Deep Learning Based Attack Detection in IIoT using Two-Level Intrusion Detection System

Kathiroli Raja (kathirolig@gmail.com)
Anna University Chennai
Krithika Karthikeyan
Anna University Chennai
Abilash B
Anna University Chennai
Kapal Dev
University of Johannesburg
Gunasekaran Raja
Anna University Chennai

Research Article

Keywords: Internet of Things, Intrusion Detection System, Deep Learning, Dragonfly Algorithm, Dempster Shafer's theory

Posted Date: November 3rd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-997888/v1

Deep Learning Based attack detection in IIoT using Two-Level Intrusion Detection System

Kathiroli Raja[1*], Krithika Karthikeyan[2†], Abilash B[3†], Kapal Dev[4†] and Gunasekaran Raja[5†]

[1]Department of Computer Technology, Madras Institute of Technology, Anna University, Chennai, 600044, Tamil Nadu, India.
[2]Department of Computer Technology, Madras Institute of Technology, Anna University, Chennai, 600044, Tamil Nadu, India.
[3]Department of Computer Technology, Madras Institute of Technology, Anna University, Chennai, 600044, Tamil Nadu, India.
[4]Department of Institute of Intelligent Systems, University of Johannesburg, Johannesburg, 600044,South Africa.
[5]Department of Computer Technology, Madras Institute of Technology, Anna University, Chennai, 600044, Tamil Nadu, India.

*Corresponding author(s). E-mail(s): kathirolig@gmail.com;
Contributing authors: krithikamit17@gmail.com;
abilashbalaraman6@gmail.com; kapald@uj.ac.za;
dr.r.gunasekaran@ieee.org;
†These authors contributed equally to this work.

Abstract

The Industrial Internet of Things (IIoT), also known as Industry 4.0, has brought a revolution in the production and manufacturing sectors as it assists in the automation of production management and reduces the manual effort needed in auditing and managing the pieces of machinery. IoT-enabled industries, in general, use sensors, smart meters, and actuators. Most of the time, the data held by these devices is surpassingly sensitive and private. This information might be modified, stolen, or even the devices may be subjected to a Denial of Service (DoS) attack. As a consequence, the product quality may deteriorate or sensitive information may be leaked. An Intrusion Detection System (IDS), implemented in the network layer of IIoT, can detect attacks, thereby protecting the data and devices. Despite substantial advancements in attack detection in IIoT, existing works fail to detect certain attacks obfuscated from detectors resulting in a low detection performance. To address the aforementioned issue, we propose a Deep Learning-based Two Level Network Intrusion Detection System (DL-TL-NIDS) for IIoT environment, emphasizing challenging attacks. The attacks that attain low accuracy or low precision in level-1 detection are marked as challenging attacks. Experimental results show that the proposed model, when tested against TON_IoT, figures out the challenging attacks well and achieves an accuracy of 99.97%, precision of 95.62%, recall of 99.5%, and F1-score of 99.65%. The proposed DL-TL-NIDS, when compared with state-of-art models, achieves a decrease in false alarm rate to 2.34% (flagging normal traffic as an attack) in IIoT.

Keywords: Internet of Things, Intrusion Detection System, Deep Learning, Dragonfly Algorithm, Dempster Shafer's theory

1 Introduction

IDS can be seen as either hardware or software which detects the anomalous activity and report it to the administrator. It helps in alleviating the impact of the attack by informing the system administrator prior. James Anderson presented the definition of IDS in 1980[1]. He examined access logs and server case logs using a collection of methods. Dorothy E. Denning created an anomaly-based IDS based on statistics in 1986 and called it Intrusion Detection Expert System (IDES). Teresa F. Lunt applied Artificial Neural Networks (ANN) to IDES to boost it. Wisdom & Sense created a rule-based anomaly detector using mathematical analysis. Researchers at the University of California, Davis developed the Distributed Intrusion Detection System in 1991 (DIDS).

In general, based on the type of detection, there are two types of IDS, Signature-based detection and Anomaly-based detection [2]. Signature-based detection matches the incoming network traffic with the existing patterns extracted. This is also called Misuse based detection or Knowledge-based detection. An Anomaly-based Intrusion Detection System (AIDS) is used to detect unknown attacks. It works on the principle that any abnormal network traffic could be malicious. As a result, such network activity can be identified as an exception and subjected to further investigation to determine the nature of the traffic. Machine Learning (ML) algorithms are proven to work effectively and are reliable for anomaly detection [3].

Many research papers have presented machine learning based IDS. Both supervised and unsupervised are used to detect intrusions. Ensemble techniques have been used to detect attacks [4]. Semi-supervised models have also been found beneficial to balance the dataset to detect anomalies [5]. Attacks can also be detected using unsupervised approaches such as clustering and Self Organized Maps (SOM) [6, 7]. Later, deep learning models have been used to train intrusion detection models. This is because machine learning techniques normally require a huge processing time. Deep learning has been employed as it has the capability of training large datasets. It can be combined with big data techniques to train a huge dataset and proven to detect better than other methods [8]. It has been proven that it can reduce the false positive rate [9].

The challenges faced in building the IDS are twofold: to begin with, an imbalanced dataset will lead to a low detection rate of the minority attacks; secondly, even though the IDS model can detect the vast majority of assaults, some attacks go undetected due to misclassification. Such attacks necessitate sophisticated identification. Hence, the proposed work aims to develop DL-TL-NIDS that detects malicious activity. The following workflow is used; initially, the dataset is balanced and standardized. In the first-level detection, Deep Neural Networks (DNN) are trained and tested. The attacks that get a lower detection rate or lower precision are declared as challenging attacks. The challenging attacks are fed to second-level detection. Second-level detection employs two models namely the Negative Selection Algorithm (NSA) and Deep Neural Networks (DNN) trained using Dragonfly Algorithm. Dempster Shafer's combination rule is used to combine the outputs of both models. We have evaluated our work against CICIDS 2017, CICIDS 2018 [10] and TON_IoT datasets[11–18].

2 Motivation

The major concern for IIoT is security and privacy [19]. The real-world examples demonstrate that security in IIoT environments has become a necessity since failure to do so has severe consequences. Many researchers have advocated the use of Intrusion Detection Systems (IDS) to identify and mitigate attacks. However, IDS in the IIoT context poses several problems. The difficulties are listed below.

1. IIoT consists of heterogeneous components interconnected via networks. Due to the distributed nature of IIoT devices, hackers can simply get access to them via networks [19–21].
2. IIoT devices generate massive data on benign traffic and less data on intrusions that occur sporadically [22–24]. That is, the dataset's distribution is extremely skewed, which has an impact on the model's detection performance.
3. Even though IDS can be constructed with Machine Learning methods, it might lead to overfitting when a large dataset is used for training [25].

We propose a DL-TL-NIDS to combat the above challenges. The main contributions of our work are as follows

1. The dispersion of the training dataset has a dormant impact on the behavior of the model. Hence oversampling techniques have been employed to balance the dataset.
2. Deep Learning was employed as it can process a huge amount of data and can avoid overfitting.
3. We concentrate on challenging attacks. Attacks that are difficult to detect by a model are termed challenging attacks. Accuracy and precision were the measures used to evaluate the model's performance in detecting attacks.
4. The proposed DL-TL-NIDS is capable of identifying challenging attacks. In the first level, the Deep Neural Networks separate challenging attacks from other attacks (termed as easy-to-detect attacks). Second-level detection will be applied only to challenging attacks. Easy-to-detect attacks will not be subjected to second-level detection to avoid additional computations.
5. In second-level detection, two detectors namely the NSA and DNN (trained using enhanced Dragonfly Algorithm) were employed. The output obtained from each detector is the probability of a test data being an attack. Dempster Shafer's combination rule is used to fuse the decision obtained from the detectors.

3 Literature Survey

In the IIoT, attacks majorly occur in (1) Operational Technology (OT) and (2) Information Technology (IT). IP spoofing, eavesdropping, bruteforce password guessing, and data manipulation may occur at the OT level for IoT devices and controllers such as programmable logic controllers, gateways, and operator stations. Phishing, SQL injection, brute force attacks, and DoS attacks can affect IT level components such as data centers, online and mail services, edge devices, and mobile devices. The following are some of the consequences due to the aforementioned attacks in IIoT.

1. Business secret data can be stolen by competitors, resulting in the manufacture of duplicate items.
2. Data collected from sensors and smart meters can be tampered with, resulting in a loss of product quality.
3. End-user devices and IoT devices could be compromised, resulting in service unavailability.

Hence it is essential to detect and mitigate the attacks in the IIoT context. Many researchers have proposed various types of IDS or the IIoT environment that are capable of detecting attacks. Aboelwafa et.al [26] detected falsified data injection attacks using Autoencoders (AE) and cleaned them using Denoising Autoencoders (DAE). The evaluation was done via simulation with an accuracy of 97.02%. Maede et al [27] addressed the use of machine learning approaches to identify attacks. The machine learning models were tested on real-world test datasets and were successful in detecting backdoor, command injection, and SQL injection. However, the study lacks complex hybrid models, and false negatives are significant. To discover abnormal behaviors, Li et al [28] employed time series analysis. The researchers used multilayer Long Short-Term Memory (LSTM) and an enhanced bidirectional LSTM. The suggested work was evaluated against the CTU-13 and AWID datasets, with 95.01% and 97.58% accuracy, respectively. Mahbub et al [29] concentrated their research on selecting adversarial samples that can deceive a classifier. The models were retrained after eliminating adversarial samples. The samples were chosen based on the malware's cluster center's closeness and likelihood calculated using Kernel-Based Learning (KBL). The work was evaluated against a publicly available dataset with an accuracy of 86.08%.

Yan et al. [30] created a deep learning model, trained using mini-batch gradient descent with adjustable learning rate and momentum. The work was evaluated against web domains obtained from Alex top1w and 360 netlab and achieved a precision of approximately 90%. Bhunia et.al [31] proposed SoftThings, a Software Defined Networks (SDN) based IIoT security framework. The machine learning algorithms were employed at the SDN controller to monitor the network. The precision of SoftThings was tested using the mininet emulator, and it was found to be 98%. Deep-IFS, implemented in a Fog environment, was proposed by Abdel-Basset et al [19]. The master fog node distributes the training parameters to the workers and then combines their decisions. The suggested work was tested against the Bot-IIoT dataset, yielding a 98.1% accuracy. Latif et.al [32] proposed a Deep Random Neural Network (DRaNN) and evaluated using UNSW-NB15. The experimental results show that the model was able to detect attacks with an accuracy of 99.41%.

Yao et.al proposes a Multi-level intrusion detection model framework named MSML to overcome the imbalance of network traffic and non-identical distribution between the training set and test set in feature space [33]. This framework includes pure cluster extraction, pattern discovery fine-grained classification, and model updating. KDD'99 dataset was used for evaluation. The framework can effectively distinguish known unknown pattern samples with accuracy (99.3%). Liang et.al [34] resolves the inherent problems in IDS such as low detection rate, low real-time performance, and high false-positive rate by proposing a multi-feature data clustering optimization model. The clusters are formed based on the distance between the cluster center and the data point. For the NSL-KDD dataset, the average time saving is 7.8% than the existing models. An overall detection rate (accuracy) of 97.8% is achieved. Yan et.al [35] proposed a multi-level DDoS mitigation framework to identify DDoS attacks in the IIoT. The proposed approach was evaluated against real-time DDoS attacks generated using the ping of death and TCP SYN flood.

In [36], the Apache framework is used and proposes a hybrid algorithm to exploit deep learning and machine learning advantages. The latent features are extracted using stacked autoencoders. The accuracy achieved in ISCX 2012 dataset is 90%. The overall accuracy of 99% is achieved in CICIDS 2017 dataset. Maharani et.al [37] presented the IIoT attack detection in the fog layer of the IIoT environment. The proposed model employed ML models like Decision Trees, K-means and Random Forest and was evaluated against the KDD Cup'99 dataset. The study concluded that K-means outperformed other algorithms, reaching a 93% accuracy rate. A non-symmetric deep autoencoder (NDAE) was proposed in [9] for unsupervised feature learning. The benchmark KDD Cup '99 and NSL-KDD datasets were used for random forest evaluation, obtaining 85.43% and 97.85% accuracy, respectively. Though the proposed model has better overall performance, it could not detect the User to Root (U2R) and Remote to Local (R2L) attacks. Zhong et.al [38] tried to adopt big data for the machine learning model to train on a massive amount of data. The behavioral and content features were extracted and used a combination of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) for the prediction. The proposed method was evaluated against the ISCX2012, CICIDS2017, and DARPA1998 datasets.

It is inferred from the existing works that many works had presented the overall performance of the model. However, they lacked analysis of attack-wise performance, and single-level detection was performed. Single level detection will be an issue when certain attacks, that are extremely harmful (eg., ransomware, DoS, infiltration, etc.,), cannot be identified by the model. Hence in our proposed work, we focus on attack-wise performance and two-level detection. We propose a DL-TL-NIDS, a two-level IDS framework. At the first level, the challenging attacks were identified and segregated using DNN. In second-level, the challenging attacks were detected and classified using hybrid models namely NSA and DNN(trained using enhanced dragonfly algorithm). The decision is fused using Dempster-Shafer's combination.

4 Proposed Methodology

This section explains the DL-TL-NIDS in detail. Figure 1 illustrates the workflow of DL-TL-NIDS. As mentioned earlier, challenging attacks are those attacks identified with low precision or low accuracy by the model. The key justification for focusing on challenging attacks is that, while the model identifies other types of attacks effectively, challenging attacks might cause the model's performance to deteriorate. Hence, the proposed work seeks to develop a two-level IDS that identifies attacks in the IIoT and focuses on challenging attacks. The sensors and smart meters in IIoT create monstrous traffic. The data will have a wide range and magnitude. So, the data preprocessing and normalization are done before the first-level detection.

4.1 Oversampling and Standardization

The attack distribution in the IIoT dataset is highly disproportionate as the benign traffic spawns frequently, and the intrusion traffic seldom occurs. Hence, balancing the dataset is necessary to overcome the variance in IIoT dataset distribution. Attack distribution of the dataset has a significant impact on the

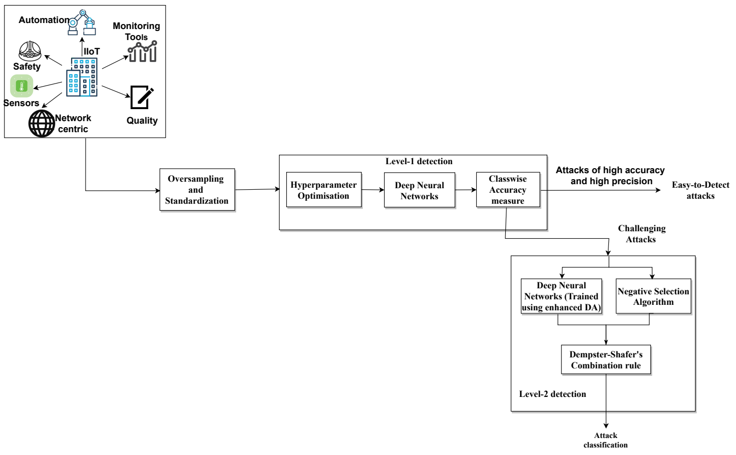

Automation
Monitoring Tools
IIoT
Safety
Sensors
Network centric
Quality
Oversampling and Standardization
Level-1 detection
Hyperparameter Optimisation
Deep Neural Networks
Classwise Accuracy measure
Attacks of high accuracy and high precision
Easy-to-Detect attacks
Challenging Attacks
Deep Neural Networks (Trained using enhanced DA)
Negative Selection Algorithm
Dempster-Shafer's Combination rule
Level-2 detection
Attack classification

Fig. 1: Flowchart of our Proposed work

model's performance. SMOTE, which stands for Synthetic Minority Oversampling Technique, is the oversampling technique employed. SMOTE takes each of the minority attacks and applies the nearest neighbor algorithm to find the data points in a class that are neighbors to each other. It aids to decide the amount of synthetic data needed to balance the dataset. For example: If the ratio between majority to minority class is 3:1, then we need to increase the amount of minority class by 200%. So the number of neighbors should be 2. After finding the nearest neighbors, the algorithm takes each set of neighbors and produces the new data point. The flowchart of SMOTE is given in Figure 2.

The data is standardized using a Z score technique. The technique uses the mean and standard deviation of a feature. The formula of Z score is given in equation (1):

$$x_{norm} = \frac{x - \mu}{\sigma} \tag{1}$$

Here x_{norm} is the normalized feature value, x is the original feature, μ is the mean value of the feature and σ is the standard deviation of the feature value.

4.2 Level-1 Detection

First level detection distinguishes between easy-to-detect and challenging attacks. A challenging attack may not be appropriately classified because it may be detected as benign or misclassified. The challenging attacks will be handled in Level-2 detection. The components of Level-1 detection are as follows

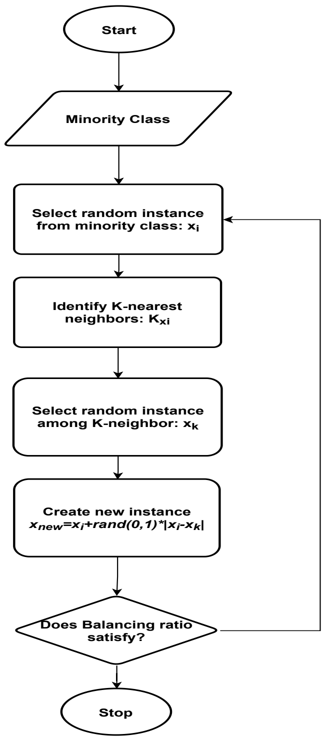
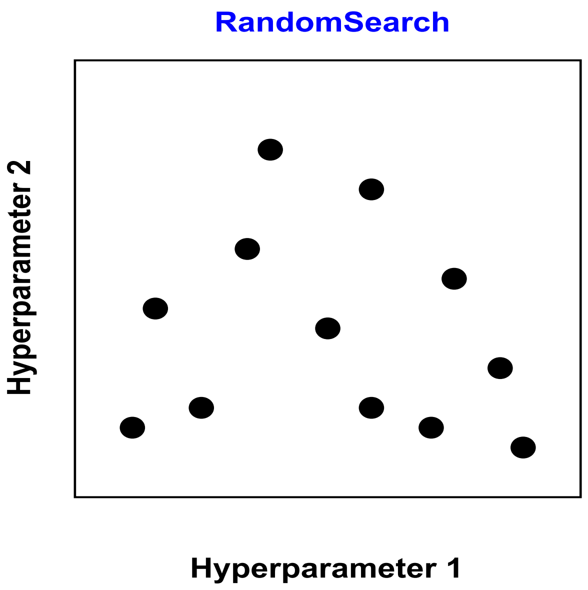

Start
Minority Class
Select random instance from minority class: x_i
Identify K-nearest neighbors: K_{xi}
Select random instance among K-neighbor: x_k
Create new instance $x_{new}=x_i+rand(0,1)*|x_i-x_k|$
Does Balancing ratio satisfy?
Stop

Fig. 2: Flowchart of SMOTE

4.2.1 Hyperparameter Optimization

Hyperparameter optimization is employed to find the optimal hyperparameters required for the model. Hyperparameters are the parameters set before learning and can exert on the model's performance (here it's DNN). Learning rate, momentum, epoch size, batch size, kernel initialization, and dropout regularisation are the hyperparameters considered here and optimized for a DNN. A fairly performing baseline model is opted to perform hyperparameter optimization. The hyperparameter optimization algorithm adopted is Random Search. Random search picks a random set of hyperparameter values at each instance. Since the values are random, there is a high likelihood that the entire search space of hyperparameters is exploited. In Figure 3, we have considered two hyperparameters, hyperparameter-1 and hyperparameter-2. The black dots represent the set of values taken by random search at each instance. The model is trained and tested using the set of hyperparameters chosen at each instance. The random search returns the hyperparameters for which the model works best after a fixed number of iterations.

Fig. 3: Random Search



4.2.2 Deep Neural Networks

The Deep Neural Networks (DNN) are similar to ANN, however, there are more hidden layers in DNNs. Each DNN has an input layer, multiple hidden layers, and an output layer. Each hidden layer has several neurons. Each neuron gets fired or remains neutral based on the inputs it receives.

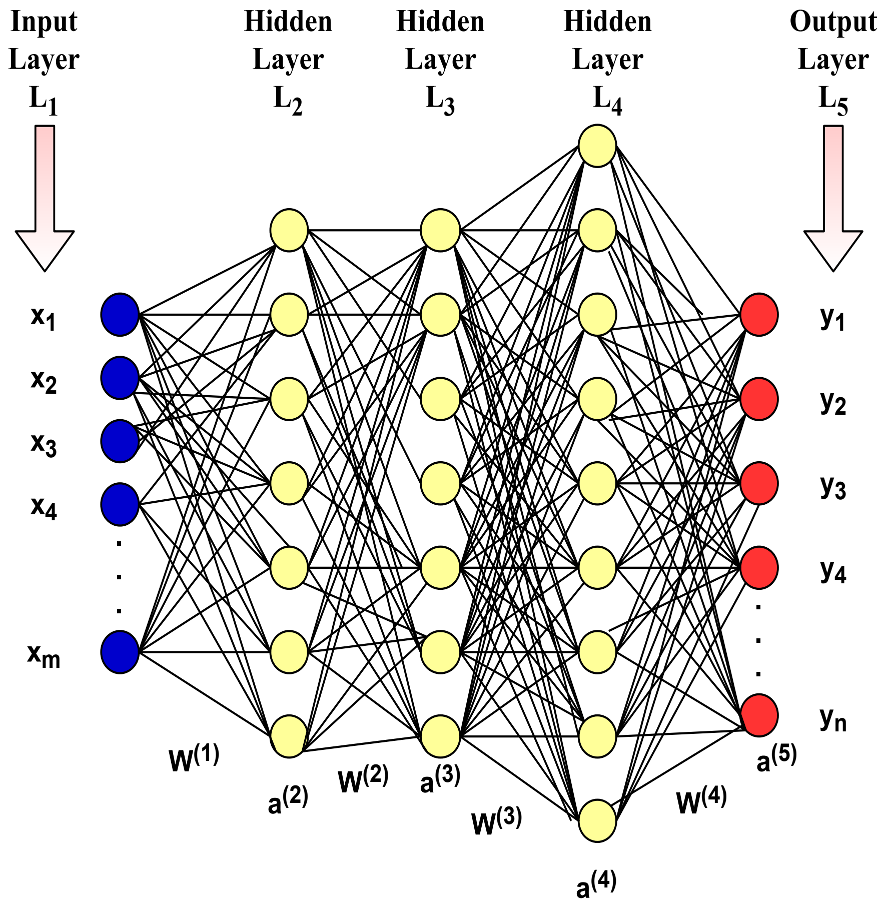
In figure 4, x_i is the input feature, $W^{(i)}$ are the weights of the connection between neurons from layer L_i to layer L_{i+1}, y_i is the output, and g is the activation function that fires the neuron based on the computation done during forwarding propagation. Each layer can use different activation functions. In our proposed work, we employed the relu function for hidden layers, as it worked better in the baseline model, and the softmax function as the activation function for the output layer, as the output we need is in terms of probability. The relu and softmax functions are given in (2) and (3).

$$softmax(g_i) = \frac{e^{g_i}}{\sum_{j=1}^{n} e^{g_j}} \tag{2}$$

$$relu(g_i) = max(0, g_i) \tag{3}$$

In the above equation, g_i is the output obtained from neuron i in the output layer and n is the number of classes (the number of neurons in the output

Fig. 4: Deep Neural Networks



layer). DNN consists of two phases: forward propagation and backpropagation. During the forward propagation, the inputs are multiplied by weights and bias assigned by each neuron and travel through each hidden layer, and then finally predicts the output y_i. Each neuron in hidden layer l calculates the following

$$g_l = W_l^T a^{l-1} + b^l \tag{4}$$

$$a_l = g(a_l) \tag{5}$$

In the above equation, g denotes the activation function. The DNN is trained using backpropagation. Backpropagation employs gradient descent. The motive of backpropagation is to update the weights such that the error between expected and predicted output is minimal. Gradient computation involves computing changes in weight with respect to the expected output (i.e dW and db). The error between actual and predicted output in the output layer is calculated and backpropagated to the preceding hidden layers. The weights and bias values are updated according to the value of gradients.

Gradient descent algorithms have many extension algorithms that are optimized. One of the optimizers is the Adam optimizer. This algorithm is efficient and it is a combination of gradient descent with momentum and the Root Mean Square (RMS) Prop algorithm. In the momentum method, the velocity with which the gradient is changing is calculated and RMSProp employs an exponentially weighted average method on the second moment of the gradients (dW_2). Adam optimiser uses decays both past squared gradient (V) and past momentum (S) calculated using (6) and (7). The adam adds bias correction to V and S using (8) and (9). Finally, the weights are updated using (10).

$$V = \beta_1 S + (1 - \beta_1) dW \tag{6}$$

$$S = \beta_2 S + (1 - \beta_2) dW^2 \tag{7}$$

$$V = \frac{V}{1 - \beta_1^i} \tag{8}$$

$$S = \frac{S}{1 - \beta_2^i} \tag{9}$$

$$W = W - \alpha \frac{V}{\sqrt{S} + \epsilon} \tag{10}$$

In our proposed work, DNN, as shown in Figure 4, is trained and tested using the preprocessed IIoT dataset. The hyperparameters obtained from the optimization algorithm were used for training the DNN.

4.2.3 Classwise Accuracy Measure

The module assesses the performance of the DNN by examining the accuracy and precision of each IIoT attack. An attack is considered easy-to-detect if it has both high accuracy and high precision, otherwise, it is labeled as a challenging attack.

4.3 Level-2 Detection

The goal of the second-level detection is to detect the challenging IIoT attacks that were misclassified at the first level detection. As hybrid models can enhance detection accuracy [39], Two models, DNN (trained using enhanced dragonfly algorithm) and NSA, are employed in the second level detection to identify challenging attacks.

4.3.1 Deep Neural Networks (Trained Using Enhanced Dragonfly Algorithm)

The same DNN architecture as in the first level is used. Softmax classifier is used as an activation function. DNN is trained using the Enhanced Dragonfly algorithm. Backpropagation is susceptible to noisy input and learning rate, and it has an issue with local optima. We use the dragonfly algorithm to determine ideal weights and biases for DNN. The dragonfly algorithm is a Swarm-based metaheuristics technique that is used to identify the best solutions to problems.

The algorithm is motivated by dragonflies' quest for food sources while fleeing from predators. A food source is the best solution, in terms of fitness, from an algorithmic standpoint, whereas an enemy source is the worst choice so far. The algorithm considers 5 elements namely separation (S), cohesion (C), alignment (A), food source (F) (i.e., best solution), and enemy (E) (i.e., worst solution). Also, each element is associated with the coefficients namely s, a, c, f, and e. The values of S, C, A, F, and E are computed as follows.

$$S_i = -\sum (X - X_j) \tag{11}$$

$$A_i = ((\sum V_j)/n) \tag{12}$$

$$C_i = ((\sum X_j/n) - X) \tag{13}$$

$$F_i = X^+ - X \tag{14}$$

$$E_i = X^- - X \tag{15}$$

$$\triangle X_t = (sS_i + aA_i + cC_i + fF_i + eE_i) + wX_t \tag{16}$$

$$X_{t+1} = X_t + levy(d) * \triangle X_t \tag{17}$$

The population and the number of iterations are both initialized, and each individual represents a vector of weights and bias values. At each iteration, the fitness of an individual is calculated using the fitness function. The fitness function for Neural Networks (NN) can be one of the performance metrics used to evaluate NN. In our work, we've used accuracy as the fitness function. Among all the individuals, the one with the highest fitness is taken as the best solution and it is assigned as the food source. The individual with the least fitness is taken as the worst solution and marked as the enemy source. The weights w, s, a, c, f, and e are assigned. For each individual, if there exists at least one neighbor the values of S, A, C, F, and E are calculated using the equations (11) – (15). The weights bias value is updated using equation (16) if there is a neighbor. If there is no neighbor to an individual, then the solution is updated using equation (17). The Levy flight is calculated using (18). Here b and r1 are constant values.

$$levy(x) = \frac{0.01X(r_1 * \sigma)}{|r_1|^{\frac{1}{b}}} \tag{18}$$

The drawback of this algorithm is as follows [40]

1. The lack of internal memory in this algorithm is a disadvantage since it might lead to premature convergence to the local optimum.

2. Levy flight mechanism is utilized to model the random flying behavior of dragonflies in nature. The disadvantages of Levy flight are overflowing of the search area and interruption of random flights due to its big searching steps.

The aforementioned flaws are addressed in our enhanced dragonfly algorithm. The changes are made on the use of the Levy flight and the introduction of the concepts of local best and global best solutions. Algorithm 1 describes the algorithm. For the neighbors in a cluster, the local best solution is used as a food source. The reason for selecting the local best is that the global best solution found thus far may lead to a local minimum in subsequent iterations, narrowing the search space. At each cycle, a local best solution can be used to overcome this. When a dragonfly has no neighbors, the global best and worst solution will be used.

Algorithm 1 Enhanced Dragonfly Algorithm

Input Challenging attack: dataset
Output Optimal Weight: X_o
Data X (population), no_gen (iterations), c (cohesion), s (seperation), a (alignment), f (friend), e (enemy)

```
1: Initialize $X, no\_gen$ $i = 1$
2: while $i \leq no\_gen$ do
3:     for each $x \in X_t$ do
4:         if $no\_of\_neighbors(x)! = 0$ then
5:             $l_{best}, l_{worst} = get\_local\_extreme(x, local_{best}, local_{worst})$
6:             Update w,s,a,c,f,e using equation
7:             Calculate $S_i, A_i, C_i$ using equation (1)-(3)
8:             $F_i = l_{best} - x$
9:             $E_i = l_{worst} - x$
10:            $\triangle x = (sS_i + aA_i + cC_i + fF_i + eE_i) + wx$
11:            $x = x + \triangle x$
12:        else
13:            $x = x + w * |g_{best} - x|$
14:        end if
15:    end for
16: end while
```

4.3.2 Negative Selection Algorithm (NSA)

The negative Selection algorithm [41, 42] is a kind of Artificial Immune System (AIS).AIS, a type of rule-based machine learning system [43, 44] and it is inspired by the human immune system. It works similarly to a pattern-based selection. There are two phases namely

1. Training phase

2. Testing phase

Each IIoT traffic of the dataset will be taken and compared to the current detectors using similarity scores throughout the training phase. We'll add the instance as a detector if it belongs to a new pattern or doesn't have any similarities to current detectors. Incoming IIoT traffic will be considered during the testing period. The similarity scores will be used to compare the incoming test traffic to the detectors. The similarity scores will be used to compare the incoming test traffic to the detectors.

Since the size of the IIoT dataset is large, checking each data point and then selecting it as a detector would be difficult. Hence, in our proposed algorithm, each IIoT attack in the dataset is clustered. The detectors will then be generated by taking random samples from each cluster. The average similarity score between each challenging IIoT attack and the test data is calculated and stored. The probability is determined by dividing the average score of each attack by the overall score. At the end of the test phase, we'll acquire the probability of test data belonging to each IIoT attack. The training and testing phase of NSA is presented in Algorithm 2 and Algorithm 3.

Algorithm 2 Negative Selection Algorithm: Training Phase

```
Input dataset: $Challenging\_attack$
Output Set of detectors: $detector$

1: Initialize $detector = []$
2: for each $class \in Challenging\_attack$ do
       $cluster\_class = Clustering(class)$
3:     for each $cluster \in cluster\_class$ do
           $data\_point = Take\ random\ points\ from\ cluster$
           $detector.add(data\_point)$
4:     end for
5: end for
6: return $detector$
```

4.3.3 Dempster Shafer's Combination rule

The Dempster-Shafer theory is a probabilistic combination strategy that has been used to combine the output from classifiers [45]. The output that we get from the DNN will be the probability of test data belonging to a particular IIoT attack. The output that we get from NSA will be also based on the probability that a data point belongs to a particular class. So, Dempster Shafer's theory will be used to combine the two. The following equation (19) presents the combination rule:

Algorithm 3 Negative Selection Algorithm: Testing Phase

Input Testing data instance: *test_point*
Output Probability of test instance belonging to each attack: *result*
Data *detector*

```
1: Initialize $i = 0, score = 0$
2: Initialize $result = []$
3:
4: for each $attack \in Challenging\_attack$ do
       $s = getSimilarity(test\_data, detector[attack])$
       $score = score + s$
       $result[attack] = s$
5: end for
6: for each $attack \in Challenging\_attack$ do
       $result[attack] = result[attack]/score$
7: end for
8: return $result$
```

$$m_{1,2}(A) = \frac{1}{K-1} \sum_{B \cap C} m_1(B) m_2(C) \tag{19}$$

where $m_1(B)$ is the probability obtained from deep neural networks and $m_2(C)$ is the probability obtained from novel Negative selection algorithm. K is a measure of the amount of conflict between the two mass sets.

$$K = \sum_{B \cap C = \phi} m_1(B) m_2(C) \tag{20}$$

5 Experimental Setup

The proposed work is implemenented using Python. For implementation, Google colab is being used. Our proposed model is evaluated using benchmark datasets such as CICIDS-2017, CICIDS-2018, and Ton IoT datasets. Accuracy, precision, recall, and F1-score are the performance measures that are used to evaluate DL-TL-NIDS. The aforementioned measures are computed from the confusion matrix. The following four terms make up the confusion matrix:

1. True Positive (TP): It is the number of test samples that is predicted positive and it's actually a positive sample
2. True Negative (TN): It is the number of samples that is predicted negative and it's actually a negative sample.
3. False Positive (FP): It is the number of samples that is predicted positive and it's actually a negative sample.
4. False Negative (FN): It is the number of samples that is predicted negative and it's actually a positive sample.

With the help of these terms, the performance metrics can be calculated. The calculation method for each performance metrics is described and the formula is given through equation (21)-(24).

1. Accuracy: This metric takes into consideration the number of samples correctly predicted. It is the ratio of the correct predictions to all the predictions made by the model.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{21}$$

2. Precision: It is the ratio of correct positive predictions made to the actual number of positive samples in the dataset.

$$Precision = \frac{TP}{TP + FP} \tag{22}$$

3. Recall: It is the ratio of correct positive predictions made by the model to the total positive predictions made by the model.

$$Recall = \frac{TP}{TP + FN} \tag{23}$$

4. F1-score: F1-score is the harmonic mean of recall and precision.

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{24}$$

The performance of our proposed model across different datasets is evaluated and compared against the existing state-of-art models. In both the levels, DNN opts for the optimized hyperparameters using Random Search, a hyperparameter optimization method. Table 1 summarises the default hyperparameter values utilized across all datasets. Learning rate, batch size, epoch size, and kernel initializer were the hyperparameters that were tuned via Random Search. Table 2 shows the results of the Random Search executed for each of the datasets mentioned above.

Table 1: Hyperparameter values

Hyperparameter	Value
Activation function for hidden layers	ReLU
Activation function for output layer	Softmax
Loss	Categorical-cross-category

5.1 CICIDS2017

CICIDS 2017 dataset was generated in 2017 by the Canadian Institute of Cyber Security (CIC) and the University of New Brunswick. The dataset composes

Table 2: Results of Random Search

Dataset	Learning rate	Epochs	Batch size	Kernel initializer
CICIDS 2017	0.01	70	100	uniform
CICIDS 2018	0.001	60	150	uniform
TON IoT	0.001	40	110	normal

of DoS, Bruteforce, PortScan, Web attack, and Infiltration attacks. It has 15 classes and 83 features. The upside of this dataset is that it has recorded up-to-date attacks, and the drawback is that it has profoundly imbalanced classes ranging from SQL Injection (0.0007%) to benign (80.30%).

Table 3 represents the performance of our proposed DL-TL-NIDS with the CICIDS-2017 dataset. The attacks that were highly misclassified (had accuracy/precision below 90%) in the first level detection were Benign, Bot and Portscan. The overall accuracy, precision, recall and F1-score is given in Table 4. The challenging attacks were fed for advanced detection. The false alarm rate (flagging Benign as an attack) was around 5% in the first level detection, and it has come down to 1% in second level detection.

Table 3: Performance of DL-TL-NIDS using CICIDS-2017 (Attack-wise)

Class	Accuracy (in %)	Precision (in %)	Recall (in %)	F1-Score (in %)
Benign	99.50	99.00	99.00	99.00
Bot	99.80	99.90	99.80	99.80
DDoS	99.86	100.00	100.00	100.00
DoS Goldeneye	100.00	100.00	100.00	100.00
DoS Hulk	99.99	99.00	99.00	100.00
DoS Slowhttptest	99.99	100.00	100.00	100.00
DoS Sloworis	99.90	99.00	100.00	100.00
FTP patator	99.83	100.00	100.00	100.00
Heartbleed	100.00	100.00	100.00	100.00
Infiltration	100.00	99.00	100.00	100.00
Portscan	99.70	99.59	99.50	99.54
SSH-patator	100.00	100.00	100.00	100.00
Web Bruteforce	100.00	100.00	100.00	100.00
Web attack SQL injection	100.00	100.00	100.00	100.00
Web attack XSS	100.00	100.00	100.00	100.00

Table 4: Overall Performance of DR-TL-IDS with CICIDS-2017

Class	Proposed DL-TL-NIDS
Accuracy (in %)	99.86
Precision (in %)	99.69
F1-score (in %)	99.88
Recall (in %)	99.82

In [46], DNNs were used for their study, changing the number of hidden layers from 1 to 5 and it's been referred here for comparison with DL-TL-NIDS.

For each attack, the best accuracy obtained by the existing work is taken, and we have compared it with our results. It can be inferred from Table 5 that our model performs well. The False Positive Rate (FPR) has been reduced in DL-TL-NIDS when compared with [46].

Table 5: Performance of DL-TL-NIDS with CICIDS-2018 dataset

	Accuracy (in %)	
Class	DL-TL-NIDS	DNN [46]
Benign	99.50	55.90
Web	100.00	98.50
FTP Patator	99.83	92.80
SSH-patator	100.00	95.90
DDoS	99.94	85.40
PortScan	99.70	84.00
Bot	99.80	98.20

5.2 CICIDS-2018

CSE-CIC-IDS 2018 dataset was generated in 2018 by the CIC and Communication Security Establishment. It's one of the most recent datasets produced by CIC. There are 15 classes and 83 features in the dataset. The major attacks generated were DoS, DDoS, SQL Injection, and Bruteforce.

The results are tabulated in Table 5. In the First level or simple detection, six attacks, namely, Benign, XSS, SQL Injection, DoS-SlowHTTPTest, Infiltration, and FTP-Bruteforce attacks had accuracy/precision less than 90%. The aforementioned attacks were marked as challenging attacks and fed for Level-2 detection. Table 7 presents the comparison, in terms of accuracy, of an existing state-of-art method [47] that employed ensemble learning along with feature selection. It can be inferred from Table 8 that the percentage of misclassification in our proposed method is far less than the ensemble method.

When compared against [48], which implemented CNN, DL-TL-NIDS performed well, especially the detection was better for Infiltration, FTP Bruteforce, and DoS attacks. Figures 5 to Figure 12 present comparisons between the [48] and DL-NL-TIDS in terms of accuracy, recall, precision, and F1-score.

5.3 TON_IoT

The dataset was generated by UNSW Sydney, especially for attack detection in IIoT. The dataset is generated by collecting information from sensors in IoT devices such as Garage, GPS-tracker, Fridge, and Thermostat. It comprises Backdoor, Command Injection, XSS, Scanning, DDoS, and Ransomware attacks.

Our proposed DL-TL-NIDS is compared against Kumar et.al [49], which presents Ensemble-Anomaly based detection. It can be inferred from Table

Table 6: Performance of DL-TL-NIDS with CICIDS-2018 dataset

Class	Accuracy (in %)	Precision (in %)	Recall (in %)	F1-Score (in %)
Benign	97.58	89.77	96.50	93.51
XSS	99.15	96.38	98.60	97.75
FTP	99.85	99.11	100.00	99.48
SQL	99.52	98.60	98.50	99.06
SlowHTTP	100.00	100.00	100.00	100.00
Infiltration	97.30	94.29	89.19	95.77
Bot	100.00	100.00	100.00	100.00
Web BruteForce	98.60	100.00	99.00	99.00
DDoS attack and HOIC	100.00	100.00	100.00	100.00
DDoS attack LOIC-UDP	100.00	100.00	100.00	100.00
DoS attack Goldeneye	100.00	100.00	100.00	100.00
DoS attack Hulk	100.00	100.00	100.00	100.00
DoS Slowworis	100.00	100.00	100.00	100.00
SSH Bruteforce	100.00	100.00	100.00	100.00
DDoS attack LOIC-HTTP	100.00	100.00	100.00	100.00

Table 7: Comparison of DL-TL-NIDS and [47] in terms of accuracy

Performance Metrics	DL-TL-NIDS	Ensemble (with feature selection)
Accuracy (in %)	99.97	98.80
Precision (in %)	99.76	98.80
Recall (in %)	99.74	97.10
F1-Score (in %)	99.75	97.90

Table 8: Misclassification comparision in [47] and proposed method

Class	Misclassification in CNN [47]	Misclassification in the proposed DL-TL-NIDS
Accuracy (in %)	99.97	98.80
Precision (in %)	99.76	98.80
Recall (in %)	99.74	97.10
F1-Score (in %)	99.75	97.90

Table 9: Performance Comparison of Proposed DL-TL-NIDS with [49]

Class	Proposed DL-TL-NIDS	Ensemble-Anomaly based IDS[49]
Accuracy (in %)	99.97	96.35
Precision (in %)	95.62	90.54
F1-score (in %)	99.65	95.03
False Alarm Rate (in %)	2.34	5.59

9 that the false alarm rate is reduced in our proposed model. Also, other performance metrics like accuracy and precision have been improved.

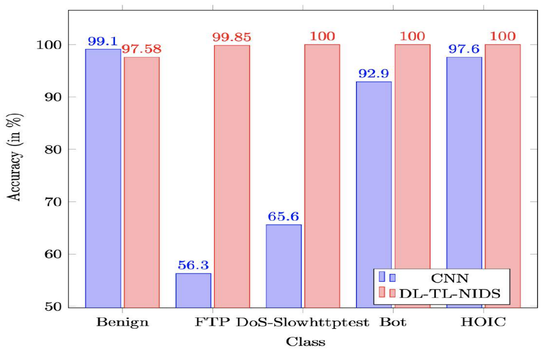


Fig. 5: Accuracy comparison of CNN [48] vs DL-TL-IDS

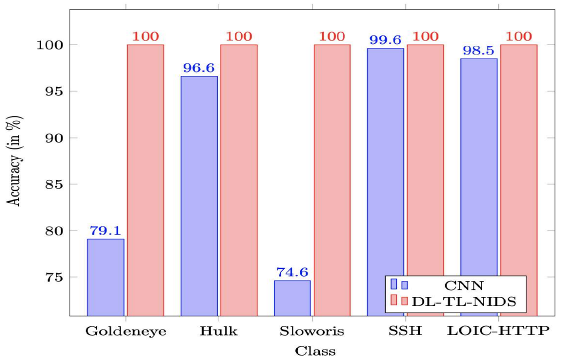


Fig. 6: Accuracy comparison of CNN [48] vs DL-TL-IDS

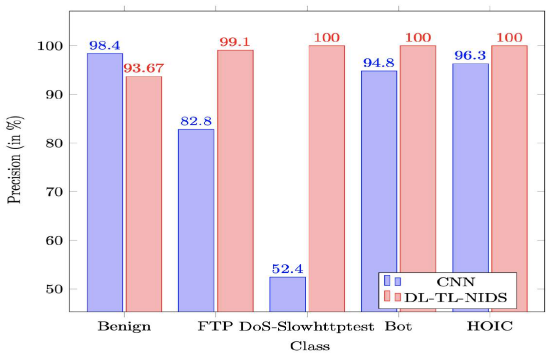


Fig. 7: Precision comparison of CNN [48] vs DL-TL-IDS

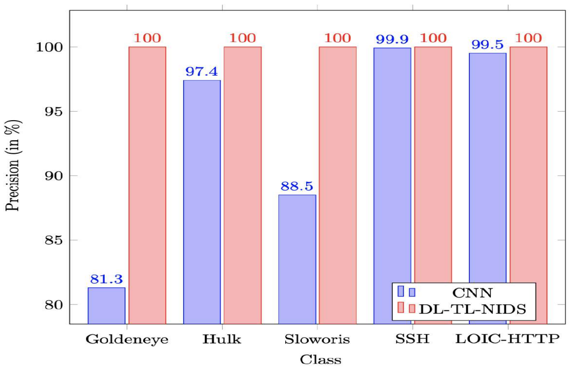


Fig. 8: Precision comparison of CNN [48] vs DL-TL-IDS

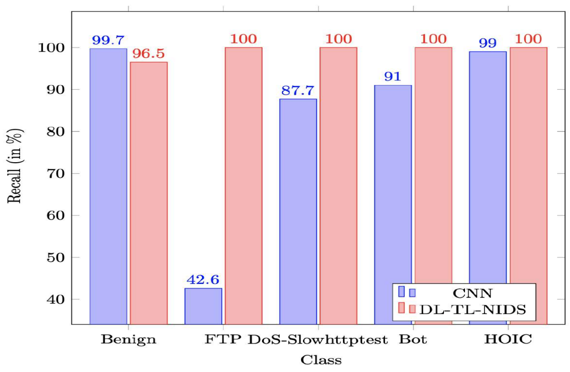


Fig. 9: Recall comparison of CNN [48] vs DL-TL-IDS

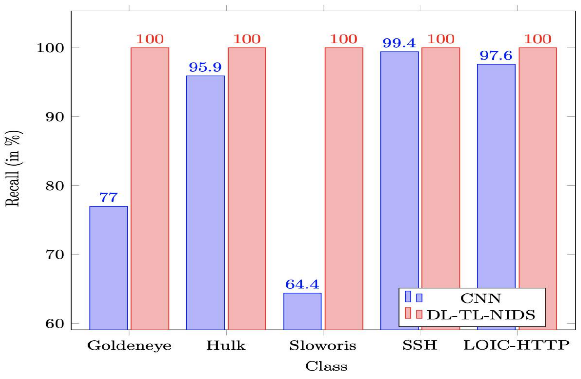


Fig. 10: Recall comparison of CNN [48] vs DL-TL-IDS

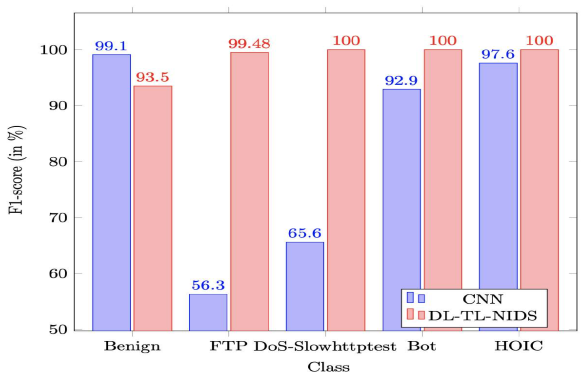


Fig. 11: F1-score comparison of CNN [48] vs DL-TL-IDS

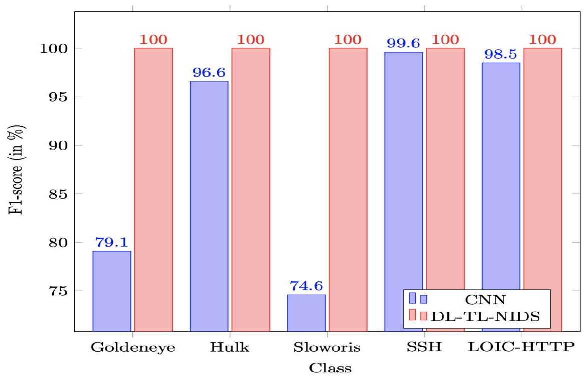


Fig. 12: F1-score comparison of CNN [48] vs DL-TL-IDS

6 Conclusion and Future Work

The Industrial Internet of Things (IIoT) provides a number of benefits. This study explored various applications, security issues induced by IIoT, and presented a Deep Learning-based two-level Intrusion Detection System. The proposed model first segregates challenging attacks in the first-level detection and performs Advanced detection in the second-level. The model is evaluated using benchmark and IoT and network datasets. It is demonstrated that it performed better than the existing state-of-the-art models.

In the future, the proposed two-level IDS can be extended to perform well on zero-day attacks or new attacks. Also, the proposed work can be extended to fit other IoT specific environments like the Internet of Medical Things (IoMT) and Machine-to-Machine (M2M), etc, to protect the devices and data.

Declarations

- Funding: The work is carried out without any financial support
- Conflict of interest/Competing interests: We declare that there is no conflict of interest
- Ethics approval: We declare that the work presented here is original and it is submitted nowhere.
- Consent to participate: Not Applicable
- Consent for publication: Not Applicable
- Availability of data and materials: Not Applicable
- Code availability: Not Applicable
- Authors' contributions

Role	Contributed Authors
Conceptualization	Kathiroli Raja, Krithika Karthikeyan, Abilash B
Methodology	Kathiroli Raja, Krithika Karthikeyan, Abilash B
Project Administration	Gunasekaran Raja, Kapal Dev
Software (Implementation)	Kathiroli Raja, Krithika Karthikeyan, Abilash B
Supervision	Gunasekaran Raja, Kapal Dev
Validation	Gunasekaran Raja, Kapal Dev
Visualization	Kathiroli Raja, Krithika Karthikeyan, Abilash B
Writing-Original Draft	Kathiroli Raja, Krithika Karthikeyan
Writing - Review & Editing	Gunasekaran Raja, Kapal Dev

References

[1] Pietraszek, T., Tanner, A.: Data mining and machine learning—towards reducing false positives in intrusion detection. Information Security Technical Report **10**(3), 169–183 (2005). https://doi.org/10.1016/j.istr.2005.07.001

[2] Khraisat Ansam, V.P. Gondal Iqbal, Joarder, K.: Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity **2** (2019). https://doi.org/10.1186/s42400-019-0038-7

[3] Meng, Yu-Xin: The practice on using machine learning for network anomaly intrusion detection. In: 2011 International Conference on Machine Learning and Cybernetics, vol. 2, pp. 576–581 (2011). https://doi.org/10.1109/ICMLC.2011.6016798

[4] Abirami, M.S., Yash, U., Singh, S.: Building an ensemble learning based algorithm for improving intrusion detection system. In: Dash, S.S., Lakshmi, C., Das, S., Panigrahi, B.K. (eds.) Artificial Intelligence and Evolutionary Computations in Engineering Systems, pp. 635–649 (2020). https://doi.org/10.1007/978-981-15-0199-9_55

[5] Ran, J., Ji, Y., Tang, B.: A semi-supervised learning approach to ieee 802.11 network anomaly detection. In: 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), pp. 1–5 (2019). https://doi.org/10.1109/VTCSpring.2019.8746576

[6] Jianliang, M., Haikun, S., Ling, B.: The application on intrusion detection based on k-means cluster algorithm. In: 2009 International Forum on Information Technology and Applications, vol. 1, pp. 150–152 (2009). https://doi.org/10.1109/IFITA.2009.34

[7] Zaman, M., Lung, C.-H.: Evaluation of machine learning techniques for network intrusion detection. In: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, pp. 1–5 (2018). https://doi.org/10.1109/NOMS.2018.8406212

[8] Faker, O., Dogdu, E.: Intrusion detection using big data and deep learning techniques. (2019). https://doi.org/10.1145/3299815.3314439

[9] Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence **2**(1), 41–50 (2018). https://doi.org/10.1109/TETCI.2017.2772792

[10] Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization, pp.

108–116 (2018). https://doi.org/10.5220/0006639801080116

[11] Moustafa, N.: A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. Sustainable Cities and Society **72**, 102994 (2021). https://doi.org/10.1016/j.scs.2021.102994

[12] Booij, T.M., Chiscop, I., Meeuwissen, E., Moustafa, N., den Hartog, F.T.H.: Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion datasets. IEEE Internet of Things Journal, 1–1 (2021). https://doi.org/10.1109/JIOT.2021.3085194

[13] Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., Anwar, A.: Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems. IEEE Access **8**, 165130–165150 (2020). https://doi.org/10.1109/ACCESS.2020.3022862

[14] Moustafa, N., Keshky, M., Debiez, E., Janicke, H.: Federated ton_iot windows datasets for evaluating ai-based security applications. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 848–855 (2020). https://doi.org/10.1109/TrustCom50675.2020.00114

[15] Moustafa, N., Ahmed, M., Ahmed, S.: Data analytics-enabled intrusion detection: Evaluations of ton_iot linux datasets. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 727–735 (2020). https://doi.org/10.1109/TrustCom50675.2020.00100

[16] Moustafa, N.: A Systemic IoT-Fog-Cloud Architecture for Big-Data Analytics and Cyber Security Systems: A Review of Fog Computing (2019)

[17] Moustafa, N.: New generations of internet of things datasets for cybersecurity applications based machine learning: Ton_iot datasets. In: Proceedings of the eResearch Australasia Conference (2019)

[18] Iotbot-ids: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities. Sustainable Cities and Society **72**, 103041 (2021). https://doi.org/10.1016/j.scs.2021.103041

[19] Abdel-Basset, M., Chang, V., Hawash, H., Chakrabortty, R.K., Ryan, M.: Deep-ifs: Intrusion detection approach for iiot traffic in fog environment. IEEE Transactions on Industrial Informatics, 1–1 (2020). https://doi.org/10.1109/TII.2020.3025755

[20] Jayalaxmi, P., Saha, R., Kumar, G., Kumar, N., Kim, T.-H.: A taxonomy

of security issues in industrial internet-of-things: Scoping review for existing solutions, future implications, and research challenges. IEEE Access **9**, 25344–25359 (2021). https://doi.org/10.1109/ACCESS.2021.3057766

[21] Qureshi, K.N., Rana, S.S., Ahmed, A., Jeon, G.: A novel and secure attacks detection framework for smart cities industrial internet of things. Sustainable Cities and Society **61**, 102343 (2020). https://doi.org/10.1016/j.scs.2020.102343

[22] Zolanvari, M., Teixeira, M.A., Jain, R.: Effect of imbalanced datasets on security of industrial iot using machine learning. In: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 112–117 (2018). https://doi.org/10.1109/ISI.2018.8587389

[23] Zhou, X., Hu, Y., Liang, W., Ma, J., Jin, Q.: Variational lstm enhanced anomaly detection for industrial big data. IEEE Transactions on Industrial Informatics **17**(5), 3469–3477 (2021). https://doi.org/10.1109/TII.2020.3022432

[24] Popoola, S.I., Adebisi, B., Ande, R., Hammoudeh, M., Anoh, K., Atayero, A.A.: Smote-drnn: A deep learning algorithm for botnet detection in the internet-of-things networks. Sensors **21**(9) (2021). https://doi.org/10.3390/s21092985

[25] Qiu, T., Chi, J., Zhou, X., Ning, Z., Atiquzzaman, M., Wu, D.O.: Edge computing in industrial internet of things: Architecture, advances and challenges. IEEE Communications Surveys Tutorials **22**(4), 2462–2488 (2020). https://doi.org/10.1109/COMST.2020.3009103

[26] Aboelwafa, M.M.N., Seddik, K.G., Eldefrawy, M.H., Gadallah, Y., Gidlund, M.: A machine-learning-based technique for false data injection attacks detection in industrial iot. IEEE Internet of Things Journal **7**(9), 8462–8471 (2020). https://doi.org/10.1109/JIOT.2020.2991693

[27] Zolanvari, M., Teixeira, M.A., Gupta, L., Khan, K.M., Jain, R.: Machine learning-based network vulnerability analysis of industrial internet of things. IEEE Internet of Things Journal **6**(4), 6822–6834 (2019). https://doi.org/10.1109/JIOT.2019.2912022

[28] Li, X., Xu, M., Vijayakumar, P., Kumar, N., Liu, X.: Detection of low-frequency and multi-stage attacks in industrial internet of things. IEEE Transactions on Vehicular Technology **69**(8), 8820–8831 (2020). https://doi.org/10.1109/TVT.2020.2995133

[29] Khoda, M.E., Imam, T., Kamruzzaman, J., Gondal, I., Rahman, A.: Robust malware defense in industrial iot applications using machine

learning with selective adversarial samples. IEEE Transactions on Industry Applications **56**(4), 4415–4424 (2020). https://doi.org/10.1109/TIA.2019.2958530

[30] Yan, X., Xu, Y., Xing, X., Cui, B., Guo, Z., Guo, T.: Trustworthy network anomaly detection based on an adaptive learning rate and momentum in iiot. IEEE Transactions on Industrial Informatics **16**(9), 6182–6192 (2020). https://doi.org/10.1109/TII.2020.2975227

[31] Bhunia, S.S., Gurusamy, M.: Dynamic attack detection and mitigation in iot using sdn. In: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), pp. 1–6 (2017). https://doi.org/10.1109/ATNAC.2017.8215418

[32] Latif, S., Idrees, Z., Zou, Z., Ahmad, J.: Drann: A deep random neural network model for intrusion detection in industrial iot. In: 2020 International Conference on UK-China Emerging Technologies (UCET), pp. 1–4 (2020). https://doi.org/10.1109/UCET51115.2020.9205361

[33] Yao, H., Fu, D., Zhang, P., Li, M., Liu, Y.: Msml: A novel multilevel semi-supervised machine learning framework for intrusion detection system. IEEE Internet of Things Journal **6**(2), 1949–1959 (2019). https://doi.org/10.1109/JIOT.2018.2873125

[34] Liang, W., Li, K.-C., Long, J., Kui, X., Zomaya, A.Y.: An industrial network intrusion detection algorithm based on multifeature data clustering optimization model. IEEE Transactions on Industrial Informatics **16**(3), 2063–2071 (2020). https://doi.org/10.1109/TII.2019.2946791

[35] Yan, Q., Huang, W., Luo, X., Gong, Q., Yu, F.R.: A multi-level ddos mitigation framework for the industrial internet of things. IEEE Communications Magazine **56**(2), 30–36 (2018). https://doi.org/10.1109/MCOM.2018.1700621

[36] Mighan Soosan Naderi, K.M.: A novel scalable intrusion detection system based on deep learning. International Journal of Information Security, 387–403 (2021). https://doi.org/10.1007/s10207-020-00508-5

[37] Maharani, M.P., Tobianto Daely, P., Lee, J.M., Kim, D.-S.: Attack detection in fog layer for iiot based on machine learning approach. In: 2020 International Conference on Information and Communication Technology Convergence (ICTC), pp. 1880–1882 (2020). https://doi.org/10.1109/ICTC49870.2020.9289380

[38] Zhong, W., Yu, N., Ai, C.: Applying big data based deep learning system to intrusion detection. Big Data Mining and Analytics **3**(3), 181–195 (2020). https://doi.org/10.26599/BDMA.2020.9020003

[39] Peddabachigari, S., Abraham, A., Grosan, C., Thomas, J.: Modeling intrusion detection system using hybrid intelligent systems. Journal of Network and Computer Applications **30**(1), 114–132 (2007). https://doi.org/10.1016/j.jnca.2005.06.003

[40] Peddabachigari, S., Abraham, A., Grosan, C., Thomas, J.: Dragonfly algorithm and its applications in applied science survey. Computational Intelligence and Neuroscience (2019). https://doi.org/10.1155/2019/9293617

[41] Igawa, K., Ohashi, H.: A negative selection algorithm for classification and reduction of the noise effect. Applied Soft Computing, 431–438 (2009). https://doi.org/10.1016/j.asoc.2008.05.003

[42] Gong, M., Zhang, J., Ma, J., Jiao, L.: An efficient negative selection algorithm with further training for anomaly detection. Knowledge-Based Systems, 185–191 (2012). https://doi.org/10.1016/j.knosys.2012.01.004

[43] Dutt, I., Borah, S., Maitra, I.K.: Immune system based intrusion detection system (is-ids): A proposed model. IEEE Access **8**, 34929–34941 (2020). https://doi.org/10.1109/ACCESS.2020.2973608

[44] Ou, C.-M.: Host-based intrusion detection systems adapted from agent-based artificial immune systems. Neurocomputing, 78–86 (2012). https://doi.org/10.1016/j.neucom.2011.07.031

[45] Surathong, S., Auephanwiriyakul, S., Theera-Umpon, N.: Decision fusion using fuzzy dempster-shafer theory. In: Recent Advances in Information and Communication Technology 2018, pp. 115–125 (2019)

[46] Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., Venkatraman, S.: Deep learning approach for intelligent intrusion detection system. IEEE Access, 41525–41550 (2019). https://doi.org/10.1109/ACCESS.2019.2895334

[47] Fitni, Q.R.S., Ramli, K.: Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems. In: 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), pp. 118–124 (2020). https://doi.org/10.1109/IAICT50021.2020.9172014

[48] Kaur, G., Habibi Lashkari, A., Rahali, A.: Intrusion traffic detection and characterization using deep image learning. In: 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 55–62 (2020). https://doi.

org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00025

[49] Kumar, P., Gupta, G.P., Tripathi, R.: An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for iomt networks. Computer Communications, 110–124 (2021). https://doi.org/10.1016/j.comcom.2020.12.003