International Journal of Communication Networks and Information Security
2024, 16(1), 6358
ISSN: 2073-607X,2076-0930
https://https://ijcnis.org/
Research Article

A Constructive Model for Cyber-Attack Prediction Using Efficient Weighted Bi-Directional Learning Approaches

Bondili Sri Harsha Sai Singh [1], Mohammed Fathima [2], Thota Teja Mahesh [3], Mohammad Sameer [4], Dinesh Kumar Anguraj [5], Padmanaban Kuppan [6]

[1,2,3,4,5,6]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddesswaram, Guntur, Andhra Pradesh, India.
*__Corresponding Author:__ dineshinnov@outlook.com

Citation: B. S. H. S. Singh, M. Fathima, T. T. Mahesh, M. Sameer, D. K. Anguraj, K. Padmanaban, "A Constructive Model for Cyber-Attack Prediction Using Efficient Weighted Bi-Directional Learning Approaches," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 16, no. 1, pp. 1-18, Jan. 2024.

ARTICLE INFO

Received: 12 Jan 2024
Accepted: 15 Apr 2024

ABSTRACT

Anomaly detection algorithms based on machine and deep learning are currently the most promising techniques for identifying cyber-attacks. However, hostile attacks lower forecast accuracy which is made against these techniques. The resilience of anomaly detection has been measured using a variety of methods in the literature. They neglect to consider the fact that a little disruption in an anomalous sample caused by an assault like a denial of service might cause it to become a genuinely normal sample, but a huge perturbation can transform an anomalous sample into a truly normal sample without affecting the whole system. Even so, it can lead to it being wrongly classified as normal. The approach for determining an anomaly detection model's resilience in industrial contexts is presented in this work. To detect abnormalities brought on by various cyber-attacks; this work used the method of a Support Vector Machine (SVM) for feature extraction and weight analysis. In this case, a unique deep learning-based Bi-LSTM (Bi-directional Long Short Term Memory) only requires a disruption of 60% with 99.6% accuracy of the original sample to create adversarial samples as opposed to the model, which requires a disruption of the entire original sample.

Keywords: Cyber-Attack, Prediction, Deep Learning, Feature Representation, Anomaly

INTRODUCTION

Industry 4.0, or the fourth industrial revolution, is now in progress. It is largely propelled by the industrial processes' adoption of new computer concepts and technology. We highlight the adoption of big data, the Industrial Internet-of-Things (IIoT), which connects numerous the fifth generations of mobile networks (5G), which enables communications to have low high bandwidth [1], delay and optimizes the analysis of massive volumes of data, connects diverse and devices with limited resources to the Internet. Industry 4.0 has benefits, yet it's also making it easier for new cyber-attacks to target industrial equipment and vital operations [2]. Traditional security methods soon become ineffective due to the frequency and variety of cyberattacks. Due to the increasing amount of highly specific and one-of-a-kind the academic community is utilizing machine learning (ML) and deep learning (DL) techniques to identify attacks in a semi-supervised or unsupervised way for (zero-day) attacks impacting many businesses [3]. The most promising and efficient ways to identify hidden attacks in the existing situation involve systems for anomaly detection (AD) that use ML and DL [4]. In industrial processes, these systems can differentiate between normal and aberrant behaviour in contrast to conventional methods, without depending on databases that now preserve the patterns of cyber-attacks.

However, adversarial attacks may be used to compromise the present AD solutions based on ML/DL, rendering them inappropriate for use in actual systems. Adversarial attacks involve manipulating ML/DL models to change the behaviour of the models or obtain sensitive data. Evasion attacks are among the most pertinent adversarial attacks now in use since they are used in the system's assessment phase after the model has been trained. The basic objective of evasion attacks in malware-affected industrial contexts is to create samples that mimic the behaviour of the virus (Anomaly samples), and erroneously identify them (as normal samples), allowing the virus to surreptitiously harm commercial equipment or processes.

Adversative attacks provide fresh security and trust issues that have an impact on methods for detecting cyber-attacks based on AD, and ML/DL in general. Data scientists are already putting in great effort to provide correct and reliable AI-based solutions in a variety of application settings [5]. The pillars required to create trustworthy AI have recently been highlighted by IBM [6]. Robustness is one of these pillars, and its major objective aims to assess how resistant ML/DL models are to malicious assaults. once the degree of robustness has been determined, it may either be utilized in adversarial training, which modifies the network using adversarial samples, to raise the model's resistance, or in addition to conventional performance indications, it can be provided to end users.

The literature has provided a variety of measures to gauge the robustness of a model. The most often used metrics are empirical robustness (ER), local loss sensitivity (LLS), and cross-Lipschitz extreme value for network robustness (CLEVER) [7]. The computer vision field is only one of the many application areas where these measures are quite helpful. When used to assess the durability of AD in business environments, they have limitations [8]. The inability to distinguish between an adversarial sample and a malicious sample that fools the anomaly detection that is excessively turned into a normal sample change is one of the most important constraints. Think about a water distribution procedure, for instance, where a cyber-attack known as a denial of service (DoS) is conducted. The objective of this cyberattack is to cut off the water supply for a certain location. The valves that regulate the water supply accept values between 0 and 1, or entirely closed to fully open. To shut the valves and stop the supply, the DoScyber-attack can thus change such properties. Furthermore, an attacker may alter the DoS samples to make them hostile if they want to carry out a DoScyber-attack without being detected by the AD system [9].

The DoScyber-attack would have no effect if these characteristics took the value 1 (totally exposed), though, because of the severe disruption [10]. The adversarial attack is seen as successful in both situations, although it has no detrimental effects on the industrial gadget in the second. As a result, a technique is required to distinguish between these two adversarial versions and offer a trustworthy indicator of the model's resilience [11]. The diversity of data formats employed in industrial settings is another issue. There are discrete values, continuous values, or even timestamps, typically with internal consistency constraints, which makes it difficult to calculate a gradient or create a reliable adversarial sample, in contrast to image recognition and other domains like audio signals, where values frequently float. The following additions are made in the current study to address the prior constraints [12]. In some ML and DL-based approaches for testing to determine if a prediction model is robust in industrial settings contexts, numerous auxiliary DL models (support models) are used because abnormality persists in a poor sample. This method offers a robustness metric that modifies the performance metric while taking into account four key phases. It is important to note that the suggested technique focuses on evaluating rather than creating a strong model and emphasizes the adaptability of a trained AD model. An analysis of an online dataset was used to validate the recommended approach [13], a genuine, though simulated, industrial setting. Some authors notably show how standard Long Short-Term Memory (LSTM) and 1D-CNN models calculate resilience while dealing using time-series information. The most resilient model should be considered for deployment in a real setting. According to tests, the robustness of the model is 1.1, which is about twice as strong as the LSTM model's. The major contributions of this work are:

1) The input data is taken from online resources where the pre-processing step is adopted to enhance the work quality and performance;

2) The feature extraction process is done with SVM to predict the features of attacks. It helps to prevent the spread of attack consequences and predicted in earlier stages;

3) The classification process is done with BiLSTM to provide accurate prediction outcomes with better outcomes than conventional approaches. It helps to reduce error by sequentially analysis and prediction without aggressive process; Metrics like the Detection Rate (DR) and False Alarm Rate (FAR) are used to assess performance.

The project is structured as follows: Section 2 lists relevant gaps and active projects. The procedure is described in greater depth in Section 3. The numerical findings are presented in Section 4, and the conclusion is presented in Section 5.

LITERATURE REVIEW

The most current techniques of network intrusion for SDN are covered in this section. A variety of ML and DL-based botnet technique identification attacks have recently been developed by researchers [14]. Recognizing botnet and DDoS attacks has been the subject of some pertinent research. Su et al. presented "high-level PSI-rooted subgraph-based features" and also employed a hybrid approach that combines botnet detection on the IoT using deep learning and machine learning techniques [15]. Their main objective was to swiftly and effectively identify the attack using the fewest attributes possible to save space. An unsupervised learning-based IoT botnet sensor system was created by Gamage et al. [16]. They use the "Grey Wolf Optimization (GWO) algorithm" to perform hyperparameterization on SVM to identify the key characteristics of a botnet attack. The recommended method aids in identifying IoT botnet attacks launched by knowledgeable, exposed nodes [17]. We found N-BaIoT, a different network-based solution to anomaly detection that uses deep auto-encoders to find attack traffic while taking snapshots of network activity. Aldweesh et al. proposed a hybrid PSO algorithm along with a voting mechanism to detect botnet attacks in IoT environments [18].

The PSO is utilized in this technique to choose the important and noteworthy characteristics, DNN, Decision Tree (DT) C4.5, and SVM are used in the election process to detect botnet attacks. A CNN-based technique called BoTIDS was suggested by Ferran et al. [19]. In comparison to LSTM and RNN, they found that their method produced superior outcomes when applied to both the entire dataset and a set of 10 characteristics that were chosen at random. However, compared to the entire dataset, the assessment findings on a subset are superior. We noticed yet another Anomaly Intrusion Detection System (AIDS) based on CNN [20]. Datasets from Network Intrusion Detection (NID) and BoT-IoTare were used to evaluate this system. When compared to BoT-IoT, the suggested system achieved efficient results on the NID dataset.

Hassan et al. discovered a two-level DDoS assault detection method based on information entropy (IE) and DL [21]. The suspicious port's components must be located in coarse granularity they first used information entropy. Next, they employed a fine-grained detection strategy based on CNN to differentiate between both the attack and regular traffic. For the purpose of demonstrating the efficacy of DL approaches for the detection of hazardous traffic, A DNN-based network intrusion detection system was created [21]. DDoS attacks may be detected in SDN settings using the Autoencoder and RNN combination approach known as DDoSNet [22]. A technique for detecting DDoSLSTM was presented by Yin et al. and evaluated its effectiveness in comparison to a Random-Forest (RF) approach [23]. The error rate was lowered from 7.517% to 2.103% by employing this technique. Wang et al. depict the LUCID DDoS detection system, as a straightforward and useful deep learning-based solution demonstrating CNN's capacity to categorize traffic flows as legitimate or malicious [24]. The approach described by Yang et al. relies on SVM and is supported by Genetic Algorithm (GA) and Kernel Principle Component Analysis (KPCA) [25]. The inventors of this approach tuned the hyper-parameters of the SVM by using GA and KPCA to minimize the dimensionality of the features. They offer an improved Kernel Function (N-RBF) to lessen the noise brought on by feature fluctuations. Otomo et al. conducted research to identify the DDoS attack using neural networks using Particle Swarm Optimization (PSO)-BP and standard entropy measures [26].

Li et al. made use of the first use of trigger mechanism to increase DDoS attack detection performance. The overhead of the controller and switches is decreased by using this technique [27]. In the study by Kunang et al., another hybrid approach based on DNN and ANNs was presented [28]. The authors integrated the RNN and LSTM methods to categorize Emotet, Zbot, Dridex, and Salty are the four most significant bot assaults [29]. In the study by Sherubha et al., a different hybrid approach was recommended based on CNN and LSTM [30]. Information gathered from nine commercial IoT gadgets was used by them to deploy this strategy in a genuine testbed, which they used to identify the attack. To identify DDoS and phishing attacks, the authors of [31] suggested a distributed method built on the foundation of LSTM and CNN with a further cloud-based component. Gamage et al. described a technique for detecting DDoS attacks that combines Deep Belief Network (DBN)-inspired Fuzzy and Taylor-Elephant Herd Optimization (F-TEHO) [32]. The three components that make up this methodology are extraction, selection, and categorization of features. To choose the most important characteristics, the Holo-entropy approach is utilized after the unprocessed packet data, and the feature extraction module extracts features. The FTEHO approach is then utilized to complete the categorization procedure. A new strategy based on dual address entropy and computing influenced by cognition is suggested in [33]. This method initially extracts the characteristics from the flow table using the dual address entropy before classifying the traffic as normal or an attack. This technique makes it easier to identify an attack early on and quickly resume regular communication [34-35].

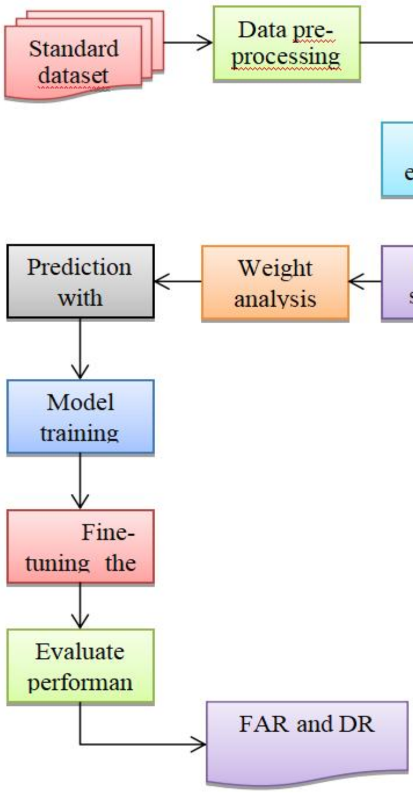

Figure 1. Flow Diagram of the Proposed Model

METHODOLOGY

Identifying and thwarting attacker attack flows is crucial for successfully defending the IoT environment from DDoS attacks using botnets. A flow is made up of several packets that contain identical information (source and destination port numbers, source and destination IP addresses, and other protocol-related information) and is sent in both directions. In an attack flow, the originating IP address belongs to the attacker. Assume we have y classifications and N flow samples. Assume, that each flow sample is represented by the notation $X = \{F_1, F_2, F_3, \dots, F_N\} \in R^{d*N}$, where N is the total number of flows, F_i is the ith flow, and d is the number of original features incorporated in a flow. The real tables of a F_i flow are $y_i = 0$ and 1. We want to create a complete procedure ($y_{pred(i)} = y_i$) for anticipating a label as the real label.

The comparative research of deep learning and feature selection for flow categorization is covered in this part. It would increase effectiveness and accuracy and assist in categorizing the normal and attack flows. The planned research's operational phases are depicted in Figure 1. The first step is to arrange the incoming parcels on a lovely table. Second, each packet flow is independently calculated and extracted. Third, the best feature threshold values for the feature weights are used to choose the best features. Finally, a subset of the chosen features is created, and this feature subset is then used as input for the DL classifiers.

Dataset

The experiments' websites are taken from the actual network environment. The collection of safe websites comes from Alexa. The world rankings of websites are made public on the Alexa website, which is run by Amazon. It includes a large amount of URLs and comprehensive website rating data. We gather popular Alexa lists of websites that are deemed to be safe. We obtained 24,800 regular web pages from Alexa after removing a few incorrect, error, and duplicate pages. The collection of phishing websites is from PhishTank.com. An up-to-date and reliable list of phishing websites may be found at the well-known phishing webpage collecting website PhishTank.Before publication, PhishTank gathers all suspected phishes reported by anyone and assesses each one to see if a fraud attempt has been made. Due to the short lifespan of phishing websites from September 2019 to November 2019 gathered 21,303 PhishTank-listed phishing websites and pre-processed those that did not adhere to the grammatical norms. Between the training and test sets, there is a ratio of 0.75:0.25.

Pre-Processing

Port and protocol information, flow ID, source and destination IP addresses, and date are some of the characteristics that are left out to make the dataset more general. By eliminating the aforementioned attributes, the DL techniques adopt the broader DL methods' inability to attribute certain IPs, ports, and protocols as attack nodes. After that, empty and unbounded values are screened out of the dataset. Following normalization, the dataset is labelled with 0 and 1 for the normal and attack flows, respectively. There are 41,242 attack flows and 48,390 normal flows in the sample as was mentioned in the section above. When testing DL algorithms in real-time environments where the proportion of normal to attack flows is consistent, the imbalance is maintained in the dataset.

Feature Extraction

In the context of an IoT network, the Pcap format is used to capture the dataset. To train deep learning algorithms, we converted the dataset from pcap format to CSV using the CIC Flow Meter V4. With 83 features, CIC V4 generates bidirectional network flow statistics from the recorded file. A collection of unidirectional packets makes up a network flow that follows a certain protocol and moves from source to destination over time. The transformed dataset contains 89,632 flow records overall and 83 characteristics. The dataset also includes the same 83 characteristics for 41,242 attack flows and 48,390 normal flows.

Feature Selection

Even though some of the aforementioned elements are important for determining attack vectors, others might not have much of an impact on the classification outcomes, they all increase computational time and cost. To improve classification accuracy and raise performance, we must choose the best characteristics that can differentiate between the regular and assault flows. The research initial dataset included 83 features, which were refined down to 15 key features using SVM for weighting and iterative selection, allowing us to enhance model efficiency and maintain computational effectiveness. In our research, we employ a feature selection technique that chooses an ideal subset of features from a given collection of characteristics without altering the original features, i.e. $\{F_1, F_2, \ldots, F_d\}$ where d is the high-dimension features of X, and the number of characteristics is d. We propose a technique that can convert features with high dimensions d into features with low dimensions $r(r < d)$ to choose the best flow features for attack recognition. Our research's two concepts of feature weighting and threshold adjustment form the foundation of the feature selection approach. Algorithm 1 shows the feature selection process in action.

Algorithm 1:

```
Input: Set feature $F = \{F_1, F_2, \ldots, F_d\}$, threshold weight $\propto$, no. of chosen features r.
Output: Chosen feature set $F'$
Computer weighted features
1. Compute $F'$;
2. $For(i = 0; i < d; i++)do$
3. Evaluate weight $(F_i)$ based on $r(i)$;
4. $if(|r(i) < \propto)then$
5. Compute features $F_i$;
6. else
7. $F = F_i$;
8. end if
9. end for
10. Evaluate the feature set with $(F[\quad])$
11. Return$F'$
```

Weight Analysis

In our trials, we divided the entire dataset into five feature subsets with the best features using iterative wrapper-based feature selection using SVM. Based on how well each characteristic predicts the outcome, the SVM gives weights to each one. For identifying the attack, the characteristics with greater weights are regarded as crucial. The absolute values that the SVM allocated to the weights for various attributes are provided. During iteration, the attributes are chosen based on a threshold value produced by the weights' threshold tuning procedure. A feature subset of the characteristics that had weights larger than or equal to the threshold

(F_n (w_n) $\geq$ α) was chosen. The DL classifiers were then fed feature subsets with the best features as input. Based on feature weights, a straightforward threshold tuning technique chooses the best threshold value. The tuning approach calculates a cut-off value that lies between the lowest and maximum feature weights. As a cut-off that can shrink the dimension of the features, the ideal threshold values can be calculated.

Prediction Model

The comparative research of deep learning and feature selection for flow categorization is covered in this part. It would increase effectiveness and accuracy and assist in categorizing the normal and attack flows. The first step is to arrange the incoming parcels on a lovely table. Second, each packet flow is independently calculated and extracted for the properties listed in Table 1.

Table 1. Parameter Setup

Parameter	**Value**
Network size	3
Kernel size	128
Strides	1
Total cell	128
Batch	64
Dropout	0.5
Epochs	10
Learning rate	0.001

Third, the best feature threshold values for the feature weights are utilized to choose the top characteristics. The selected qualities are then aggregated into a feature subset, which is then given as input to the maximum pooling of the DL classifiers. The representation X_i of the network i serves as the convolutional layer's input. The matrix R_i the following convolution is: for a certain convolution kernel W.

$$R_j = f(W \otimes V_{j;j+h-1} = b) \tag{1}$$

When the bias is represented by b, $\otimes$ the convolution process is indicated by, and the activation function by f (•). Convolution is followed by 1-Maxpooling, which takes the feature map's most significant properties and decreases network parameters:

$$X_j = \max(R_j) \tag{2}$$

By combining the input data with convolution, CNN extracts the local characteristics. Due to its tolerance for noise and distortion, it can deal with common obfuscation strategies that do not alter harmful attacks better.

Algorithm 2:

```
Input: Dataset, learning rate, optimizer, epochs, round
Output: Prediction accuracy
1. Set learning parameters based on the prediction model;
2. Initialize network parameter based on random values;   //vector matrix
3. For training round do
4. Choose subset from dataset to form training batch;
5.  if no. of rounds = 0 then
6. Return accuracy
7. end if
8.  Initialize batch and evaluate labels and prediction values y;
9. Compute loss among the actual and predicted values;
10. Compute loss for provided optimizer;
11. Update vector matrix with learning rate (p);
12. if training round reaches the epochs then
```

```
13. Terminate classifier training process;
14. end if
15. end for
```

Bi-LSTM

By combining LSTM in both forward and reverse directions, the BiLSTM is generated and receives the output vector from CNN. In typical neural networks, the hidden layer is replaced with LSTM, which employs a memory cell structure. Figure 2 depicts its cell structure. The main components of an LSTM the input gate, output gate, and forget gate all belong to a memory cell. Below is a description of the t^{th} cell's updating procedure.

$$\begin{Bmatrix} i_t \\ f_t \\ o_t \end{Bmatrix} = \sigma\left(\begin{Bmatrix} W_i \\ W_f \\ W_o \end{Bmatrix} h_{t-1} + \begin{Bmatrix} U_i \\ U_f \\ U_o \end{Bmatrix} x_t + \begin{Bmatrix} b_i \\ b_f \\ b_o \end{Bmatrix}\right) \tag{3}$$

$$\tilde{C} = \tanh\left(W_c h_{t-1} + U_c x_t + b_c\right) \tag{4}$$

$$C_t = f_t \,.\, C_{t-1} + i_t \,.\, \tilde{C}_t \tag{5}$$

$$h_t = o_t \,.\, \tanh\left(C_t\right) \tag{6}$$

Here, "x_t" stands for "current input, "i," "o," "f," "C˜", "C "h," temporary memory cell state, memory cell state and hidden layer output value respectively. The weight matrices are denoted by W_f, U_f, W_i, U_i, W_o, U_o, W_c, and U_c while biases are denoted by b_f, b_i, b_i, and b_c. Finally, BiLSTM creates the output feature vectors gathered from both directions inside cell t. The connections between website sections make perfect sense. From the websites, many different time scales and long-distance connections may be learned via BiLSTM with success. Based on the local features that CNN has learned, it can then extract possible semantic information. By focusing on a single piece of information, the very fine-grained qualities that may be extracted from sequence data are accomplished via the attention mechanism. Each feature item in the BiLSTM-produced feature vector has a different effect on the degree to which the phishing website may be detected. The model's complexity primarily stems from the SVM and Bi-LSTM components. The SVM operates with a computational complexity of O(n^3), and the Bi-LSTM processes data with a complexity of O(tdh), where t is the sequence length. This complexity ensures robustness while balancing the computational load.

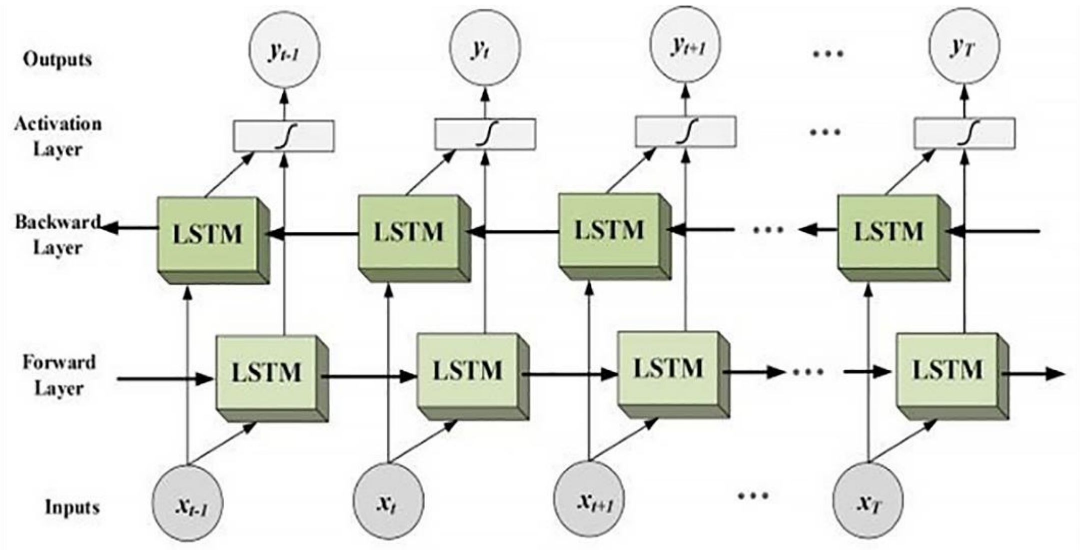

Figure 2. Bidirectional LSTM

The model can improve its capacity for decision-making for crucial aspects by providing various features varying attention. The following are the calculating formulas:

$$a = \tanh\left(h\right) \tag{7}$$

$$\propto\ = \mathrm{softmax}\left(w^T a\right) \tag{8}$$

$$x = \sum_t \propto h \tag{9}$$

Where w stands for weight, h denotes the attention of h, represents the output of the BiLSTM, and after weighted summing, the feature vector is known as x. After the features have been recovered, the feature vectors from each channel are combined to create the fusion vector Xi for the site. In the following step, the fully connected layer and sigmoid function are used to forecast the website category. During the training phase, the difference between the actual value and the predicted value is computed using a cross-entropy loss function. The loss function is as follows if the actual category and anticipated value are i and yi, respectively:

$$J(Y', Y) = -\frac{1}{n}\sum_{i=1}^{n} [y_i \log y_i' + (1 - y_i) \log (1 - y_i')] \tag{10}$$

RESULTS AND DISCUSSION

The tests are carried out using the network controller in the virtual environment. In our tests, we made use of MATLAB 2020a which is compatible. MATLAB 2020a is frequently used for simulating IoT-based networks that are free to use and support network protocols. The operating system, hardware specs, and RAM are Intel Core i7, 8GB. The MATLAB 2020a framework is used to implement the deep learning classifiers in MATLAB 2020. The customized centralized IoT-based network architecture is created for experimentation. In this architecture, a complicated tree network structure is used, moreover, on the data layer, hosts and switches are centrally controlled by a network deployed in the control layer. The elements that make up the application layer are the feature extractor, DL classifier, flow statistics collector, and mitigator. The system structure also includes switches and a controller. Six hosts are linked to switch S1, while three hosts are connected to each of the other switches. The "ping" command is run on every host once the topology has been successfully developed to confirm that every host can connect to the other hosts. Our target server is H13, and we chose H2 as the bot master, along with the bots H3, H4, H5, and H6. To create genuine or background traffic, the other hosts are employed. The traffic is sent by the switches, while the controller manages the entire network and spots attacks.

In our tests, in an IoT environment, we conducted botnet-based DDoS strikes using Python scripts. After the topology has been established, to make host H13 a target server, we run "target.py" on it. Then, on H2, H3, H4, H5, and H6, respectively, "botmaster.py" and "bot.py" are run. The idea of socket programming was used to build particular ports for the bots. The bots wait for the botmaster commands at their designated ports. The botmaster sends a message to all of the bots with the date, time, and preparation instructions for the attack. The target server receives the attack traffic after all of the bots' dates and times match the time specified by the bot-master. The onslaught lasted 14.26 minutes in total.

To provide typical or background traffic, this work employed a Distributed Internet Traffic Generator (D-ITG). Each host issues the ITGSend and ITGRecv commands to transfer and receive data. We inject more than 200 flows onto the network as background flows to make the background activity seem real. For the transmission rate of each flow, the TCP protocol follows constant, uniform, exponential, Poisson, and gamma distributions. Additionally, we change the packet size for each flow using several distributions, including gamma, Poisson, exponential, uniform, and constant. Fig. 3 depicts an illustration of a flow rule used by the D-ITG to provide regular traffic. Flow rules are used on many sites to produce and accept traffic, and each row. As an illustration, after the second backdrop flow in the third background flow occurs in the red box, and so on. Host H13 (10.0.0.13, port 5000) is the required one. The amount of packets delivered is equally split between 500 and 1000, and each packet has 512 bytes. The traffic in this flow, which is based on the TCP protocol, lasts for 12000 milliseconds. We can see that the network has successfully received an immunization against background traffic.

Experimental Results

The assessment metrics mentioned below are widely utilized to show off the DL-based IDS's efficiency and performance. Among the performance measurements are the following metrics: F1 score, specificity, accuracy, detection rate (DR), and false positive rate (FPR). These metrics are calculated using the network anomaly classification confusion matrix. These performance measurements can be used to evaluate the effectiveness of any approach that has been put into use. In contrast to True Positives (TP), which are attack records that are correctly classified as attacks, False Positives (FP) are attack records that are correctly identified as assaults, according to the confusion matrix and attack records that are wrongly labelled as normal is known as False Negatives (FN).It is possible to quantify each of these assessment measures as follows:

Accuracy: The percentage of accurate answers identified normal and attack recordings to all recordings is used to measure accuracy.

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{FP} + \text{FN} + \text{TP}} \quad (11)$$

Detection Rate (DR): A precise ratio identified recordings of actual attacks is known as DR.It also goes by the name Sensitivity or True Positive Rate (TPR). It's calculated as:

$$\text{Detection rate} = \text{recall} = \text{TPR} = \frac{\text{TP}}{\text{FN} + \text{TP}} \quad (12)$$

False Positive Rate (FPR): FPR calculates the proportion of normal records that were mislabelled as attack records.

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}} \quad (13)$$

Precision: The proportion of accurately recorded attack recordings which are genuine recordings of attacks is called precision.

$$\text{Precision} = \frac{\text{TP}}{\text{FP} + \text{TP}} \quad (14)$$

F1 Score: This acronym stands for the harmonic mean of recall and accuracy. When a dataset is used to train an unbalanced method, efficiency is regarded as being more important than accuracy. A DL method's F1 score is calculated as follows:

$$\text{F1} - \text{score} = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} * \text{Precision}} \quad (15)$$

Any implemented DL approach is considered to be the best if it has high accuracy, DR, precision, and F1 scores. In a similar vein, if the FPR is low.

Feature Selection Outcomes

Using strategies for feature weighting and threshold tweaking, we determined the significance of each feature and divided them into five subgroups. The best threshold value was chosen using the weight values of all characteristics in the threshold tuning approach. A feature subset was chosen and comprised of features with weights that were equal to or more than the cut-off point. All features are included in F_1. The tuning procedure selects 43 characteristics with weights of at least 1.80 to be incorporated into F_2, giving a value of $\alpha \geq 1.8$ as the ideal result. The best threshold value was found to be F_3, which has 30 features with weights of $\alpha \geq 2.70$. Similarly, 23 characteristics from F_4 were chosen based on $\alpha \geq 3.15$. Last but not least, F_5 consists of 15 chosen characteristics based on $\alpha \geq 4.90$ threshold value. For instance, Fig 6 displays the traits that were chosen forF_4.

In an IoT context, the purpose of this part is to assess the DL methods' performance in detecting botnet-based DDoS attacks. With almost similar method architectures, we were able to mimic and overserve the performance of the various classification methods. The performance indicators mentioned in the section before were calculated using a confusion matrix. Table 4 depicts the confusion matrix's general design. To locate the attack traffic, we evaluated the effectiveness of five alternative DL strategies. Five subgroups of the training set are created depending on the best attributes. For all of the techniques, we use almost identical structures (learning rate, optimizer, batch size, learning rate, the amount of neurons and covert layers they contain as well as the activation of both hidden and output layers). With the same structure and the same collection of characteristics, we saw that the techniques yielded varying outcomes. To choose the best technique without adding to the complexity of the procedures, the structural performance of the various approaches is evaluated.

Classification Performance

The likelihood of both attack and steady states being predicted has been effectively quantified using the assessment metrics described in the preceding section. Most of the DL techniques we looked at show output fluctuations that are linked to the number of features, suggesting that the prediction is best accurate when the number of features is optimum. For traffic categorization, this study employed five DL algorithms: RNN, CNN, MLP, LSTM, and DNN. We compared the predictions made by these algorithms, which have various categorization capacities. Table 4 displays the confusion matrices for each of the five techniques for the F_3 characteristics. The algorithms that are efficient in recognizing assaults with underreporting rates (FNR) of 0.78%, 0.40%, 0.59%, 0.87%, and 0.89%, respectively, are CNN, MLP, LSTM, RNN, and DNN. With a small variation, all algorithms can recognize both normal and attack data with an accuracy of roughly 99%. Since the attack traffic might put the IoT in peril, highly sensitive detection techniques are required.

Table 2. Feature Representation

Feature Set	Approaches	Computational Time (s)	Accuracy (%)	DR (%)
F_1	RNN	264	98	97.1
	CNN	202	98	97.5
	MLP	142	98.1	89
	DNN	202	98	97
	LSTM	323	98.4	98.3
	Proposed	102	99.5	99.5
F_2	RNN	256	98.2	99.1
	CNN	185	98.5	98.5
	MLP	117	98.1	98.4
	DNN	185	98.2	98.4
	LSTM	223	98.3	98.3
	Proposed	110	99.5	99.4
F_3	RNN	263	98.3	98.2
	CNN	181	98.3	98.6
	MLP	142	98.1	98.1
	DNN	154	98.2	98.1
	LSTM	211	98.2	98
	Proposed	112	99.5	99.6
F_4	RNN	278	98.2	98.02
	CNN	185	98.3	98.8
	MLP	119	98.2	98.5
	DNN	155	98.01	98.6
	LSTM	231	98.1	98.7
	Proposed	114	99.6	99.7
F_5	RNN	259	97.5	97
	CNN	202	97.8	97.5
	MLP	117	97.8	97.6
	DNN	202	97.3	97.8
	LSTM	203	97.5	98.2
	Proposed	115	99.6	99.8

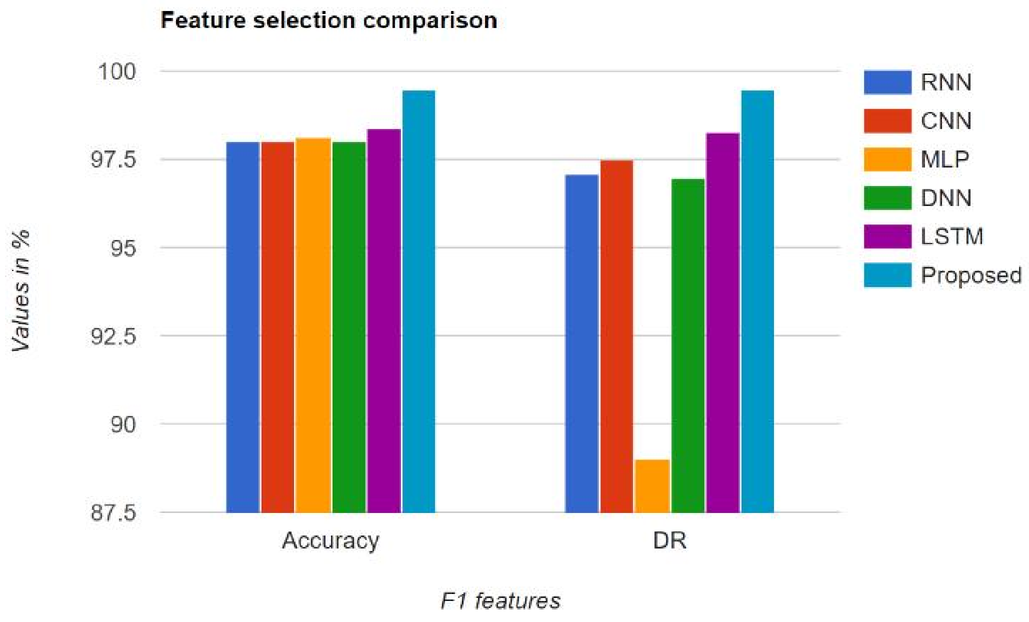

Figure 3. F1-Feature Subset Comparison

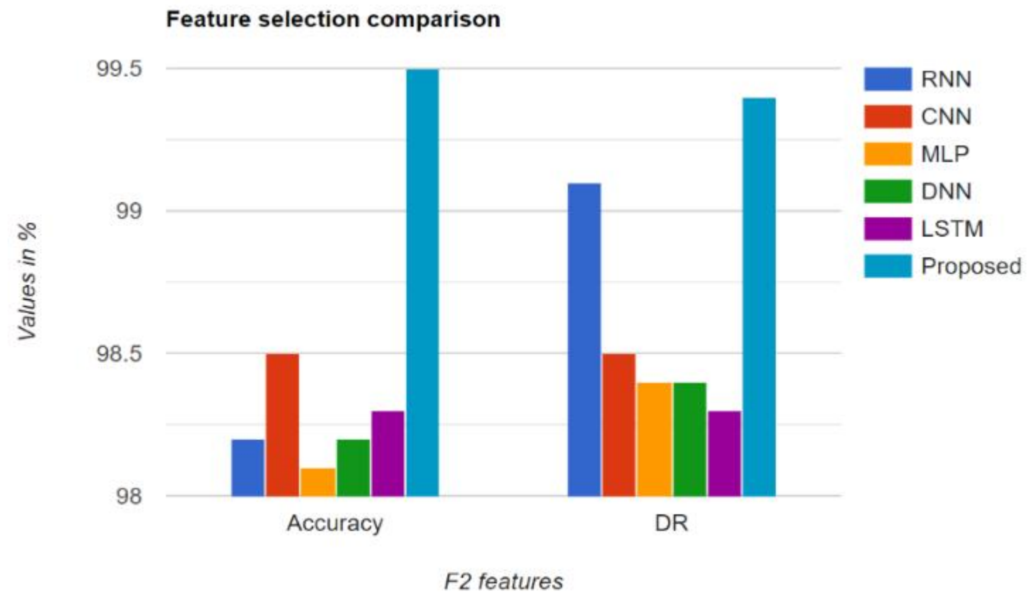

Figure 4. F2-Feature Subset Comparison

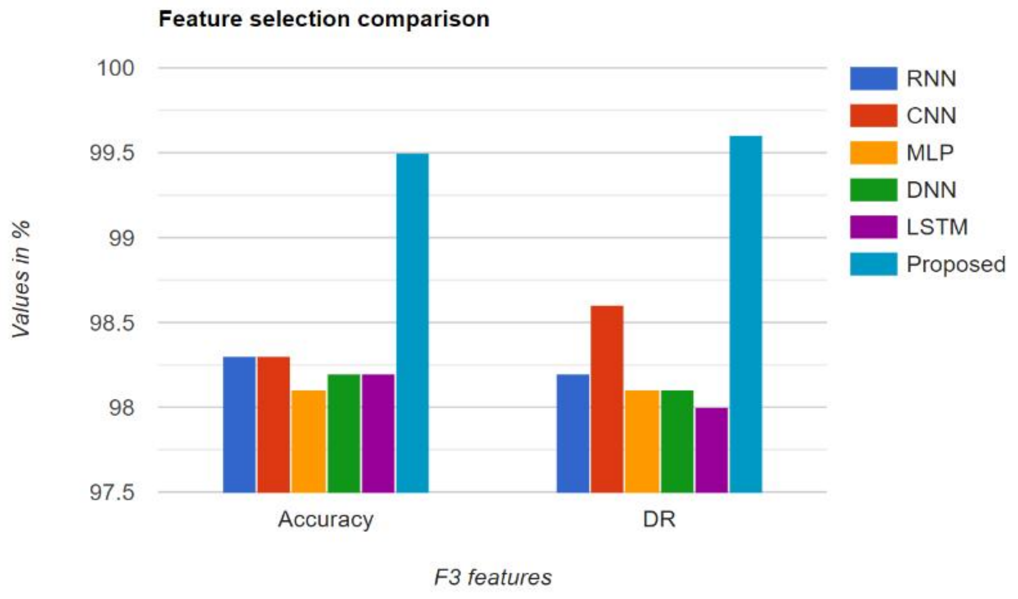

Figure 5. F3-Feature Subset Comparison

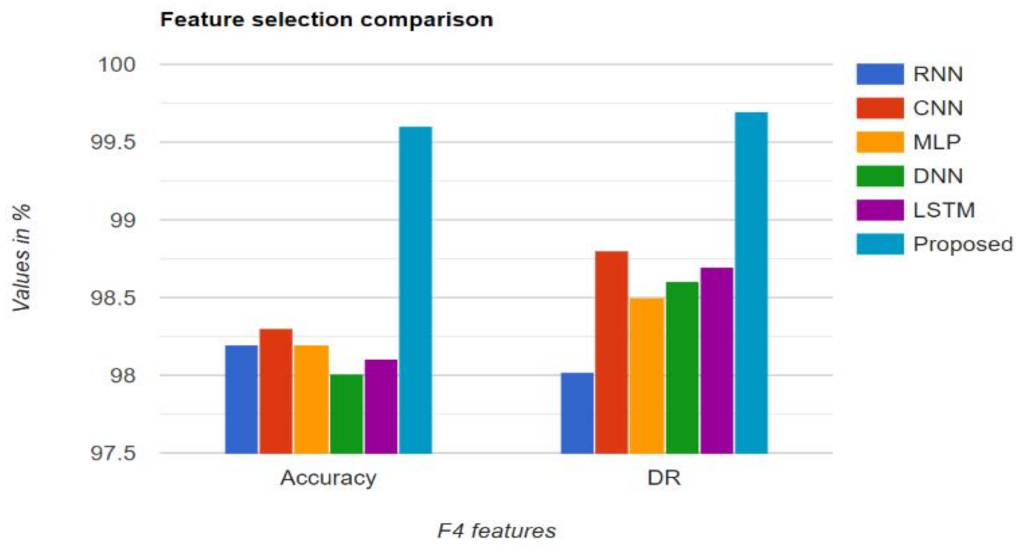

Figure 6. F4-Feature Subset Comparison

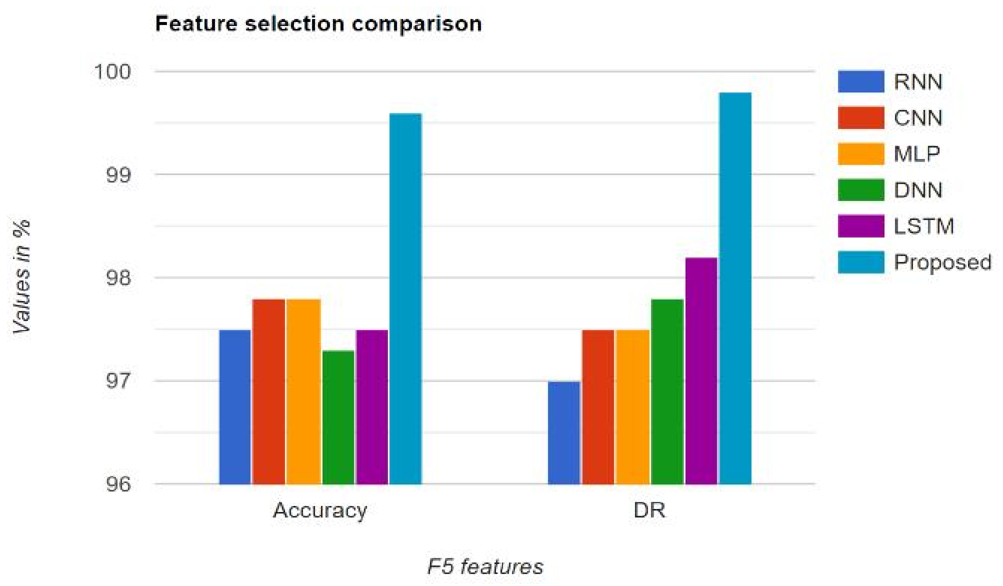

Figure 7. F5-Feature Subset Comparison

Then, as shown in Figure 5, for F_3, all technique validation accuracy and loss curves are compared. Compared to the other approaches, the accuracy curves for DNN and CNN are more stable and steeper as can be seen in Figure 6. This indicates that the DNN and CNN approach function well and are significantly better than the other algorithms.DNN, CNN, MLP, RNN, and LSTM each have validation accuracy of 99.30%, 99.37%, 99.33%, and 99.25%, respectively. The loss ratio of CNN and DNN is lower than that of the opposition. The loss curve's fluctuation is most consistent with CNN. It can be observed that the CNN approach for attack detection is more stable than other methods by comparing the accuracy and loss curves. This section compares the computing time accuracy, detection rate, and training duration of each technique. Table 3 presents the results of the performance comparison. About the broader trends, lowering the number of characteristics to a certain point enhances classification performance.

Table 3. Overall Performance Evaluation

Approaches	**Web attack dataset**		**UNSW-NB15 dataset**		**Alexa-based dataset**	
	Original dataset					
	DR (%)	FAR (%)	DR (%)	FAR (%)	DR (%)	FAR (%)
CART	94.5	6	90	10	92	11
k-NN	92.5	8	88	11.8	90	12
SVM	92.6	7.4	91	10	93	11
RF	95.4	5	93	7.5	95	8
GM	97	2.8	96	4	96.5	5
ConvNet	98.5	9.5	98	12	98.2	13
Proposed	99	13.5	99.5	13.8	99.5	14

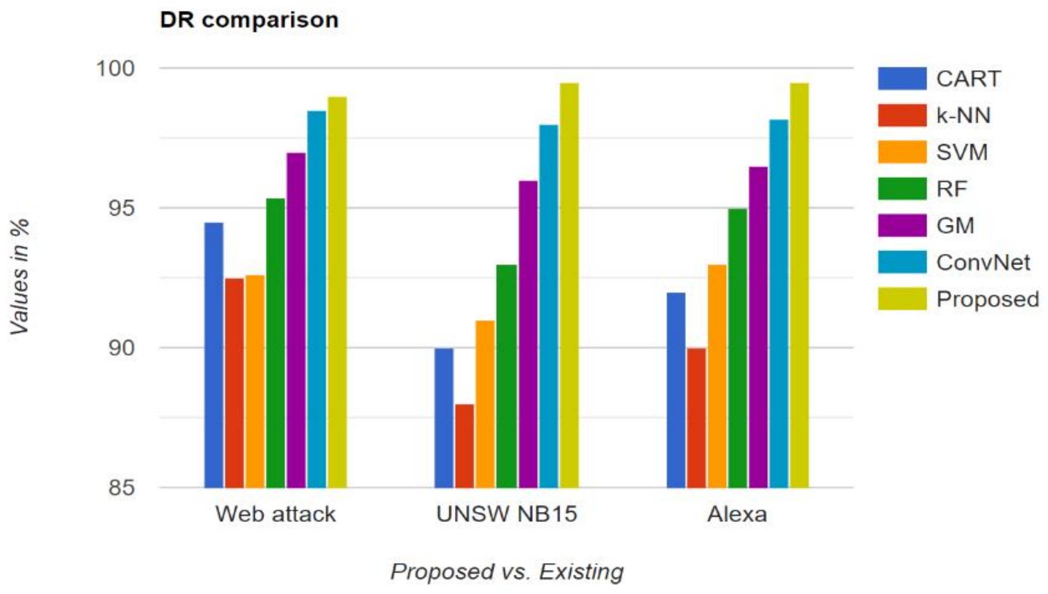

Figure 8. DR Comparison

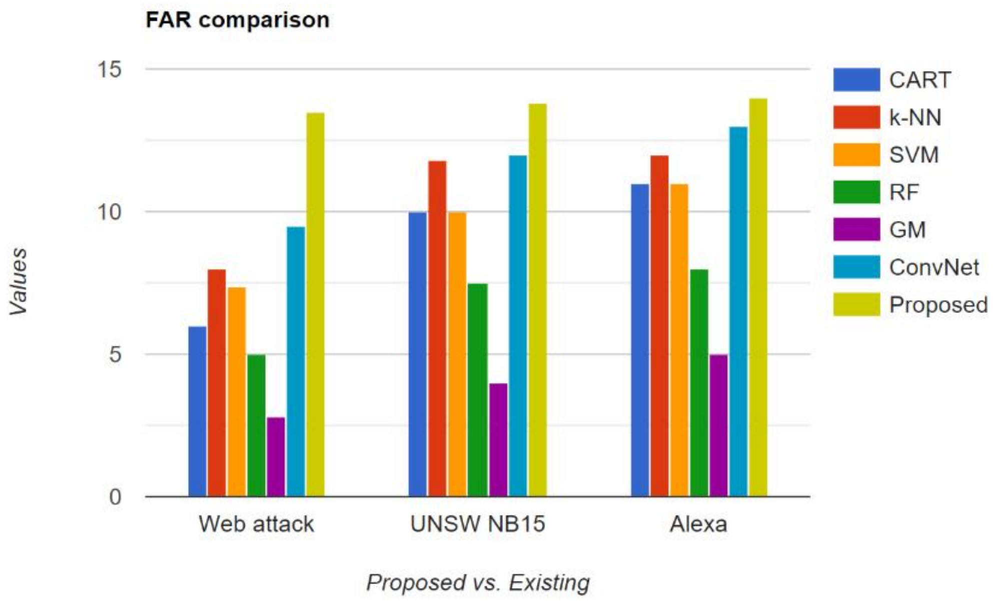

Figure 9. FAR Comparison

On F_5, LSTM has a minimum accuracy of 97.60%, although CNN and DNN frequently reach maximum accuracy on subset-2 of 99.43% and 99.53%, respectively. In F_3, the CNN reaches a maximum detection rate of 99.60%, whilst the MLP only manages a minimum detection rate of 89.99%. We evaluate the accuracy rate, detection rate, and computation time for subset 3 as an example. RNN, MLP, DNN, and LSTM are inferior to the other four classifiers in comparison CNN's accuracy is 0.12%, 0.04%, 0.07%, and 0.21% higher. In a similar vein, CNN has higher detection rates than other classifiers of 0.39, 0.19, 0.49, and 0.47 respectively. Additionally, CNN's computation takes 39.51 seconds longer than MLP's, 27.75 seconds longer than DNN's, 81.29 seconds less than RNN's, and 29.85 seconds less than LSTM's, but its accuracy is increased by roughly 0.04% to 0.21% and its detection rate is increased from 0.19% to 0.49%. The lengthy training period may be tolerated since classifiers are taught offline and aren't regularly updated, providing a high detection rate and accuracy. It implies that the detecting algorithms need to be very accurate and take little time to learn. In addition, we found that the CNN method outperformed the other two subsets (4 and 5) on F_3 with 30 characteristics. With F_4 characteristics, CNN is 99.29% accurate and 99.26% accurate when using F_5 characteristics. Even though the other subsets have fewer features than F_3, the training time for CNN which is F_3 is 3.98 seconds and 20.53 seconds faster than it would be with the other subsets and its accuracy is 0.08%, 0.11%, and the detection rate is 0. Better by 25% and 0.53%. Figure 3 to Figure 7 additionally displays the effectiveness of all classifiers utilizing all five subgroups in terms of additional metrics, including accuracy and DR. RNN obtains a maximum accuracy rate of 99.29% with all features set (Table 4).

Table 4. Confusion Matrix

Predicted Class			
		Class = yes	Class = No
Actual Class	Class = yes	TP	FN
	Class = no	FP	TN

A total score of 99.37% was given to CNN for all characteristics, with subsets 3 and 5 receiving minimum scores of 99.03% and 99.11% respectively. All features were used by MLP to obtain a maximum of 99.94%, and Subset-5 was utilized for a minimum of 98.98% of the data. DNN achieved a maximum accuracy of 99.75% using all characteristics and at least 99.16% accuracy using F_5. About all characteristics, LSTM obtained F_5 with a maximum of 99.37% and a minimum of 97.36%. RNN's highest F1 score with F_2 features is 99.22%, and with all features, it is 98.57%; the F1 score for CNN was 99.37% with F_2 and 98.61% with all features set, the F1 score for MLP was 99.34% with F_2 and 94.70% with all features, the F1 score for DNN was 99.48% with F_2 and 98.50% with all features, and the F1 score for LSTM was 99.09% with F_3 and 77.30% using F_5. As a result, the RNN's TPR with F_3 features may reach a maximum of 99.21% and at least 97.86%. With characteristics from F_3, CNN attained a maximum TPR of 99.60% and a minimum TPR of 97.87; MLP reached a maximum TPR of 99.45% and a minimum TPR of 89.99; DNN reached a maximum TPR of 99.49% and a minimum TPR of 97.29%; and LSTM reached a maximum TPR of 99.31% and a minimum TPR of 97.44% with F_5. The work may infer from the findings above that by utilizing F_3 characteristics, all classifiers yield accurate results.

In a particular case used in this article, the CNN approach outperforms other classifiers, producing more consistent and useful results. As a result, this result has certain benefits: Without having specific knowledge of traffic flow specifics, it is straightforward to gather training sets; using training sets with optimum features makes the training phase easier; and using training sets with optimal features reduces the complexity of the procedures and the need for resources. We choose the DL methods that have been tested on subset-3 characteristics to assess and confirm how well they perform on the actual test bed. For the actual test bed, the network architecture is the same as that in Figure 8. The flow statistics for forecasting regular or attack traffic flows in a genuine test-bed setting have been created/collected using the same methodology as the training data. Each taught DL technique is implemented separately in the controller. The technique then assigns either a "0" or a "1" to the incoming flow. (For instance, all methods in our test specify "0" for the ordinary flow and "1" for the attack flow because there are only two possibilities available). We utilized 50 consecutive judgments for each approach under two different network conditions (normal and attack) to test the overall performance in real-time traffic. Figure 7 and Figure 8 display the accurate detection rate for each technique in real-time traffic. We found that all approaches' output forecasts for normal flows are superior to those for attack flows. All techniques exceeded a 90% detection rate, particularly CNN, which predicts typical flows with a 99% accuracy rate. Similarly to that, the detection rate of techniques for attack flows is 87%, 97%, 85%, 93%, and 85%, respectively.

RNN takes longer to train than other techniques when comparing training times (for instance, in s) for F_3 characteristics of DL approaches. It took some time for CNN to become proficient. CNN also has a somewhat lower Detection time per flow and is faster (measured in microseconds (s)) than with other methods. A few flows per second can be handled by the LSTM for attack detection as evidenced by the fact that its detection time is substantially longer than that of other approaches. We may get to the conclusion using what we demonstrate shows, CNN is the most effective technique for identifying botnet-based DDoS attacks in an SDN setting, according to assessment parameters that include detection rate, training, and detection times. Fig 8 respectively provides graphic representations of detection and training periods.

Discussion

In this work, we examine and implement DL methods to assist in locating botnet-based DDoS assaults in an environment with assistance. As we assess the efficacy of the DL techniques, to locate the assault, we employ the best features and the baseline hyper-parameters. Using a variety of evaluation criteria (including precision, detection rate, training, detection time, etc.), we analyze the effectiveness of the DL approaches. The complete training dataset is created in an SDN, as its collection setting rather than depending on out-of-date or traditional datasets, is a critical component of this work. Almost all studies employed conventional datasets, which are unsuitable for SDN owing to imbalance problems as well as a flow-based nature. Additionally, rather than relying on a dataset that has a lot of features, we partition the entire dataset into subsets (such as ideal features) according to the relevance of each feature. These subsets have then been evaluated to see how optimal features affect the performance of the approach. The effectiveness of each DL technique on the same group of features varies according to simulation results; consequently, it follows that the best characteristics may raise the rate of detection. We come to the conclusion that the suggested research should use CNN as the optimum method and the chosen scenario based on the trial results and the preceding justification. With characteristics and the created dataset, its accuracy is 99.37%. The detection rate of CNN during actual test-bed traffic is 97% for attack flows and 100% for normal flows. We also took temporal measures (such as training and detection timings) into account and found that CNN's training and detection times were appropriate. As a result, CNN displays a respectable detection efficiency or precision when identifying botnet-based attacks in the present in an SDN setting. Another important advantage of the security technique utilized in this study is that it defends the IoT from DDoS attacks that use botnets.

This study has the drawback of only applying botnet-based flooding DDoS attacks in IoT systems. Attacks that aren't volumetric, such as malicious or low-rate DDoS attacks, cannot be reliably detected by it. Furthermore, we concentrated on an environment with a single IoT device. This work will be expanded upon to examine low-rate, spoof DDoS, and other malicious attacks utilizing hybrid DL methods on actual traffic. It's also a good idea to train in real-time ways for updating IDS systems.

CONCLUSION

The resilience of deep models against attacks in industrial contexts was measured using a novel technique explored in this study. The potential element of certain samples may revert to true normality after being subjected to adversarial attacks and therefore not need to be considered when computing robustness. The process consists of four steps: dataset acquisition, feature selection, extraction and prediction. To be exact, the technique distinguishes between really adversarial and non-adversarial data using a set of models referred to as support models and robustness is calculated exclusively for truly adversarial samples. Additionally, this work used the proposed technique in an actual industrial setting. We assessed the adaptability of the models in this context. The testing results demonstrated that the model outperformed the existing approaches more consistently in this specific situation. As a consequence, the only needs to disrupt 60.1% of the original data compared to the other standard CNN which has to perturb over 110% of the original samples. As part of our ongoing research, we want to use the dataset to assess the resilience of prediction systems in other industrial contexts. This work also intends to research the characteristics of several factors that will serve as support models. In addition, the work also intends to research how robustness, adversarial samples, and interpretability techniques interact. The robustness of the prediction model may be strengthened by employing interpretability approaches to identify adversarial samples as one application of this work.

ETHICAL DECLARATION

Conflict of interest: No declaration required. **Financing:** No reporting required. **Peer review:** Double anonymous peer review.

REFERENCES

[1] L. Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang, and Y. Deng, “A new framework for DDoS attack detection and defense in SDN environment,” *IEEE Access*, vol. 8, pp. 161908-161919, 2020.

[2] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, “A DDoS attack detection method based on SVM in software defined network,” *Security and Communication Networks*, vol. 2018, 2018.

[3] J. A. Perez-Diaz, I. A. Valdovinos, K. K. R. Choo, and D. Zhu, “A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning,” *IEEE Access*, vol. 8, pp. 155859-155872, 2020.

[4] O. Habibi, M. Chemmakha, and M. Lazaar, “Imbalanced tabular data modelization using CTGAN and machine learning to improve IoT Botnet attacks detection,” *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105669, 2023.

[5] H. S. Ilango, M. Ma, and R. Su, “A feedforward–convolutional neural network to detect low-rate dos in IoT, *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105059, 2022.

[6] K. N. Rao, K. V. Rao, and P. R. PVGD, “A hybrid intrusion detection system based on sparse autoencoder and deep neural network,” *Computer Communications*, vol. 180, pp. 77-88, 2021.

[7] H. T. Nguyen, Q. D. Ngo, D. H. Nguyen, and V. H. Le, “PSI-rooted subgraph: A novel feature for IoT botnet detection using classifier algorithms,” *ICT Express*, vol. 6, no. 2, pp. 128-138, 2020.

[8] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, “N-baiot—network-based detection of IoT botnet attacks using deep autoencoders,” *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12-22, 2018.

[9] A. Al Shorman, H. Faris, and I. Aljarah, “Unsupervised intelligent system based on one class support vector machine and grey wolf optimization for IoT botnet detection,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 7, pp. 2809-2825, 2020.

[10] T. Saba, A. Rehman, T. Sadad, H. Kolivand, and S. A. Bahaj, “Anomaly-based intrusion detection system for IoT networks through deep learning model,” *Computers and Electrical Engineering*, vol. 99, p. 107810, 2022.

[11] M. S. Elsayed, N. A. Le-Khac, S. Dev, and A. D. Jurcut, “Ddosnet: A deep-learning model for detecting network attacks,” in *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Aug. 2020, pp. 391-396.

[12] K. S. Sahoo, B. K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy, M. Khari, and D. Burgos, “An evolutionary SVM model for DDOS attack detection in software defined networks,” *IEEE Access*, vol. 8, pp. 132502-132513, 2020.

[13] A. A. Ahmed, W. A. Jabbar, A. S. Sadiq, and H. Patel, “Deep learning-based classification model for botnet attack detection,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 7, pp. 3457-3466, 2022.

[14] G. D. L. T. Parra, P. Rad, K. K. R. Choo, and N. Beebe, “Detecting Internet of Things attacks using distributed deep learning,” *Journal of Network and Computer Applications*, vol. 163, p. 102662, 2020.

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

[16] S. Gamage and J. Samarabandu, “Deep learning methods in network intrusion detection: A survey and an objective comparison,” *Journal of Network and Computer Applications*, vol. 169, p. 102767, 2020.

[17] C. Robberts and J. Toft, “Finding vulnerabilities in IoT devices: Ethical hacking of electronic locks,” *School of Electrical Engineering and Computer Science (EECS)*, 2019.

[18] A. Aldweesh, A. Derhab, and A. Z. Emam, “Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues,” *Knowledge-Based Systems*, vol. 189, p. 105124, 2020.

[19] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,” *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[20] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, “Deep learning-based intrusion detection for IoT networks,” in *2019 IEEE 24th pacific rim international symposium on dependable computing (PRDC)*, Dec. 2019, pp. 256-25609.

[21] M. M. Hassan, A. Gumaei, A. Alsanad, M. Alrubaian, and G. Fortino, “A hybrid deep learning model for efficient intrusion detection in big data environment,” *Information Sciences*, vol. 513, pp. 386-396, 2020.

[22] D. Li, L. Deng, M. Lee, and H. Wang, “IoT data feature extraction and intrusion detection system for smart cities based on deep migration learning,” *International journal of information management*, vol. 49, pp. 533-545, 2019.

[23] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, “Anomaly detection based on convolutional recurrent autoencoder for IoT time series,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 1, pp. 112-122.

[24]B. Wang, Y. Su, M. Zhang, and J. Nie, "A deep hierarchical network for packet-level malicious traffic detection," *IEEE Access*, vol. 8, pp. 201728-201740, 2020.

[25]H. Yang, and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," *IEEE Access*, vol. 7, pp. 64366-64374, 2019.

[26]Y. Otoum, D. Liu, and A. Nayak, "DL-IDS: A deep learning–based intrusion detection framework for securing IoT," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 3, p. e3803, 2022.

[27]Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, and L. Cui, "Robust detection for network intrusion of industrial IoT based on multi-CNN fusion," *Measurement*, vol. 154, p. 107450, 2020.

[28]Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization," *Journal of Information Security and Applications*, vol. 58, p. 102804, 2021.

[29]P. Sherubha, S. P. Sasirekha, V. Manikandan, K. Gowsic, and N. Mohanasundaram, "Graph based event measurement for analyzing distributed anomalies in sensor networks," *Sādhanā*, vol. 45, pp. 1-5, 2020.

[30]P. Sherubha and N. Mohanasundaram, "An efficient network threat detection and classification method using ANP-MVPS algorithm in wireless sensor networks," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 11, pp. 1597-1606, 2019.

[31]P. Sherubha and N. Mohanasundaram, "An efficient intrusion detection and authentication mechanism for detecting clone attack in wireless sensor networks," *J Adv Res Dyn Control Syst*, vol. 11, no. 5, pp. 55-68, 2019.

[32]S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," *Journal of Network and Computer Applications*, vol. 169, p. 102767, 2020.

[33]C. A. Rieth, B. D. Amsel, R. Tran, and M. B. Cook, "Issues and advances in anomaly detection evaluation for joint human-automated systems," in *Advances in Human Factors in Robots and Unmanned Systems: Proceedings of the AHFE 2017 International Conference on Human Factors in Robots and Unmanned Systems, July 17– 21, 2017, The Westin Bonaventure Hotel, Los Angeles, California, USA 8*, 2018, pp. 52-63.

[34]J. Men, Z. Lv, X. Zhou, Z. Han, H. Xian, and Y. N. Song, "Machine learning methods for industrial protocol security analysis: Issues, taxonomy, and directions," *IEEE Access*, vol. 8, pp. 83842-83857, 2020.

[35]T. W. Weng, H. Zhang, P. Y. Chen, J. Yi, D. Su, Y. Gao, C. J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," 2018, doi: https://doi.org/10.48550/arXiv.1801.10578.