogeneous devices including sensors, actuators, processors, network devices, data transfer devices and application controllers. Figure 1 depicts a three-layer IIoT architecture with perception, network and application layers. The perception layer establishes connectivity between sensor and field device and forward for communication. Wireless communication technologies such as 2G, 3G, and Bluetooth are used for data transfer in the network layer. Finally application layer controls the user-end communication. To provide a secure transfer, fundamental security tools and services are embedded into the network layer. Firewalls, intrusion detection systems (IDS) and user authentication are the key security techniques for identifying external threats. IIoT architecture is adaptable based on the researcher's perception and the application. The traditional three-layer IIoT structure is given in Figure1.

1.1. IIoT architecture

Some of the connected IIoT components used to operate the industrial structure are discussed below: Centralized Industrial Control System (ICS) act as an interface between sensors and physical manufacturing devices [10]. The sensors collect information from connected physical components and share it with a centralized source. The Supervisory Control And Data Acquisition (SCADA) component of ICS is used to access external activities. Distributed Control Systems (DCS) manage the shared component services and Programmable Logic Controllers (PLC) configure the industrial infrastructure. Finally, Human Machine Interface (HMI) processes huge data into effective information. The application layer processes the data received from the network devices and manages the services provided to external and internal users [11]. Risk prediction is very high in this type of structure results with data and life threat [12]. Moreover, traditional insecure communications like Modbus and Transmission Control Protocol (TCP) are creating authentication issues in IIoTs.

1.2. Motivation and contribution

Existing ML and DL models cannot detect zero-day vulnerabilities as they compare incoming traffic with out-of-dateed signature patterns. PIGNUS is created to address the increasing security challenge in IIoT applications and analyse the behavioural aspect. PIGNUS made the following contributions to this end.

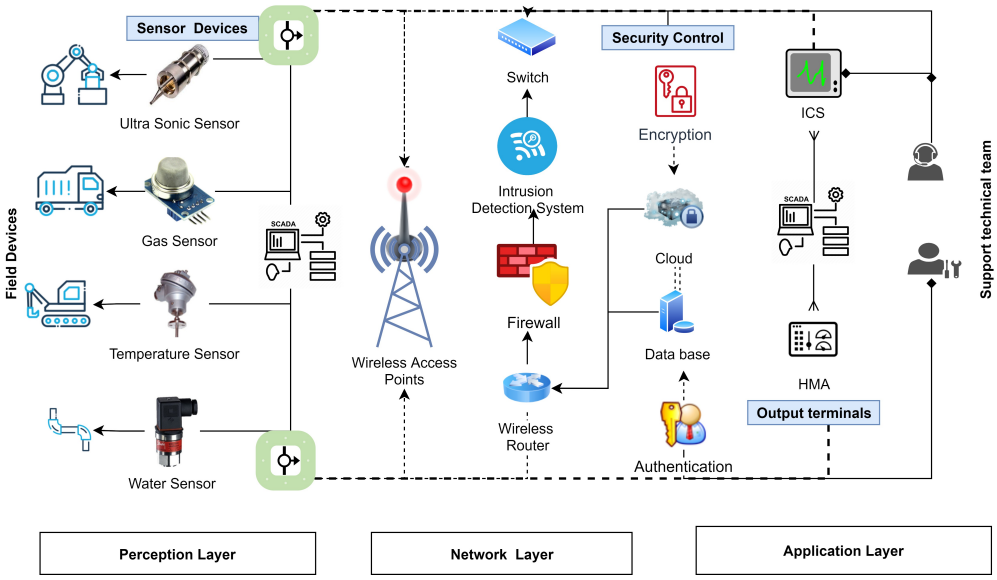


Figure 1: IIoT architecture.

- **Optimal features**: PIGNUS is a hybrid model that incorporates Auto Encoder (AE) for feature selection with detection technique. AE is an unsupervised data compression technique that produces encoded data for further processing. The approach generates discrete values for latent attributes, then forward them for decoding, and produces a compressed dataset. PIGNUS uses the results of AE to train the classification model in order to improve feature mappings.
- **Cascading model**: DL-based Cascading Forward Back Propagation Neural Network (CFBPNN) model in PIGNUS establishes a sophisticated relationship with the raw data using multiple levels of abstraction. The layers are interlinked and each layer receives the output of the previous layer as input. The hierarchical features are extracted to classify the traffic based on the behavioral pattern.
- **Performance**: CFBPNN performs the best among the various DL classification methods currently available. To evaluate the effectiveness of PIGNUS, we compared it with NSLKDD+, UNSW-NB15, and other IIoT datasets. The PIGNUS model is more efficient than the state-of-the-art models based on a comparative analysis.

1.3. Paper organization

We organize the rest of the paper as follows. Section 2 reviews some deep learning-based detection models and their techniques. Section 3 shows the proposed approach of IDS using the combinations of deep learning techniques. We analyze the selected method with evaluation metrics in Section 4. Finally, we conclude our work in Section 5.

2. Related Work

The persuasive quality of DL techniques leverages popular applications in computer vision, bio-informatics, Natural Language Processing (NLP) etc. This imprints a significant growth with efficient performance over Machine Learning (ML) techniques. Self-training and learning methods of DL handle a substantial volume of data with minimum human interaction. IIoT is an extension of IoT to establish industrial applications with sensor connectivity. A review of the research status and the proposals of DL models in both areas is essential.

Artificial Neural Network (ANN), Deep Neural Network (DNN) and Convolutional Neural Network (CNN) are the popular supervised instance learning methods. These techniques are accessed with a feed-forward neural network to develop sequential and image-based detection models [11]. Recurrent Neural Networks (RNN) and the extension Long Short-Term Memory (LSTM) are the popular methods used in IDS [13]. Semi-supervised techniques such

as Restricted Boltzmann Machine (RBM) and Deep Belief Network (DBN) are suitable for training undefined patterns. Transfer learning approaches are also supported by DL for generating generic models that is applied to similar challenges [14]. The prominent DL-based IDS models for IoTs and IIoTs are discussed in the following subsections.

2.1. IDS models using deep learning in IoTs

The primary goal of DL approaches is to create smart and compact models that provide high-end security with minimal resources. As a result, DL-based models are beneficial for large data analytic, video and speech processing, image recognition and building secure IoT applications. Extracting optimal features and developing a detection model is a challenging task. A few integrated models with feature reduction and classification strategies are discussed below.

A DBN model with real-time testing to combine network virtualization with anomaly detection gave 95% accuracy on five attack scenarios [15]. A comparative improvement using DBN for bot attack detection on port scanners by [16] results with 2.8 false rates. An integrated DL model uses Spider Monkey Optimization (SMO) and Stacked Deep Polynomial Network (SDPN) by [17] experimented for three-layer attack detection on huge IoT traffic This model is compared with four Deep Feature Embedding Learning (DFEL) classifiers as Gradient Boosting-based Tree (GBT), K-Nearest Neighbor (KNN), Decision Tree (DT), and Support Vector Machine (SVM). GBT's performance is low comparatively, even then overall precision is 99.38%. Other integrated techniques with Text-CNN and Gated Recurrent Unit (GRU) for feature reduction and conversion by [18] resulted in a high F!-score. A combination of Principle Component Analysis (PCA), Information Gain (IG), Correlation Attribute Evaluation (CAE), and SMO with DNN results with 99.27% F1 score [19]. The best feature set is considered for the experiment. Stand-alone procedures entail time and cost complications; as a solution, hybrid models outperform traditional techniques. LSTM-GRU, a hybrid IDS model for the Internet of Vehicles (IoV), addresses the vanishing gradient problem encountered by RNN in a limited time [20]. CNN-based anomaly detection results in 99% accuracy and provides qualities to examine whole traffic across the IoT [21]. A unique combination of Reptile Search Algorithm (RSA) for feature reduction boosts the anomaly detection accuracy with CNN [22]. All the detection models discussed above have a strategy to reduce memory usage, improve the detection rate, reduce the false rate, and trace the new attacks. Table 2 summarises the most recent DL-based IDS for IoTs.

Table 2: Review on DL-based IDS models in IoT

Author and Reference	DL Technique	Dataset	Accuracy (%)
Thamilarasu et al. [15]	DBN	Real-time	99.50
Manimurugan et al. [16]	DBN	CICIDS 2017	98.37
Yazan Otoum et al. [17]	SMO, SDPN	NSLKDD+	99.30
Tabassum et al. [18]	AE,CNN	NSLKDD+, Real-time	99.90
Muneeba Nasir et al. [19]	DFS with DNN	real-time	99.03
Safi Ullah et al. [20]	LSTM,GRU	CSE-CICIDS 2018	99.50
TanzilaSaba et al. [21]	CNN-AIDS	BoT-IoT	92.85
Abdelghani Dahou et al. [22]	CNN,RSA	BoT-IoT	99.91
Ming Zhong et al. [23]	Text-CNN,GRU	NSLKDD+	98.90
Wumei Zhang et al. [24]	Stacked Sparse AE	NSLKDD+	95.42

2.2. IDS models using deep learning in IIoT

Collaborative smart industrial technology with integrated IoT devices has numerous benefits with effective productivity; however, their online nature makes them vulnerable to cyber-attacks. Due to attack upgrades, traditional firewalls and antivirus software cannot address the security gaps caused by the complex structure of the smart factory. Thus, the situation induces a high risk of device proliferation resulting with direct or indirect intrusions. The researchers propose various ML and DL techniques to build an effective IDS system. In this section, we explore some of the models and the challenges faced by each.

The Deep Feed-Forward Neural Network (DFNN) and Deep Auto-Encoder (DAE) based anomaly detection system developed by [25] has a distinctive style of training using supervised and unsupervised techniques. The experiment on the NSL-KDD dataset results in 1.4 false rates compared to UNSW-NB15. The researches show a hybrid

model experimenting on the CICIDS dataset and big data for IDS. The combination of Random Forest (RF) and GBT for feature selection, and DNN for detecting multi-class attacks performs well in literature [26]. A feed-forward neural network model illustrates high classification accuracy. The model is tested using a new dataset generated with generic features at the packet level. The model identifies Denial Of Service (DoS), Distributed DoS (DDoS), reconnaissance, and information theft attacks with improved performance [27].

DNN with ML-based classifiers trained to learn abstract and high-dimensional IDS features in IIoT performs well on benchmark dataset [28]. Further, a multi-CNN fusion model for IDS to capture graphical intrusions [29] tested on KDDtest+ and NSLKDD+ resulted in 13.5% false rate. The model divides the dataset into four parts. The one-dimension dataset is converted into a gray-scale graph using the flow data visualization method for dimensional reduction. The subset is processed with the CNN model for detection. On real-time graphical datasets, other integrated models are significantly practical for IIoT environments [30].

The researcher tests the Ensemble-learning model with the combination of Random Subspace (RS) and Random Tree (RT) with 15 SCADA IIoT network datasets. The RS learning method solves the sensitivity of irrelevant features, whereas RT reduces the over-fitting problem encountered in IIoTs [31]. Feature normalization and identification of patterns are more important in intrusions. A balanced representation of the imbalanced datasets processed with an ensemble model using DNN and DT by [32]. The model evaluates with 10-fold cross-validation on Gas Pipeline (GP) and Secure Water Treatment (SWaT) resulting apt for industrial structure. Integrity features-based DL prediction model using Sparse Evolutionary Training (SET) results in 6.25% improved accuracy [33]. We summarize the latest DL-based IDS models for IIoTS in Table 3 .

Table 3: Review on DL-based IDS models in IIoTs

Author and Reference	DL Technique	Datasets	Accuracy (%)
Muna et al. [25]	DAE and DFNN	NSLKDD	98.06
Osama Faker et al. [26]	DNN, GBT	UNSW-NB15	99.99
Ge. Mengmeng et al. [27]	FNN	BoT-IoT	98.02
Vinay Kumar et.al. [28]	DNN	KDDCup 99	92.90
Yanmiao Li et al. [29]	Multi-CNN	KDDtest+ 21	86.95
Eric Gyamfi et al. [30]	OI-SVDD	UNSW-NB15	95.02
Mohammad M.H et al. [31]	RS,RT	15 SCADA test-bed	96.71
Al-Abassi et al. [32]	DNN,DT	GP,SWaT	99.67
Shahid Latif et al. [34]	DRaNN	UNSW-NB15	99.54
Sarika Choudhary et al. [35]	DNN-IDS	UNSW-NB15	90.02
Zhihong Tian et al. [36]	Multiple concurrent DL	CSIC 2010,	99.41
A.Mendon et al. [33]	SET prediction	Real- time	99.02
Beibei Li et al. [37]	CNN-GRU	Industrial CPS	99.20
Awotunde et al. [38]	DFFNN and DAE	UNSW-NB15	98.91
OthmaneFriha et al. [39]	DNN,CNN,RNN	CSE-CICIDS 2018	99.01
S.Tharewal et al. [40]	DRL-IDS	Gas pipeline	99.10

The model combines Online Incremental Support Vector Data Description (OI-SVDD) with Adaptive Sequential Extreme Learning Machine (AS-ELM) and is tested on Multi-Access Edge Computing (MEC) server [30]. A real-time data stimulation and attack modules introduced for reinforcement learning by [40] tested with Natural Gas pipeline transportation. This model gave very marginal false rate.

An attack scenario with the administrator, user, and attack modules is launched for testing. The author proposes a light-GBM technique to select the optimal features and a PPo2 algorithm with ReLU for detecting the attacks. The model performs well compared to traditional DL techniques such as CNN, RNN, LSTM, and DQN [40]. From the above mentioned literature study, we observe that none of the models project 100% accuracy for a single or multi-class attack detection; thus, it provides a scope of improvement. Considering this, we integrate an AE feature reduction technique with a cascading IDS model. This provides better detection accuracy for the anomalies available in the industrial network.

3. Proposed model: PIGNUS

We propose, deeP learnIG model intrusioN detection in indUStrial internet-of things (PIGNUS). PIGNUS signifies security in Latin. PIGNUS in a novel DL-based IDS model with the combination of Auto Encoder (AE) and Cascade Forward Back Propagation Neural Network (CFBPNN). The available models have an average accuracy of 95.00%; however, none of the existing works examine AE base compression with cascading classifier. Our PIGNUS addresses this issue and emphasizes the significance of feature extraction with a comparative analysis on traditional and AE-based cascade structures. In this section, we present our preliminary knowledge of the DL techniques, then focus on the assumptions made to construct PIGNUS. The methodology starts with full description of the datasets used in the experiment followed by normalisation procedure. The application of AE in feature extraction, and the usage of CFBPNN for detection are successively explored in this section.

3.1. Preliminary understanding

Figure 2 depicts the internal structure of NN with input, hidden, and output layers. The data is supplied into the neurons for the input layer in the form of numbers/images/audio. These input features are represented as x_1, x_2, x_3,..., x_n. The process multiples each input by weights ($w_1, w_2, w_3, ..., w_n$) and passes to an activation function. An activation function is a step function that maps the input and output signals for network functioning. Equation 1 depicted the mathematical representation of a NN. In this Equation 1, x represents an input, w represents a weight added for each input, z is used for output, b represents bias, and f represents the activation function.

$$z = f(b + \sum_{i=1}^{N} x_i w_i). \tag{1}$$

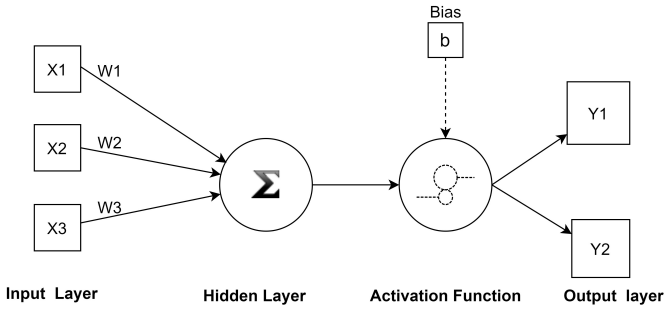


Figure 2: Structure of a NN

The feed-forward algorithm begins with the input layer moving in the forward direction to update the state of each unit [27]. This model multiplies the weights and add the bias; this process repeats till all the layers are updated. The model adjusts the weights and performs the task to improve the accuracy using back propagation. Though, the process is generic, we observe some variations of the structure based on the application requirements.

3.2. Assumptions

Considering the shortfalls of the existing detection models, we observe that the security module extended to multiple levels will be more predictive. The results of single-level detection and prevention techniques are only reliable for a short period of time [14]. To justify the multi-level detection model, a three-level security system is beneficial. A framework with a risk factor provide suitable security solution. We show the assumption of the security model in Figure 3. The first level of security is formed with basic firewall protection. The security services are applied to verify the authentication of the users and control unauthorized access in the second level. Finally, a DL-based IDS model is a required to observe the network traffic and report the suspicious activities.

Figure 3: Model Assumption

3.3. Methodology

This section provide a elaborated description of the methods integrated in PIGNUS. The traditional architecture of the IDS model is prone to security leaks. The multi-layer recursive structure analyzes the data at various levels and makes the model effective to handle minute complications. IIoTs have multiple frameworks with interconnected devices and sensor operations. A single-layered model lacks in generating appropriate sequences and drops in performance. Such models are restricted by the scope of the connected components [41]. The multi-layer model is distributed across the system and executes the processes at each level. The major and minor values are converted based on the state of the system. A self-learning model minimize human instructions and mitigates intrusions given from input sequences. We show the flowchart of the methodology in Figure 4.

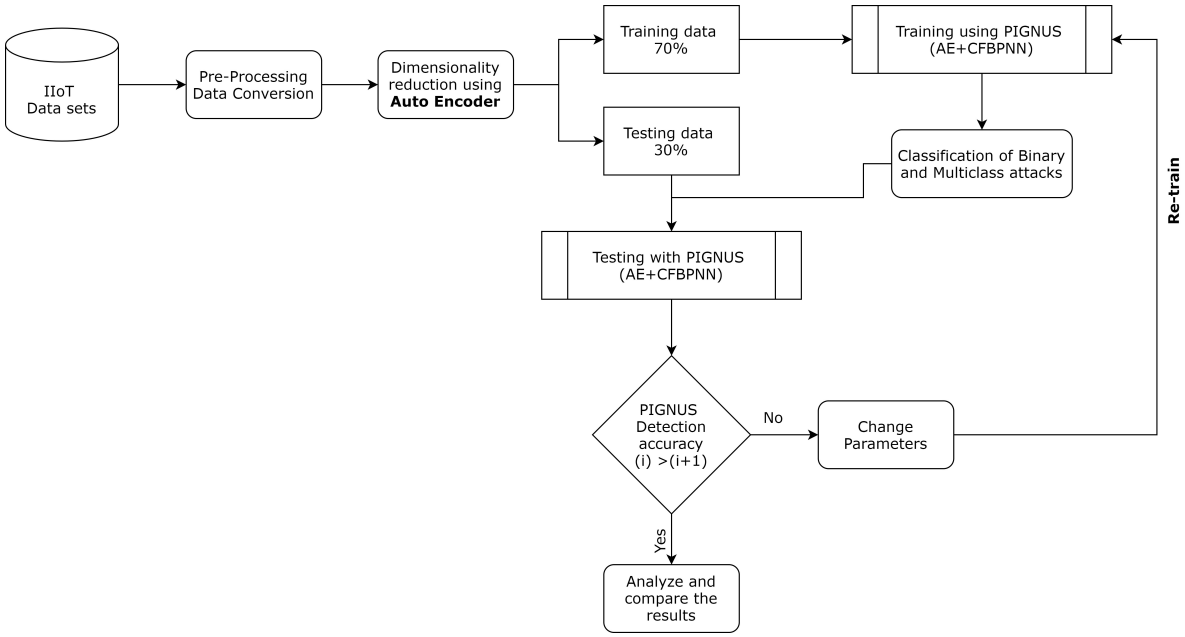


Figure 4: Methodology of proposed model

To solve the above-mentioned issues we provide PIGNUS, a hybrid model with the combination of AE and CFBPNN. AE builds an optimal feature set and CFBPNN traces the attack pattern based on predefined signatures and identify the attack variant with the interrelated link.

3.4. Dataset

We conduct a comprehensive series of experiments on PIGNUS with five different datasets: i) UNSW-NB15 [42], ii) ICS generated water storage tank dataset [43] , iii) gas pipeline dataset [44], and iv) NSLKDD+ dataset [45] v) X-IIoTID dataset [46]. All five datasets have distinctive features; however, some features are common in all, such as protocol, command-address, command memory, response count and write function length, cycle time, control mode, time, and attack class.

To check the applicability of PIGNUS, we use laboratory-scale ICS data from water storage tank dataset [43] and industrial gas pipeline dataset [44]. These are IIoT datasets specially used for evaluating AI-based cyber security applications. The water storage tank dataset contains pre-processed network transaction data with 2,36,180 samples in which 1,72,415 are normal and 63,764 are attack values. The gas pipeline dataset contains 10,619 samples, where 6,672 are normal and 3,947 are attack values. We consider 10% of each dataset as sample for testing the experiment. The gas pipeline dataset provide 27 features and the water storage tank dataset have 24 features; both datasets reflect seven attack categories.

NSLKDD+ cup dataset is the most popular dataset for IDS with huge categories of attack signatures. This dataset is prepared using the network traffic captured by the 1998 DARPA IDS evaluation program [45]. In our present research, we use NSLKDD+ dataset, which overcomes data redundancy and provides updated attack profiles over the traditional KDDCUP dataset. The training dataset contains a total of 125973 records of which 58630 are attack values and 67343 are normal records. The dataset have 41 labeled input features of binary and multi-class attack classification. A total of 38 traffic classes with 21 attack classes are available in the test data; we consider 16 attacks and one normal class for training. The attack records are grouped into four major classes including DoS, probing, User-to-Root (U2R), and Root-to-Local (R2L). The descriptions of each attack instance for gas pipeline, water storage and NSLKDD+ dataset is given in Table 4.

Table 4: Attack types and number of records in IIoT and NSLKDD+ dataset

Gas Pipeline (GPl) and Water Storage tank(WSt)			NSLKDD+	
Type	Samples (GPl)	Samples(WSt)	Type	Samples
Normal	19503	6672	Normal	67343
Naive Malicious Response Injection	1198	335	DoS	45927
Complex Malicious Response Injection	1457	1664	Probe	11656
Malicious State Command Injection	209	93	R2L	995
Malicious Parameter Command Injection	410	842	U2R	52
Malicious Function Code Injection	155	41		
Denial Of Service	135	189		
Reconnaissance	4132	783		

UNSW-NB15 dataset contains raw network packets created by the IXIA perfect storm tool in the cyber range lab of the Australian Centre for Cyber Security (ACCS). The dataset provide nine types of cyber-attacks, 45 input features and two million, 540,044 instance stored in four separate CSV files. A split of the dataset with 175,341 training and 82,332 testing instance with multi-class attack variants is considered for the experiment. The dataset consists of 56000 normal values and 119341 records with nine attack categories. The generic attack type is with 40000 records and exploits count with 33393 records. According to the literature review, higher attack records aid towards enhancement of model accuracy. Considering this, we have used UNSW-NB15 and X-IIoTID datasets to train and test our model with a high attack instance. In binary classification for this dataset, PIGNUS produces zero false rate.

The X-IIoTID dataset provides real-time labelled IIoT data that exposes host and network processes in both safe and vulnerable environments [46]. This dataset estimates the attack strategies using statistical, machine learning, and deep learning techniques. IIoT suitable features are extracted from log files and network traffic using device resources and commercial IDS logs as (OSSEC and Zeek/Bro). The X-IIoTID dataset has 820,834 instances, with 68 features out of which 421,417 are normal and 399,417 attack observations. The dataset has three label levels representing attack scenarios. Class one provides a binary category, and class two supports normal and 18 sub-categories of attacks and finally class three with normal and ten sub-sub attack categories. We chose class 3 as the sample set for our experiment. The detailed attack instance for UNSW-NB15 and X-IIoTID dataset is given in Table 5. To demonstrate the impact of a model within a time limit, the full dataset is used for experiment PIGNUS on binary class. Ten percent of instances are selected for multi-class training and classification.

Table 5: Attack types and number of records in UNSW-NB15 and X-IIoTID dataset

UNSW-NB15		X-IIoTID	
Type	Samples	Type	Samples
Normal	560000	Normal	421417
Analysis	2000	C and C	2863
Backdoor	1746	Crypto-Ransomware	458
DoS	12264	Exfiltration	22134
Exploits	33393	Exploitation	1133
Fuzzers	18184	Lateral-movement	31596
Generic	40000	RDOS	141261
Reconnaissance	10491	Reconnaissance	127590
Shellcode	1133	Tampering	5122
Worms	130	Weaponization	67260

3.5. Data normalization

As the first stage of normalisation, the fill missing function is used to replace empty values with constant and standard values. In the second step, we convert all the categorical values (NaN) into numerical identities for easy prediction using one hot-encoding technique. This technique processes the categorical variable and converts it into a numerical representation. At the same time, natural ordering between categories with integers may result in poor performance or unexpected results; we convert the string values to a new binary variable and add unique integer value for each attack. The detailed implementation is given in Algorithm 3.

The training dataset provides several attack classes, while the validation dataset provides new attack classes that were not initialized in the training dataset. In this way, we can test whether the trained model can identify new samples besides the ones that were previously trained. To prevent over- or underfitting, we adjust some model parameters based on the accuracy of the predictions after validation. Overfitting occurs when a model attempts to fit all of the training data and ends up retaining the data patterns, noise, and random fluctuations. A model's overfitting issue prevents it from generalising and making good use of unanticipated data circumstances. Due to excessive bias in the data and oversimplification of the issue, an under fitted model does not perform as expected in a training set of data. Our study PIGNUS focuses on both binary and multi-class classification, where we represent normal values as 0 and attack values as unique integers based on classification or 1 in case of binary attack.

3.6. Feature extraction with Auto Encoder (AE)

After the data is normalized, the next step is to extract the best features for training the model. AE is used to improve training efficiency and speed up detection as part of dimensionality reduction. We use the encoded values as input for detection models.

AE is an unsupervised learning technique used for a compressed representation of raw data. AE reduces the given input into the lower-dimensional format and regenerates the output as a new representation. To replicate the input vector against the output layer and train the AE model, we implement a back-propagation algorithm. For a given input x and reconstruction result as $\widehat{x}$ the network is trained by minimizing the error $L(x, \widehat{x})$ to measure the variation between the original input and the encoded output. We train AE with 25 hidden layers using the scaled conjugate gradient training algorithm. Most activation functions used in AE are non-linear, such as ReLUs (Rectified Linear Units) and sigmoid functions. The model performance is evaluated using Mean Square Error (MSE) with L2 sparsity regularize. Based on our experiments the MSE for PIGNUS is 4.56%, the lowest among all five datasets we used. To prevent over-fitting additional information is given to the model in the process of regularization. L2 regression is also considered as ridge regression, represented with the Equation 2. We represent the loss function with L2 norm of the weights given in Equation 3.

$$\widehat{x} = w_1 + x_1 + w_2 x_2 + \ldots + w_n + x_n + b. \tag{2}$$

AE minimizes the difference between the input and output; we identify the loss function given in Equation 3.

$$Loss = Error\,(x, \widehat{x}) + \lambda \sum_{i=1}^{N} w_i^2. \tag{3}$$

In the above expression for an AE model $\widehat{x}$ with x as input variables, w represents the weight and b represents the bias. We use a loss function to analyze the difference between the true and predicted values. $\lambda > 0$ represents the regularization parameter and σ represents the total calculated loss and predicts the efficiency of the model for each input and added weight. The neurons are "inactive" if their output value is close to 0 and active if it is close to 1. We invoke sparsity parameters to make the neurons active and avoid overfitting issues. We observe that the average activation of each hidden neuron is close to itself, or a value close to zero [47].

We summarize the sequences of AE model in Algorithm 1.

Algorithm 1 *Autoencoder*(X)

Initialize transpose table X
$\phi : X \rightarrow F$
$L = hiddenlayers$
Activate w, b weight and bias with random values
$performance = MSE$
$trainAutoencoder(X, L)$
$\psi : F \rightarrow X$
$i = 1$
while $i < L$ **do**
  while $Epochs = 1000$ **do**
    Train M sample $X_1, X_2, ...X_M$.
    $\phi : y = \sigma\,(W_X + b)$
    $\psi : y^{'} = \sigma^1\left(W^{'}y + b^{'}\right)$
    $MSE = \frac{1}{N}\sum_i^N \sum_j^M \left(y'_{ij} - y_{ij}\right)^2$
    $Loss = Error\,(x, \widehat{x}) + \lambda \sum_{i=1}^{N} w_i^2$
  end while
end while

The decoded output of AE is passed as an input argument for the classification model. The detailed procedure of the detection model is given section 3.7. The training of the network is divided into the encoder and the decoder segment. AE function is activated with x as input arguments from the $PIGNUS()$. A latent space F is created by mapping the original data x to an encoder method $/phi$. To repeat the process we have to initialize hidden layer then activate L weight w and bias b with random values. Next set the performance indicator to MSE. Then the AE network is created with x input features and L hidden layers with the method $trainAutoencoder(X, L)$. The decoder module by ψ maps the latent space F. AE output is the same as the input function. The original sample size is recreated with decoded content. The network construction is stated and the procedure is repeated for 1000 epochs. The encoded NN ϕ function pass through an activation function σ, where y is the latent dimension. The decoding NN ψ represent in a similar method, with different weights and bias. Final the performance of the model is evaluated based on the loss function and MSE. MSE and loss is calculated for N rows $N_i, N_i + 1....N_L$ and M columns$M_j, M_j + 1....M_L$ with the decoded values y, y'. Finally the output generated from AE : y is passed as input argument for $CFBPNN()$ method given in Algorithm 2.

3.7. Detection process

Traditional machine learning approaches are effective in detecting suspicious patterns in network traffic [41]. At the same time, working with pre-defined clusters, center point initialization, and selection of maximum and minimum radius for the clusters are some of the drawbacks of the traditional ML methods [48]. Low performance with inappropriate feature mapping and clustering are the outcomes of such models. Our PIGNUS leverages the capability of ANN

and forms inherited clusters to identify the instance of normal and abnormal behavior. This constructs boundaries between different clusters and helps to identify the most optimistic features suitable for complex architecture. A cluster model is a type of feed-forward neural network, which is trained in a supervised manner with back propagation. The method directly classifies the given input vector based on the specified target by matching it with the previous performance and repeat till it reaches the threshold. The method is superior to other clustering and feed-forward techniques as it adds a pre-clustering stage [49]; thus, PIGNUS training avoids the curse of dimensionality, which occurs when the feature space of a fixed-size gets progressively sparse as the number of dimensions increases.

NN supports multiple algorithms for classification problems. According to our study, we select Cascading Forward Back Propagation Neural Network (CFBPNN) method to classify the anomaly and distinguish the attack and normal packets. CFBPNN integrates Feed Forward (FF) and Back Propagation (BP) techniques to form the network structure. FF method consists of a single input layer, multiple hidden layers and selected output layers. The BP technique operates as a learning algorithm to train network models by calculating error values, adjusting weights, and transmitting to the previous layer. Because of the non-linear transfer function utilised in many layers, the model learn both linear and non-linear relations between input and output vectors [50].

Connecting the input weights from each successive layer is the process of a CFBPNN model. Networks with multiple layers have the potential to learn the complex relations among input and output vectors. CFBPNN starts with a single input layer and gradually adds numerous connected layers, which receive connections from the original input layer and all previously hidden units. In order to shape the connection, we combine a direct link from input neuron to a hidden layer [51]. We add the perceptions one by one in the correlation; it starts with a small number and ends up with a bigger size. Additional connections improve the speed and learning rate. When the net performance is greater than 99 % and there are no NAN results for any of the attack classifications, the process is terminated. We describe the mathematical expression of CFBPNN in Equation 4. CFBPNN clusters the combination of sequence and recurrent learning methods to connect initial inputs and their relationships to infer unseen connectivity. Thus, CFBPNN is beneficial to generalize and predict anonymous data.

$$y = \sum_{i=1}^{n} f^{i} w_{i}^{i} x_{i} + f^{0}\left(\sum_{j=1}^{k} w_{j}^{0} f_{j}^{h}\left(\sum_{i=1}^{n} w_{ji}^{h} x_{i}\right)\right). \tag{4}$$

In Equation 4, $\int$ represents activation function, w represents the weight from input to output layer, i is the number of iteration, and y is for the output layer. We add the bias to the weights and sum with the previous value till k^{th} layer. For a given n sample, we represent $\int$ the sigmoid activation function. This is added with the weights w and bias for each iteration of i to the n^{th} value in Equation 4. We first create a simple connected network with a single input and output unit, and initialize the variable from 1 to n as $x = \sum_{i<=n}^{i=1}$. We use a regression matrix to construct R as shown in Equation 5. We summarize the sequence of CFBPNN model with PIGNUS in Algorithm 2.

$$R = f^{0}\left(w^{b} + \sum_{j=1}^{k} w_{j}^{0} f^{h}\left(w_{j}^{b} + \sum_{i=1}^{n} w_{ji}^{h} x_{i}\right)\right). \tag{5}$$

The detailed approach of the CFBPNN model is shown in Algorithm 2. The input and target values are provided from $PIGNUS()$. Next we create the CFBPNN with X input, Y target and L hidden layers. Set the input arguments X to n rows and ni columns and initiate accuracy to 0. The Sum of weights and bias are calculated for each hidden layer from $X[i]$ to $X[L]$ by adding the bias of each layer $b[1]...b[L]$. We repeat every iteration internally by forwarding the values from the initial layer to the active layer in *step* 8. Next, calculate the sum of weights for each input layer. The result is passed to activation function f, and the process repeats until the last layer is reached. The process is terminated if the input-cell value is empty, else the delay unit is added and incremented to the next layer. The model performance is calculated using accuracy and MSE. The process of calculating weights and updating the next layer is repeated till the validation is completed or the accuracy of previous and active iterations remains static. Finally, the model returns the values of MSE, accuracy, Regression (R), and output y as the prediction result. We show the overall procedure of PIGNUS in Algorithm 3.

Algorithm 2 $CFBPNN(input, target)$

$X = Input, Y = Target, L = HiddenLayers.$
Create $y = cascadeforwardnet(X, L)$.
Initialize $i = 1$, $j = 1$; Set X_i^j .. X_m^n ..
$Accuracy = 0$;
while $i <= L$ **do**
  while $Accuracy(i) <= Accuracy(i+1)$ **do**
    $X_i + W_i, b_i ....X_j + W_j b_j.$
    $\mathrm{Y}=\sum_{i=1}^{n} \sum_{n}^{w} x_i^j$
    $y = \sum_{i=1}^{n} f^i w_i^i x_i + f^0 \left(\sum_{j=1}^{k} w_j^0 f_j^h \left(\sum_{i=1}^{n} w_{ji}^h x_i \right) \right)$
    $Accuracy = \sum(Y = X)/N(X, L) * 100;$
    $\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \widehat{y})^2$
    $R = f^0 \left(w^b + \sum_{j=1}^{k} w_j^0 f^h \left(w_j^b + \sum_{i=1}^{n} w_{ji}^h x_i \right) \right)$
    $i = i + 1$
  end while
end while

As PIGNUS starts with data loading and pre processing, later combines Algorithm 1 for AE and Algorithm 2 for CFBPNN respectively. Algorithm 3 starts by importing the dataset from external sources. The read-table method opens the file specified as a parameter and then stores the contents of the variable x with input features. The DL models are trained row by row; the dataset is transposed and indicated with x'. The *Autoencoder* method is called with x as the argument value and the decoded result is stored in variable $Table$. Next supervised training is implemented for classification with labeled data values. The target variable (attack type) provides multi-class string labels. This variable is converted into numerical values using one-hot encoders. Table data is assigned to input variable. Next to split the dataset, random function with *input* m and *ratio P* is activated. The split include 70% training and 30% testing instance for this experiment. Finally given input and target arguments invoke CFBPNN the detection module.

Algorithm 3 Proposed PIGNUS

$X = readtable(filename.csv)$
Transpose table $X = X'$
$Table = Autoencoder(X);$
$input = Table$
Initialize $target = categorical(attack)$;
Table conversion $target = table(target)$;
$target = onehotencode(target)$
Initialize Row m Column n
Assign input to matrix $[m, n] = size(input)$
Splitting data :$Train, Test$
Initialize ratio $P = 0.70$
$Var = randperm(m)$
$Train = Table(var(i = 1 : (P * m)))$
$Train = Table(var(P * m) + 1$ to N)
$input = Train$
$CFBPNN(input, target)$

4. Experimental analysis

In this section, we first describe the procedure for the experimental setup. We execute PIGNUS on an I5 processor (16 GB RAM and 1 TB Octan memory) with Windows 10 operating system using MATLAB R2019b environment. Then, we define the evaluation parameters and finally, discuss the results.

4.1. Experimental setup

We implement and evaluate our proposed PIGNUS model using MatLab R2019a Simulator. Using five different datasets mentioned in Section 3.4, we experiment AE for conversion. We include UNSW-NB15 dataset with 175341 records, water storage tank dataset with 236180 records, 10619 samples from the gas pipeline dataset, 1025973 samples from the NSLKDD+ dataset, and finally X-IIoTID dataset with 820,834 instances for our study with random split.

We adopt the network model with ideal parameters, which produce the highest accuracy and lowest false rate after repeated experiments. We train the model using CFBPNN with the best network structure on all the five datasets. The network structure contains one input layer with different nodes based on the total input features given by the dataset. The experimental method provides 27 input nodes for the gas pipeline dataset, 24 for the water storage tank, 45 for UNSW-NB15, 42 for the NSLKDD+ dataset and 68 for X-IIoTID dataset. It also includes five hidden layers (10 nodes each) and output layer (1 node) indicating the status in multi-class procedure. We represent the network structure of CFBPNN model with 27 input units, four hidden layers and seven output layer each with 10 neurons and 7 attack classes as output given in Figure 5.

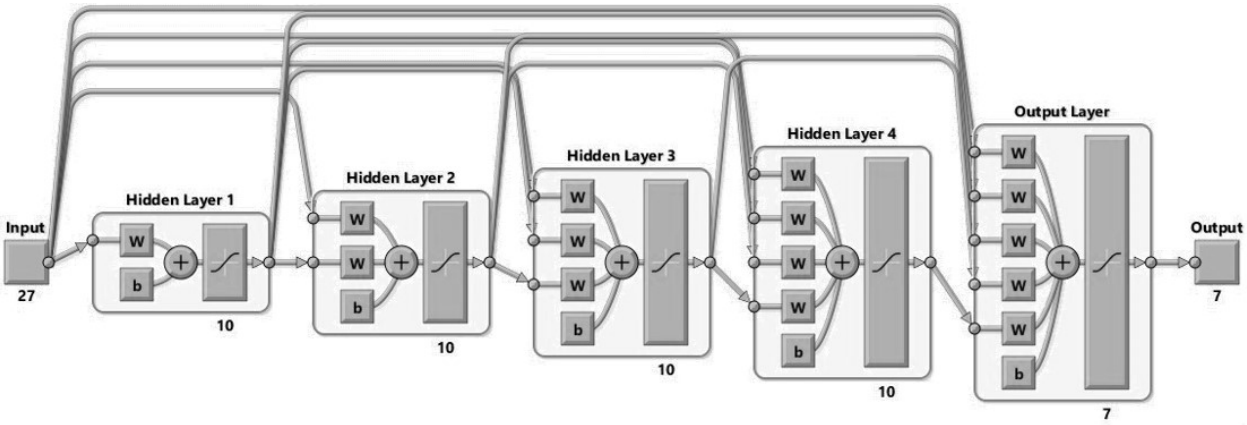


Figure 5: Network structure of CFBPNN

Cascading model is applied for all five datasets with common parameters, and variation in input size based on features provided. The internal network mask of the model with five layers and interconnected nodes is given in Figure 6. The internal architecture of the CFBPNN model for the first input layer is indicated as *process input 1* with five hidden layers $Layer_{1..5}$. We use the transfer function, $Tran - sigmoid$ [34], for each layer to calculate the weights and bias received from previous units $a1....to....4$ performed before the hidden layer. The elements a_1, a_2, a_3, and a_4 carry the weights for the next layer represented after the hidden layer structure as shown in Figure 6. Finally, we represent the classification output with *process Output* with the input layers.

We use 1000 epochs for the transfer function *sigmoid* with a learning rate 0.002, minimum performance gradient $1e - 07$, decrease factor for μ with 0.1, and increase factor for μ with 10 for all the datasets with L1 and L2 regularization. Sigmoid transfer function structure is given in Figure 7b this is applied with Levenberg-Marquardt (LM) training method. LM is the fastest method for training a moderate-sized network and specially designed to approach second-order training speed without hessian matrix. The performance is indicated with Mean Square Error (MSE). Gradient Descent Momentum (GDM) is activated as adaption learning function with learning rate ($LP.lr$) – 0.01 and

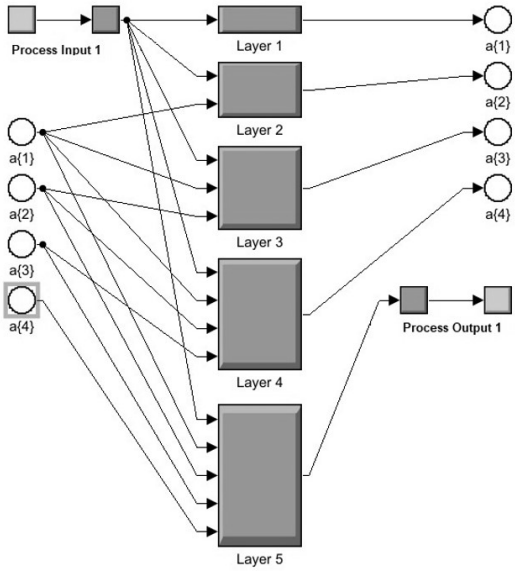


Figure 6: CFBPNN internal network architecture

momentum constant ($LP.mc-$)0.9, we show this expression in Equation 6.

$$dw = mc * dwprev + (1 - mc) * lr * gw. \tag{6}$$

From Equation 6, GDM calculates the change in weight (dw) for a selected neuron from the input P and error E with weight W. We represent the learning rate with lr, and the momentum constant with mc, gradient weight with gw. We use this to test the performance with gradient G for a weight going to the next layer. The learning rate and learning state is updated by recording the prior weight changes ($dw - prev$) and implements repeatedly.

Figure 6 represents the initial input layer $a1$ connected to each hidden layer; PIGNUS represents five hidden layers and one output layer. Every hidden layer is again connects to the next input carrying the sum of weights and bias. Figure 7a shows the structure of internal layer with added weights, bias, delay units and the activation function processed to the next input. We train our PIGNUS and evaluate its performance in detecting binary and multi class attack variants and visualize with confusion matrices.

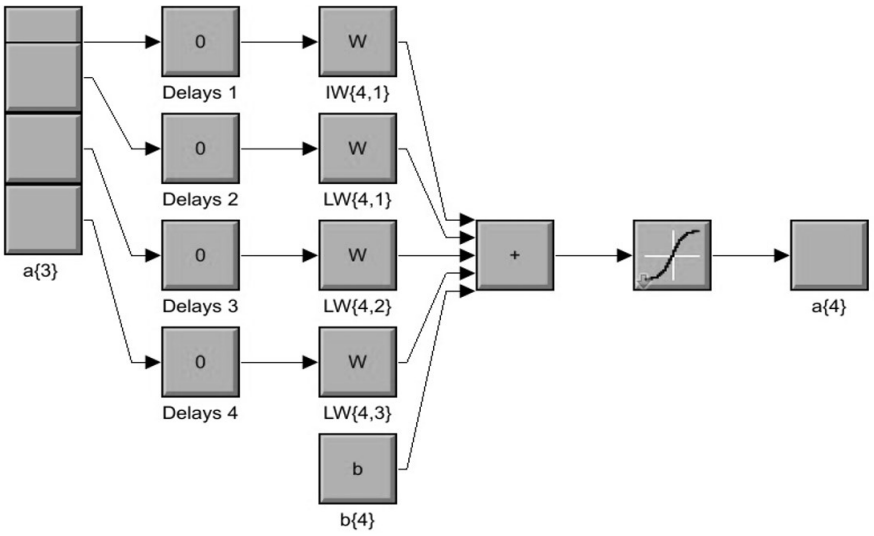
4.2. Evaluation metrics

The best method to evaluate the effectiveness of the classification model is a confusion matrix. This makes it easier to distinguish between true and false instances for training and testing. The results of this technique identifies the types of errors encountered by the model in the process of training. We analyze the number of incorrect predictions for each class assigned to the model with the target variable. The confusion matrix projects the difference in the prediction and actual assumptions. The elements of the confusion matrix are used to construct the accuracy of the overall model. The confusion matrices for all the datasets is given in Figure 8. In the following paragraphs, we define the critical metrics we consider for the review process.

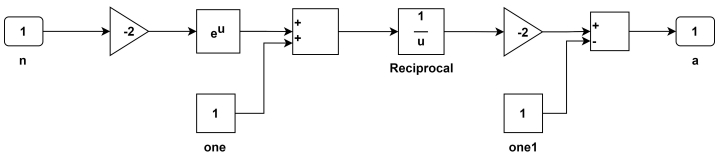
False Negative Rate (FNR): FNR returns the ratio of false cases marked as true and anomalies considered as normal activity in the network model or a missed alarm rate in detection. Our proposed PIGNUS has 0.00% FNR for binary class and 0.22% for the multi-class as an average of all the test samples. The values indicate the number of anomaly cases considered true is non-descriptive. We use Equation 7 to calculate FNR.

$$FNR = \frac{FN}{TP + FN}. \tag{7}$$

False Positive Rate (FPR): FPR indicates the presence of attack records in the network package which is identified as normal. We use Equation 8 to trace the false cases. This is a percentage of incorrect results classified. PIGNUS



(a) Internal structure of input layer of CFBPNN model



(b) Structure of Sigmoid Transfer Function

Figure 7: Structure of CFBPNN internal layer and training function

shows a 0% false rate for the binary class that indicates no count for wrong interpretation encountered.

$$FPR = \frac{FP}{TP + FP}. \tag{8}$$

Accuracy (A): We represent the ratio of correctness for the classified in Equation 9. PIGNUS is proved to be accurate for selected samples. The accuracy of the model is 100% for binary and multi-class with selected samples.

$$Accuracy(A) = \frac{TP + TN}{TP + FP + FN + TN}. \tag{9}$$

Precision (P): The ratio of true positive samples to the predicted positive samples is known as precision. We use P to represent the confidence of attack detection as in Equation 10. PIGNUS results in a 0.01% precision value, which indicates that the identification of a normal sample to a similar class is more appropriate.

$$P = \frac{TP}{TP + FP}. \tag{10}$$

Recall (R): We use recall to represent the ratio of true positive values to the total value with Equation 11. We can consider this as the detection rate and use in IDS evaluations. R reflects the model's ability to recognize the attacks from a given class. R-value for the proposed model represents that it can classify the attack category to 0.99% accurately with 0.01% error rate.

$$R = \frac{TP}{TP + FN}. \tag{11}$$

4.3. Results and comparative analysis

The goal of PIGNUS is to map the relationships between input and target values and improve the performance of the detection. We combine several threshold function with multiple compositions in layers to enhance the model accuracy. We show the comparative results in Table 6. In this table, we compare the accuracy of CFBPNN model and AE-based CFBPNN (AE+CFBPNN). Both the models are introduced by us; we observe that AE integrated with CFBPNN performs better than the traditional cascade model. As a result, we select AE and CFBPNN for PIGNUS. The findings of PIGNUS on overall detection are displayed in the discussion, followed by the outcomes of attack-specific conditions. We conclude by comparing PIGNUS with other state-of-the-art models.

Table 6: Overall Accuracy of the CFBPNN and PIGNUS model for all five datasets

Dataset	CFBPNN Accuracy (%)	PIGNUS Accuracy (%)	AE-MSE
NSLKDD+	99.02	99.02	8.46
UNSW-NB15	70.06	100.00	7.93
Gas Pipeline	98.02	100.00	4.23
Water storage tank	93.09	99.09	2.87
X-IIoTID	73.05	91.07	4.24

Table 6 display the performance of all five datasets of PIGNUS. The NSLKDD+ dataset remains constant in both models. where a huge improvement is noticed in UNSW-NB15 and XIIoT-ID datasets.

4.3.1. Overall performance

As the Industrial IoT dataset have huge data elements, compressing and decoding data elements trace the pattern more effectively than the traditional cascading method. We observe a 30.00% improvement of detection accuracy in UNSW-NB15 and X-IIoTID dataset while using AE-based CFBPNN. We observe 6.00% improvement for the water storage tank dataset in attack detection. Repeated training with change in parameters for PIGNUS has given 100.00% accuracy upon identifying the binary and multi-class attacks for gas pipeline and UNSW-NB15 dataset 6.

PIGNUS results in 0.00% false rate for UNSW-NB15 and gas pipeline dataset and 0.01% and 0.08% for water storage tank and NSLKDD+ dataset respectively. The false rate observations reflect the nonexistence of Type-I or Type-II errors. The false rate of this experiment prove that PIGNUS is suitable for detecting attacks with binary and multi-class classification. The model shows excellent performance for all datasets with 100.00% accuracy and 0.00% false rate for binary classification, projected in Figure 8.

Traditional feed-forward networks travel in one direction, passing the input data with added weights and bias to the next layer. RNN travels using a loop structure connecting the neurons of previous and successive layers. The combination of both the network with interconnection from the input to output is the quality of PIGNUS. This avoids the missing values and the interconnection to carry best weights and bias. For hidden layers sigmoid function is used and for output layers purelin function is used. We have tested the model with various training algorithms wherein the Levenberg-Marquardt results as the best fit for all five datasets with higher accuracy.

Comparative analysis of the AE+CFBPNN model gives a quite low value of MSE: nearby 0.264 for a gas pipeline, 0.217 MSE for a water storage tank, and 1.74 MSE for NSLKDD+, 1.52 MSE for UNSW-NB15 dataset and 0.76 for X-IIoTID dataset. We notice the least gradient value at 0.0039 for the water storage tank dataset with six validation checks indicating the best performance state of the model. Time taken for training is considerably low with $00:29$ minutes for the gas pipeline and the highest time is $3 hours 45 minutes$ for the X-IIoT dataset. We use regression value to identify the relation between input and the target variable; regression test shows 0.97071 R-value for NSLKDD+ dataset 0.98307 for a water storage tank and 1 for both gas pipeline and UNSW-NB15 datasets signify the absolute relations among the variables.

4.3.2. Attack wise detection results

We evaluate PIGNUS for all five datasets, and the performance is estimated using accuracy, precision, recall, and F1 score . For all attack classes the UNSW-NB15 and the gas pipeline dataset project absolute accuracy with 0.0% false rate, hence other three dataset attack performance is visualized in given in Table 8, Table 7 and Table 9. NSLKDD+ dataset have variations to identify some attack classes given in Table 8. NSLKDD+ dataset contains 16

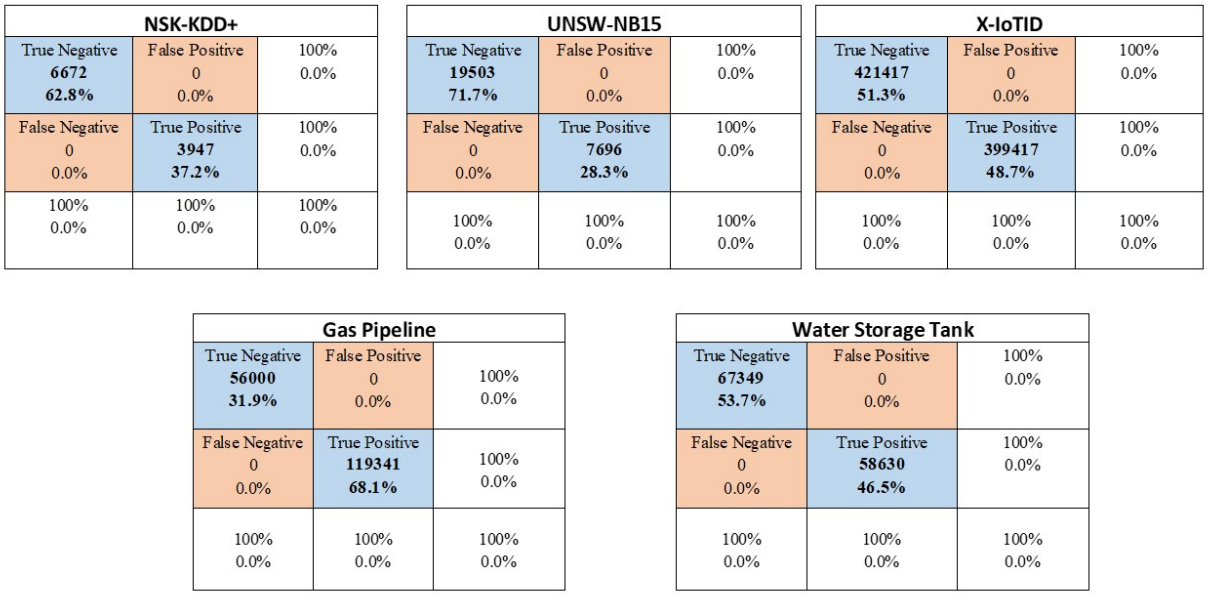


Figure 8: Binary class confusion matrices for all five datasets

attacks, which are subdivided into four classes. PIGNUS shows accuracy of 100.00% for DoS attack, probe attack, and also for normal traffic detection. R2L and U2R attacks are detected by PIGNUS with 99.86% accuracy.

Table 7: PIGNUS multi-class detection for NSLKDD+ dataset

Attack	Accuracy	Precision	Recall	F1 Score
Normal	100.00	1.0	1.0	1.00
DoS	100.00	1.0	1.0	1.00
Probe	100.00	1.0	1.0	1.00
R2L	99.86	0.5	1.0	0.95
U2R	99.86	0.5	1.0	0.95
RDOS	100.00	1.0	1.0	1.00

Water storage tank dataset provides six attack classes, in which PIGNUS found effective to identify DOS and reconnaissance attack instances compare to other four malicious packets. However, PIGNUS shows less accuracy for detecting malicious state command injection attacks. In contrast, malicious parameter command injection and function code injection only vary by 0.1% when compared to denial-of-service attacks. The water storage tank dataset shows that the naive malicious response injection attacks perform the least among with a false rate of about 0.3 percent comparatively. PIGNUS efficiency for Water storage tank is given in Table 8.

We experiment PIGNUS with industrial intrusion detection data set: X-IIoTID. PIGNUS produce significantly better results than the conventional cascade model. In the three kinds of attack occurrences that this dataset supports, PIGNUS gives 100.00% accuracy for binary class. In attack wise comparison C and C attack has given the least performance with 8.36 false rate. Apart from tampering and RDOS and lateral moveman ent all the attack detection with PIGNUS results above 99 $produce suracy. We provide a detailed summary of the PIGNUS performance against attacks using X-IIoT ID dataset in Table$ 9.

Table 8: PIGNUS multi-class detection for Water storage tank

Attack	Accuracy	Precision	Recall	F1 Score
Naive Malicious Response Injection	99.97	1.0	1.0	1.0
Complex Malicious Response Injection	99.98	1.0	0.99	1.0
Malicious State Command Injection	99.97	1.0	0.99	1.0
Malicious Parameter Command Injection	99.97	0.97	1.0	0.98
Denial Of Service	100.00	1.0	1.0	1.0
Reconnaissance	100.00	1.0	1.0	1.0

Table 9: PIGNUS multi-class detection results for X-IIoTID dataset

Attack	Accuracy (%)	Precision	Recall	F1 Score
C and C	91.64	1.0	0.68	0.81
Crypto-ransomware	99.99	1.0	0.99	1.0
Exfiltration	99.79	0.98	1.0	0.99
Exploitation	99.74	0.98	0.98	0.98
Lateral-movement	98.37	0.89	1.0	0.94
RDOS	98.95	0.93	1.0	0.96
Reconnaissance	99.55	0.97	1.0	0.98
Tampering	94.02	0.18	0.96	0.31
Weaponization	99.02	0.99	0.94	0.97

4.3.3. Comparison of PIGNUS with state-of-the-art models

We compare our proposed PIGNUS with the state-of-the-art models and display the results in Table 10. Feed forward methods have the disadvantage of non-recurrent values or missing values. We solve this problem by cascading recursive method proposed in PIGNUS. Our proposed model works with recursive connectivity to calculate weight and bias. PIGNUS also forwards the values to the next layer for fine-tuning the detection technique.

From Table 10 show that all of the models' accuracy increases as recall values rise and the other models performances are fairly comparable. With loop connection PIGNUS can track both the forward and backward process and it produces 0% false rates for detection which is comparable to the FNN model [27]. Anomaly detection using DAE and DFNN by [25] achieved 92.04 % accuracy, while PIGNUS achieved 99.02 % accuracy for NSK-KDD dataset. The nested structure of the cascading model improve performance significantly. PIGNUS is more accurate than DRaNN [34] compared with the false rate and performance for the KDDCUP+ dataset. PIGNUS performs much better than [29]. CNN methods are more usable for image-based datasets a multi-conventional network model proposed by [29] results in a 13.5 false ratio comparatively the highest false rate out of all available models. RSRT model [31], DNN and DT model [34] show 96.00% accuracy for IIoT datasets. PIGNUS outperforms all these mentioned models with 100% accuracy. DNN model [35] is tested on KDDCup99, NSLKDD, and UNSW-NB15 datasets and projects high performance compared to other datasets. Our PIGNUS results in 100% accuracy for the same dataset with five recursive and cascading layers. The false-positive ratio of our model is less than all the models. High accuracy with notable precision value is observed in [34]. However, the disadvantage with regular deep learning techniques is to determine values with the next layer; PIGNUS have the advantage of processing the previous weight and bias values to the next hidden layers.

We compare the FPR metrics for all the existing models given in Table 10. We observe different performances between Multi-CNN and the proposed PIGNUS model. The concurrent DM model is close to the results of PIGNUS, at the same time [36] has a minimum false rate indicating more detection efficiency. Comparing PIGNUS with the latest model, [39] produce 99.2% accuracy tested on water storage tank results.

The model works effectively on testing with the traditional dataset but real-time industrial dataset comparison is not much clear. Real-time testing model bu [40] for industrial gas pipeline dataset results with 99.9% accuracy; following the similar line, PIGNUS results in 100% accuracy for the industrial dataset. The experiment results in 0% FPR for

Table 10: Comparison of DL models for IDS in Industrial IoTs

Source	DL Technique	Dataset	Accuracy (%)	FPR	Precision	Recall
Muna et al. [25]	DAE and DFNN	NSL-KDD	95.05	5.00	0.94	0.84
Mengmeng .Ge et al. 2019 [27]	FNN	BoT-IoT	99.00	1.00	0.99	0.99
Mohammad M.H et al. 2019 [31]	RSRT	SCADA	96.71	3.29	0.97	0.96
Yanmiao Li et al. 2020 [29]	Multi-CNN	KDD test +21	86.95	13.05	0.89	0.87
Abassi A.AI et al. 2020 [32]	DNN, DT	GP, SWaT	96.00	4.00	0.94	0.93
Shahid Latif et al. 2020 [34]	DRaNN	UNSW-NB15	99.41	0.59	0.99	0.98
Sarika Choudhary et al. 2020 [35]	DNN-IDS	UNSW-NB15	91.50	8.50	0.93	0.92
Zhihong Tian et al. 2020 [36]	Multiple concurrent DL	CSIC 2010	99.04	0.60	0.99	0.95
Beibei Li et al. 2021 [37]	CNN-GRU	Industrial CPS	99.20	0.80	0.95	0.94
Awotunde et al. 2021 [38]	DFNN	UNSW-NB15	98.09	1.10	0.967	0.99
Othmane Friha et al. 2022 [39]	DNN, CNN, RNN	CSE-CICIDS2018	99.02	0.8	0.92	0.96
S. Tharewal et al. 2022 [40]	DRL-IDS	Gas Pipeline	99.01	0.01	0.99	0.99
Proposed PIGNUS	AE+CFBPNN	UNSW-NB15	100.00	1.00	1.00	1.00

NSLKDD+, water storage tank, and 0.22% for UNSW-NB15, and 2.89 for the gas pipeline dataset. The strong reason for this efficiency is the length of vectors passed on to the model for processing. We notice that AE implementation for the dataset has high dimensions and retains a low false rate compared to the high featured dataset. This proves that our PIGNUS model is suitable for providing security in IIoT networks.

5. Conclusion

In this paper, we propose PIGNUS a hybrid model with the combination of AE and CFBPNN. PIGNUS identifies multi-class IIoT attacks and contributes towards a secured environment with increased attack detection effectiveness. To demonstrate the model's performance we test PIGNUS on five well-known datasets: UNSW-NB15, NSLKDD+, X-IIoTID, gas pipeline, and water storage tank. On the UNSW-NB15 and gas pipeline dataset, PIGNUS performs best with a 0.0% false rate in identifying multi-class attacks. Given that the model was developed using IIoT-specific datasets, the results demonstrate how well-suited PIGNUS for the IIoT environment. In the future, we would like to extend our experiment with other open datasets and enhance multi-class attack identification to generalize the findings.

References

[1] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, R. Atkinson, Shallow and deep networks intrusion detection system: A taxonomy and survey, arXiv preprint arXiv:1701.02145.

[2] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, et al., Experimental security analysis of a modern automobile, in: 2010 IEEE Symposium on Security and Privacy, IEEE, 2010, pp. 447–462.
[3] J. P. Farwell, R. Rohozinski, Stuxnet and the future of cyber war, Survival 53 (1) (2011) 23–40.
[4] N. Falliere, L. O. Murchu, E. Chien, W32. stuxnet dossier version 1.4, Symantec Security Response.
[5] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, S. Sastry, Attacks against process control systems: risk assessment, detection, and response, in: Proceedings of the 6th ACM symposium on information, computer and communications security, 2011, pp. 355–366.
[6] F. G. Mármol, C. Sorge, O. Ugus, G. M. Pérez, Do not snoop my habits: preserving privacy in the smart grid, IEEE Communications Magazine 50 (5) (2012) 166–172.
[7] D. Edwards, infographic-a-history-of-cyber-attacks-on-the-industrial-internet-of-things, roboticsandautomationnews.com 7264.
[8] S. R. Chhetri, S. Faezi, N. Rashid, M. A. Al Faruque, Manufacturing supply chain and product lifecycle security in the era of industry 4.0, Journal of Hardware and Systems Security 2 (1) (2018) 51–68.
[9] F. Tao, Q. Qi, A. Liu, A. Kusiak, Data-driven smart manufacturing, Journal of Manufacturing Systems 48 (2018) 157–169.
[10] A. Hijazi, A. El Safadi, J.-M. Flaus, A deep learning approach for intrusion detection system in industry network., in: BDCSIntell, 2018, pp. 55–62.
[11] M. Conti, D. Donadel, F. Turrin, A survey on industrial control system testbeds and datasets for security research, arXiv preprint arXiv:2102.05631.
[12] C. A. Boye, P. Kearney, M. Josephs, Cyber-risks in the industrial internet of things (iiot): towards a method for continuous assessment, in: International Conference on Information Security, Springer, 2018, pp. 502–519.
[13] R. Balaji, S. Deepajothi, G. Prabaharan, T. Daniya, P. Karthikeyan, S. Velliangiri, Survey on intrusions detection system using deep learning in iot environment, in: 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), IEEE, 2022, pp. 195–199.
[14] K. Tsiknas, D. Taketzis, K. Demertzis, C. Skianis, Cyber threats to industrial iot: A survey on attacks and countermeasures, IoT 2 (1) (2021) 163–188.
[15] G. Thamilarasu, S. Chawla, Towards deep-learning-driven intrusion detection for the internet of things, Sensors 19 (9) (2019) 1977.
[16] S. Manimurugan, S. Al-Mutairi, M. M. Aborokbah, N. Chilamkurti, S. Ganesan, R. Patan, Effective attack detection in internet of medical things smart environment using a deep belief neural network, IEEE Access 8 (2020) 77396–77404.
[17] Y. Otoum, D. Liu, A. Nayak, Dl-ids: a deep learning–based intrusion detection framework for securing iot, Transactions on Emerging Telecommunications Technologies 33 (3) (2022) e3803.
[18] A. Tabassum, A. Erbad, A. Mohamed, M. Guizani, Privacy-preserving distributed ids using incremental learning for iot health systems, IEEE Access 9 (2021) 14271–14283.
[19] M. Nasir, A. R. Javed, M. A. Tariq, M. Asim, T. Baker, Feature engineering and deep learning-based intrusion detection framework for securing edge iot, The Journal of Supercomputing 78 (6) (2022) 8852–8866.
[20] S. Ullah, M. A. Khan, J. Ahmad, S. S. Jamal, Z. e Huma, M. T. Hassan, N. Pitropakis, W. J. Buchanan, Hdl-ids: a hybrid deep learning architecture for intrusion detection in the internet of vehicles, Sensors 22 (4) (2022) 1340.
[21] T. Saba, A. Rehman, T. Sadad, H. Kolivand, S. A. Bahaj, Anomaly-based intrusion detection system for iot networks through deep learning model, Computers and Electrical Engineering 99 (2022) 107810.
[22] A. Dahou, M. Abd Elaziz, S. A. Chelloug, M. A. Awadallah, M. A. Al-Betar, M. A. Al-qaness, A. Forestiero, Intrusion detection system for iot based on deep learning and modified reptile search algorithm, Computational Intelligence and Neuroscience 2022.
[23] M. Zhong, Y. Zhou, G. Chen, Sequential model based intrusion detection system for iot servers using deep learning methods, Sensors 21 (4) (2021) 1113.
[24] W. Zhang, Y. Zhang, Intrusion detection model for industrial internet of things based on improved autoencoder, Computational Intelligence and Neuroscience 2022.
[25] A.-H. Muna, N. Moustafa, E. Sitnikova, Identification of malicious activities in industrial internet of things based on deep learning models, Journal of information security and applications 41 (2018) 1–11.
[26] O. Faker, E. Dogdu, Intrusion detection using big data and deep learning techniques, in: Proceedings of the 2019 ACM Southeast Conference, 2019, pp. 86–93.
[27] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, A. Robles-Kelly, Deep learning-based intrusion detection for iot networks, in: 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), 2019, pp. 256–25609.
[28] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, S. Venkatraman, Deep learning approach for intelligent intrusion detection system, IEEE Access 7 (2019) 41525–41550.
[29] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, L. Cui, Robust detection for network intrusion of industrial iot based on multi-cnn fusion, Measurement 154 (2020) 107450.
[30] E. Gyamfi, A. D. Jurcut, Novel online network intrusion detection system for industrial iot based on oi-svdd and as-elm, IEEE Internet of Things Journal.
[31] M. M. Hassan, A. Gumaei, S. Huda, A. Almogren, Increasing the trustworthiness in the industrial iot networks through a reliable cyberattack detection model, IEEE Transactions on Industrial Informatics 16 (9) (2020) 6154–6162.
[32] A. Al-Abassi, H. Karimipour, A. Dehghantanha, R. M. Parizi, An ensemble deep learning-based cyber-attack detection in industrial control system, IEEE Access 8 (2020) 83965–83973.
[33] R. V. Mendonça, J. C. Silva, R. L. Rosa, M. Saadi, D. Z. Rodriguez, A. Farouk, A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithm, Expert Systems (2021) e12917.
[34] S. Latif, Z. Idrees, Z. Zou, J. Ahmad, Drann: A deep random neural network model for intrusion detection in industrial iot, in: 2020 International Conference on UK-China Emerging Technologies (UCET), 2020, pp. 1–4.
[35] S. Choudhary, N. Kesswani, Analysis of kdd-cup'99, nsl-kdd and unsw-nb15 datasets using deep learning in iot, Procedia Computer Science 167 (2020) 1561–1573.
[36] Z. Tian, C. Luo, J. Qiu, X. Du, M. Guizani, A distributed deep learning system for web attack detection on edge devices, IEEE Transactions

on Industrial Informatics 16 (3) (2020) 1963–1971.
[37] B. Li, Y. Wu, J. Song, R. Lu, T. Li, L. Zhao, Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems, IEEE Transactions on Industrial Informatics 17 (8) (2021) 5615–5624.
[38] J. B. Awotunde, C. Chakraborty, A. E. Adeniyi, Intrusion detection in industrial internet of things network-based on deep learning model with rule-based feature selection, Wireless communications and mobile computing 2021.
[39] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, K.-K. R. Choo, M. Nafaa, Felids: Federated learning-based intrusion detection system for agricultural internet of things, Journal of Parallel and Distributed Computing 165 (2022) 17–31.
[40] S. Tharewal, M. W. Ashfaque, S. S. Banu, P. Uma, S. M. Hassen, M. Shabaz, Intrusion detection system for industrial internet of things based on deep reinforcement learning, Wireless Communications and Mobile Computing 2022.
[41] H. Liu, B. Lang, Machine learning and deep learning methods for intrusion detection systems: A survey, applied sciences 9 (20) (2019) 4396.
[42] N. Moustafa, J. Slay, Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 military communications and information systems conference (MilCIS), IEEE, 2015, pp. 1–6.
[43] T. Morris, W. Gao, Industrial control system traffic data sets for intrusion detection research, in: International Conference on Critical Infrastructure Protection, Springer, 2014, pp. 65–78.
[44] T. H. Morris, Z. Thornton, I. Turnipseed, Industrial control system simulation and data logging for intrusion detection system research, 7th annual southeastern cyber security summit (2015) 3–4.
[45] KDD dataset, `http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html`, [Online; accessed 10-June-2022] (1999).
[46] M. Al-Hawawreh, E. Sitnikova, N. Aboutorab, X-iiotid: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things, IEEE Internet of Things Journal 9 (5) (2021) 3962–3977.
[47] B. Yan, G. Han, Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system, IEEE Access 6 (2018) 41238–41248.
[48] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, F. Ahmad, Network intrusion detection system: A systematic study of machine learning and deep learning approaches, Transactions on Emerging Telecommunications Technologies 32 (1) (2021) e4150.
[49] M. M. Baig, M. M. Awais, E.-S. M. El-Alfy, A multiclass cascade of artificial neural network for network intrusion detection, Journal of Intelligent & Fuzzy Systems 32 (4) (2017) 2875–2883.
[50] J. Qiao, F. Li, H. Han, W. Li, Constructive algorithm for fully connected cascade feedforward neural networks, Neurocomputing 182 (2016) 154–164.
[51] B. Warsito, R. Santoso, H. Yasin, et al., Cascade forward neural network for time series prediction, in: Journal of Physics: Conference Series, Vol. 1025, IOP Publishing, 2018, p. 012097.