Hindawi
Security and Communication Networks
Volume 2021, Article ID 5710028, 14 pages
https://doi.org/10.1155/2021/5710028

Research Article

Towards Effective Detection of Recent DDoS Attacks: A Deep Learning Approach

Ivandro Ortet Lopes,[1,2,3,4] Deqing Zou,[2,4,5,6] Francis A Ruambo ,[1,2,3,4] Saeed Akbar,[1] and Bin Yuan [2,4,5,6,7]

[1]*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*National Engineering Research Center for Big Data Technology and System, Huazhong University of Science and Technology, Wuhan 430074, China*
[3]*Cluster and Grid Computing Lab, Huazhong University of Science and Technology, Wuhan 430074, China*
[4]*Services Computing Technology and System Lab, Huazhong University of Science and Technology, Wuhan 430074, China*
[5]*Big Data Security Engineering Research Center, Huazhong University of Science and Technology, Wuhan 430074, China*
[6]*School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China*
[7]*Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China*

Correspondence should be addressed to Bin Yuan; yuanbin@hust.edu.cn

Received 28 May 2021; Revised 21 October 2021; Accepted 25 October 2021; Published 30 November 2021

Academic Editor: Wenjuan Li

Distributed Denial of Service (DDoS) is a predominant threat to the availability of online services due to their size and frequency. However, developing an effective security mechanism to protect a network from this threat is a big challenge because DDoS uses various attack approaches coupled with several possible combinations. Furthermore, most of the existing deep learning- (DL-) based models pose a high processing overhead or may not perform well to detect the recently reported DDoS attacks as these models use outdated datasets for training and evaluation. To address the issues mentioned earlier, we propose CyDDoS, an integrated intrusion detection system (IDS) framework, which combines an ensemble of feature engineering algorithms with the deep neural network. The ensemble feature selection is based on five machine learning classifiers used to identify and extract the most relevant features used by the predictive model. This approach improves the model performance by processing only a subset of relevant features while reducing the computation requirement. We evaluate the model performance based on CICDDoS2019, a modern and realistic dataset consisting of normal and DDoS attack traffic. The evaluation considers different validation metrics such as accuracy, precision, F1-Score, and recall to argue the effectiveness of the proposed framework against state-of-the-art IDSs.

1. Introduction

Recent advancements in communication technologies such as long-term evolution, smart-grids, 5G, and the Internet of Things (IoT) have caused a tremendous increase in data transmission because of many communicating entities. The number of devices might touch 29.3 billion by 2023, three times the current global population [1]. Among these devices, the IoT devices appear to be the most helpful amplification platforms to launch cyberattacks because of their large number and lack of security best practices [2, 3]. The DDoS attack is a primary threat currently observed by most service providers worldwide. In DDoS attacks, a malicious actor first explores some vulnerable systems over the Internet to control and use them to generate massive traffic. Then, it floods the target system with the generated traffic, thereby disrupting the normal operations of the target system. The first most severe DDoS attack was recorded in September 2016, when massive IP traffic having a volume of 620 Gbps originated from hundreds of thousands of IoT devices. An even greater DDoS attack at 1.1 Tbps was reported the same year. The DDoS attacks have grown severely over the past few years and have reached 2.3 Tbps in 2020 [4], resulting in extreme downtime of online services.

The massive data generated by the current network requires a communication channel to handle and transmit it efficiently, securely, and reliable. Therefore, as cybersecurity events become more sophisticated and frequent, an IDS must detect and tackle those anomalies as quickly as possible. To do so, researchers have devised various intrusion detection techniques in the literature. For instance, researchers propose solutions such as a signature-based intrusion detection system (IDS) to discriminate the DDoS traffic from a normal one. However, these schemes are unable to classify new attacks.

Motivated by the challenges described above, the research community has paid particular attention to applying state-of-the-art machine learning (ML) models for detecting DDoS traffic. ML provides various advanced tools and techniques to predict cyberattacks using a quick and automated approach. Nevertheless, one of the primary limitations of classical ML techniques is the amount of data they can handle. In contrast, deep learning (DL) can handle and learn patterns from a vast amount of data. IDSs based on DL have been proven to be very efficient and effective in identifying anomalies within IP network traffic. However, existing DL-based IDSs have been trained and evaluated using out-of-date datasets such as the KDD99 or NSL-KDD dataset, publicly available from 1999 to 2009 [1]. These datasets do not reflect the recently reported DDoS attacks in the current network environment [5–7]. Additionally, the existing architectures are more compute-intensive, which can be improved significantly by optimizing the feature selection to reduce the number of features for traffic prediction, enhancing the detection performance.

Building on this observation, we propose CyDDoS, an integrated IDS framework, which combines an ensemble of feature engineering algorithms with a deep neural network to effectively detect the most recent DDoS attacks while incurring the least processing overhead. Specifically, it uses an ensemble of feature selection based on a supervised feature ranking technique to enable dataset dimensionality reduction and lower computation complexity. The significant contributions of this paper are as follows:

(1) We proposed a lightweight IDS framework, combining the ensemble of feature selection algorithm with DNN classifier, to improve the intrusion detection performance while reducing the system processing overhead. The model is trained in 5 seconds, reducing 88% of features from the original dataset while providing a detection performance superior to 99.6% in most considered metrics.

(2) To reduce the processing overhead, several ensemble groups are evaluated, using different numbers of classifiers, to determine the feature list that can produce the best prediction performance.

(3) In the context of feature ensemble, we employed two approaches that combine decisions from multiple ranking algorithms into one. The decision is made based on a threshold that defines the minimum number of classifiers that must agree on a feature or based on the top ten feature weights.

(4) The trial-and-error approach is used to optimize the DNN hyperparameter and improve the detection performance.

(5) We argue the efficiency of our proposed DNN model against existing DL techniques using the CiCDDoS2019 dataset. The dataset covers the most recent DDoS attacks reported.

The remainder of the paper is organized as follows: we first highlight existing IDSs literature, followed by the description of our proposed system architecture. Then, we present the results obtained, and, finally, the last section draws the conclusion.

2. Related Work

The research community has presented several detections and mitigation solutions for DoS attacks since their first occurrence in 1999 [8]. In this section, we briefly describe the existing strategies that use ML for anomaly detection. A large number of studies propose network anomaly detection schemes based on the classical ML models. However, Doriguzzi-Corin et al. [9] state that adopting classical ML-based techniques in the "real world" is limited due to network intrusion challenges such as traffic variability and high error costs. Therefore, recent works have shifted their focus towards studying the effectiveness of DL models in IDSs because they can handle large-scale data and learn more complex patterns within that data.

For the reasons mentioned above, we focus only on DL-based DDoS detection systems. Recent studies such as [10, 11] and [12] propose a network intrusion detection solution based on DL. These studies use the KDD99 (1999) or NSL-KDD (2009) dataset to train and validate their proposed models. Yuan et al. [13] presented DeepDefense, an anomaly detection method based on DNN. The model can learn the network's traffic sequence to trace attacks related activity. DeepDefense leverages Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), which has proved to provide huge performance improvement in terms of accuracy when trained with a large dataset. The authors used ISCX2012 dataset to evaluate the model's performance, and the results demonstrate a considerable reduction in terms of error rate compared to the classic ML method. Since the datasets used in the said studies are outdated, they do not represent the features and characteristics of recently reported DDoS attacks [5, 6]. Therefore, the previously mentioned intrusion detection schemes cannot be reliably used to detect new threats in networks such as IoT.

Elsayed et al. [14] developed an IDS for the Software-Defined Networking (SDN) environment. The authors combined RNN with autoencoder (AE) to improve the DDoS detection accuracy. It uses the CICDDoS2019 dataset for training and validation. The authors proposed a pre-training based on an unsupervised approach and the fine-tuning stage using supervised learning. The AE is an unsupervised model which produces a compressed representation of the input samples to reduce the processing overhead. The proposed scheme achieves a substantial improvement in terms of accuracy compared to existing systems. However, the research did not discuss the performance details such as model training time or samples classified for a given amount of time. Similarly, "Kitsune" [15] uses an unsupervised approach based on AE for intrusion detection. Kitsune is a lightweight real-time detection model that uses an ensemble of autoencoders to collectively predict normal and abnormal traffic patterns. The experiments conducted using a Raspberry PI system showed that this solution can achieve performance similar to offline anomaly detection solutions. A lightweight approach based on MultiModal Deep AutoEncoder (M2-DAE) and a soft-output classifier are proposed [16]. This framework is optimized for the IoT environment and can identify both known and unknown attacks.

Doshi et al. [17] developed a DL-based design for DDoS detection in an IoT setting. The authors used network-specific IoT behavioral features, such as reduced terminals and constant arrival time between packets, for feature selection to improve detection accuracy. Similarly, Ge et al. [18] deployed an intrusion detection system for the IoT environment based on a customized deep learning method. This proposal adopted the feedforward neural network (FNN) technique with an embedding layer to perform multiclass prediction of attacks. Furthermore, they developed a second feedforward neural network model that implemented the transfer learning technique to encode a high-dimensional categorical feature and perform a binary classifier. Both classifiers achieved good detection performance.

Doriguzzi-Corin et al. [9] proposed LUCID, a lightweight online DL-based IDS. It uses CNN for making a binary classification on three benchmark datasets. Although the datasets allow for offline evaluation, they use a network traffic processing technique that generates a spatial representation of data to support online attack detection. According to the authors, it can reduce the processing time by 40X compared to its counterparts when deployed in a resource-constrained environment while matching the high accuracy of recent research. Nevertheless, this solution requires a high retraining time to update the model's weights and biases using new traffic samples, exposing the network to new DDoS attacks during this period.

Wang et al. [19] proposed an intrusion detection framework to address network and host intrusions. Network intrusion detection leverages the Stacked Denoising AutoEncoder-Extreme Learning Machine (SDAE-ELM) to address the slow training time and poor detection performance of existing neural network approaches. The host intrusion detection model is based on DBN-Softmax, aimed to improve the detection effectiveness of host intrusion. Small-batch gradient descent is used during the network training and optimization to improve both models' training efficiency and classification performance. Experiments were conducted based on several datasets to validate the higher performance of this proposal against other classic machine learning approaches. SDAE is a powerful deep learning technique to compress datasets. However, most of the datasets used in this study are outdated, and the average detection performance for binary classification still needs further improvements.

The dataset used to evaluate an IDS model is critical for determining the model's reliability in detecting potential DDoS attacks. Most of the existing DL-based strategies are unable to detect recent attacks reported by the research community due to the limitations of datasets being used for training and evaluation [5, 6]. On the contrary, some recent studies do consider datasets containing data related to recent DDoS attacks. However, they pose a high processing overhead for training the models and making predictions. The computational complexity of the detection system can be significantly improved if the number of features being used for training and detection can be reduced without affecting the performance of the IDS. To solve the said issues, we propose a DL-based IDS, which attains better DDoS attack detection accuracy and requires less computing capacity compared to its counterparts.

3. Proposed Methodology

The detection approach employs a discriminative deep neural network mechanism that makes the binary classification of the traffic sample. Figure 1 illustrates the general solution block diagram, consisting of four steps: preprocessing, feature engineering, building model and testing accuracy, and classification.

3.1. Data Preprocessing. Preprocessing is the first step that takes the IP traffic data as input and outputs cleaned data. Data preprocessing is considered to be a critical task in machine learning as it may significantly improve the efficiency and effectiveness of the training process. It includes some essential tasks such as removal of irrelevant data, handling missing values, label conversion, categorification, and data normalization.

3.1.1. Removing Irrelevant Features. The preprocessing step starts by removing irrelevant features. Features such as "Unnamed: 0," "Flow ID," and "Inbound" do not add any relevant information to the problem at hand. Features such as "Timestamp" IP address and port number information are related to IP socket or feature ID, which may cause the model to overfit. Hence, we keep only relevant features and remove the irrelevant ones manually. After identifying relevant features, the next task in the preprocessing step is to handle the missing values.

3.1.2. Handling Missing Records. There are several strategies to tackle the missing values. The proposed approach replaces the missing values by the median of the corresponding feature because it provides a better representation of the majority value within the feature. Additionally, the median provides a robust estimation when the data is skewed or

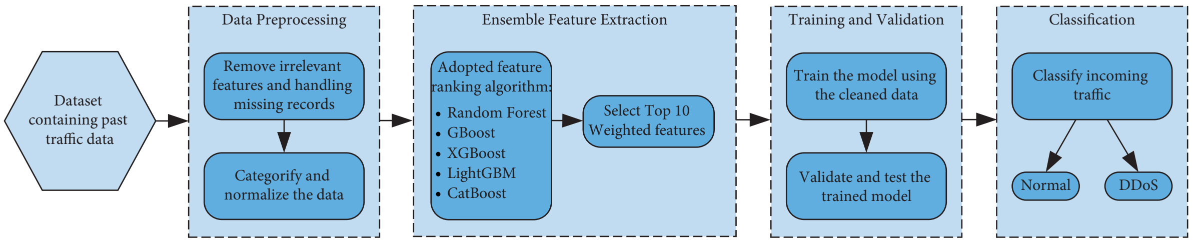

Dataset containing past traffic data
Data Preprocessing
Remove irrelevant features and handling missing records
Categorify and normalize the data
Ensemble Feature Extraction
Adopted feature ranking algorithm:
• Random Forest
• GBoost
• XGBoost
• LightGBM
• CatBoost
Select Top 10 Weighted features
Training and Validation
Train the model using the cleaned data
Validate and test the trained model
Classification
Classify incoming traffic
Normal
DDoS

FIGURE 1: CyDDoS processing flow chart.

contains outlier. Equations (1) and (2) depict the formula for calculating the median of a series of values:

$$\left(\frac{n+1}{2}\right)^{th}, \text{ for odd } n. \tag{1}$$

$$\left(\frac{(n/2)+((n+2)/2)}{2}\right)^{th}, \text{for even } n, \tag{2}$$

where n denotes the number of samples on the dataset.

3.1.3. Label Conversion. This paper considers binary classification of network traffic as normal traffic or DDoS attack. Mathematically, this function can be denoted as

$$f(y)=\begin{cases}\text{Normal}, & for\ y=\text{Normal}\\ \text{DDoS}, & for\ y\neq\text{Normal}\end{cases}, \tag{3}$$

where y denotes the data's label.

3.1.4. Categorify. Columns that have discrete string values are converted to integer category index. Category embedding is used to process the discrete variable. Categorical variable embeddings were proposed recently in [20] to improve the processing of discrete variables in neural networks (NN). Compared to one-hot encoding, its benefits include reduced memory requirements and higher processing speed. Entity embedding also allows a neural network model to better generalize for sparse data and unknown statistics. Our model processes every numerical column with cardinality (amount of distinct level or unique value) below nine thousand as categorical variables.

3.1.5. Normalization. Continuous values on the dataset have an enormous difference range, leading to higher prediction errors. Dataset normalization helps reduce classification errors significantly and allows the model to converge faster. We used the standardization method (z-score) that scales the features to have a mean of 0 and a standard deviation of 1. This approach enables the model to be less sensitive to outliers. The standardization is given by

$$\sigma=\sqrt{\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{N-1}}, \tag{4}$$

where σ denotes standard deviation, x_i denotes the feature value, N is the number of training samples, and μ represents the mean and is given by

$$\mu=\frac{1}{N}\sum_{i=1}^{N}x_i. \tag{5}$$

3.2. Feature Engineering. The CICDDoS2019 dataset is composed of 88 features extracted from IP packets captured using the CICFlowmeter tool. However, not all features can equally contribute to DDoS attack detection. Therefore, feature extraction is an essential step in ML as it reduces dataset dimensionality issues. In the context of this paper, we employ the ensemble technique using a different number of feature selection techniques to extract features that contribute the most to identifying DDoS attacks. The ensemble concept has been used for several years in the machine learning field, which combines the results of several models to outperform the model's prediction performance based on a single machine learning technique. Ensemble of feature selection integrates several feature rankings algorithms to produce the final list of features used by the predictive model.

The feature ranking techniques adopted are decision tree (DT), random forest (RF), Gradient Boost (GBoost), Extreme Gradient Boosting (XGBoost), Light Gradient Boosted Machine (LightGBM), and CatBoost. The DT creates a simple representation of the dataset using the tree structure, where the branch represents the observed data, and nodes or leaves represent the class label. RF implements an ensemble of multiple DT. Typically, it uses the majority voting approach to compute the classification results as a combination of the results of an individual tree. GBoost algorithm creates ensemble over boosted DT through minimization of error gradient. Each predictor is fitted on the residual error introduced by the previous predictor to improve the overall model performance. XGBoost, LightGBM, and CatBoost are variant implementations of GBoost aimed to tackle some inefficiency of the original algorithm.

To determine the optimal number of feature ranking algorithms that allow our deep learning model to achieve the best classification results, we created four feature ensemble groups, with each one consisting of three to six feature ranking algorithms. We refer to each group as "ensemble group x," where x has the value of 1, 2, 3, or 4, as illustrated in Table 1.

Different approaches can be employed within an ensemble of feature raking to create the final list of features from the lists of features proposed by each ranking algorithm. To select a minimal restricted list, we first applied the

Table 1: Ranking algorithms in each ensemble group.

Ensemble group	Threshold	Feature ranking
Ensemble group 1	4	Decision tree Random forest GBoost XGBoost LightGBM CatBoost
Ensemble group 2	4	Random forest GBoost XGBoost LightGBM CatBoost
Ensemble group 3	3	GBoost XGBoost LightGBM CatBoost
Ensemble group 4	2	XGBoost LightGBM CatBoost

recursive feature elimination (RFE) on each ranking algorithm to find the minimal number of features that enable them to achieve the best predictive performance. RFE is a pragmatic approach that selects features based on their impact on the model performance. It removes features one at a time and computes the impact of the remaining feature until it identifies the optimal minimum feature list. This approach generates one list of features per ranking algorithm within an ensemble group. The final feature list per ensemble group was selected using a threshold, determining how many feature rankers must agree on a specific feature. Considering that the ensemble group consists of three to six ranking algorithms, we defined a nonlinear threshold, calculated by subtracting n algorithm from the list, where n is calculated as the integer division of the total ranking algorithm within an ensemble group by three, according to Table 1. We refer to the list of features generated based on the number of ranking algorithms that agreed on it as an "ensemble list."

The list of features produced by each ensemble group is used to train the DL model. However, according to Table 2, ensemble group 1 and ensemble group 2 generated a list consisting of only five features, which resulted in a poor model classification performance. The model trained based on these feature sets achieved a low classification of 94% in most considered metrics. The reduced list of features was generated because each feature ranking algorithm within the ensemble group follows its own strategy to select or rank features from a dataset, and it is not easy to have several different algorithms to agree on the same features set. This issue occurred on the ensemble group having five and six ranking algorithms.

A scenario might have occurred in the experiments described above, where a feature obtaining high evaluation weight by multiple models gets penalized because the number of models that agree upon it is below the defined threshold. We adopted the "Top 10 Weight" strategy to select additional features according to their total average weight within the ensemble group to address this issue. Feature weight value indicates how much a model believes it contributes to the overall prediction performance. For feature ranking techniques such as LightGBM and CatBoost, which evaluate the contribution of features based on criteria other than the weight, we converted this value to the weight by dividing it by the sum of all values assigned to each model. The average feature weight within the ensemble group is used to sort the feature list before selecting the top highest raked features. Table 2 illustrates the feature produced by each of the two strategies adopted. The feature order does not have any significance.

Each ensemble group generates two lists of features. These lists are used to train three DL models. The first and second models are trained based on the features illustrated in the table above. A third model is trained using a feature list created by merging the two lists above.

3.3. Deep Neural Network Architecture.

The proposed DL-based intrusion detection model comprises an input, output, and two hidden layers, as depicted in Figure 2. The input layer has 296 neurons. The first and second hidden layers consist of 200 and 100 neurons, respectively. The network computes the weighted sum from neurons on the input layer, adds bias, and stores the results at each neuron on the first hidden layer. CyDDoS adopts the Rectified Linear Units (ReLu) activation function as it performs better than other activation functions for the said problem. The ReLu function lies between the linear layers to turn all negative numbers into zero and is given by

$$F(x) = \max(0, x), \tag{6}$$

where $F(x) = 0$, if $(x < 0)$, $F(x) = x$, if $(x \geq 0)$, and x denotes the activation value.

The model then calculates the weighted sum plus bias from 200 size vectors in the first hidden layer and stores it on each neuron in the second hidden layer. Finally, the 100 size vector arrives at the output layer, containing two units of fully connected SoftMax, representing the probability of the traffic sample belonging to the DDoS or normal category. The output layer makes a prediction based on Softmax, which is given by [21]

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for i} = 1, \ldots, \text{K}, \tag{7}$$

where z is the sample to be predicted and K is the number of classes.

CyDDoS uses the Adaptive Moment Estimator (ADAM) as an optimizer. A loss function measures the difference between the actual value and the foreseen value. The model's accuracy is higher if the loss is lower and vice versa. CyDDoS utilizes the cross-entropy loss function to compare the predicted values with actual values and adjust the hyper-parameters based on those values. The loss function is given by [22, 23]

Table 2: Feature list selected per ensemble group and strategy.

Ensemble group	Ensemble list	Top 10 Weight
Ensemble group 1	Protocol	Min Packet Length
	URG Flag Count	Init_Win_bytes_forward
	Flow Duration	ACK Flag Count
	Flow IAT Mean	Fwd Packet Length Min
	Flow IAT Std	Average Packet Size
		Fwd Packet Length Max
		Flow IAT Max
		Flow Packets/s
		Init_Win_bytes_backward
		Flow IAT Min
Ensemble group 2	Flow Packets/s	Min Packet Length
	Fwd Header Length	Init_Win_bytes_forward
	Fwd Packets/s	Fwd Packet Length Min
	Min Packet Length	ACK Flag Count
	Average Packet Size	Fwd Packet Length Mean
		Avg Fwd Segment Size
		Total Backward Packets
		Total Length of Fwd Packets
		Flow Packets/s
		Flow IAT Min
Ensemble group 3	Protocol	Min Packet Length
	Total Fwd Packets	Init_Win_bytes_forward
	Fwd PSH Flags	ACK Flag Count
	Bwd URG Flags	Bwd Packets/s
	Fwd Header Length	Fwd Packet Length Min
	Bwd Header Length	Flow IAT Min
	Flow Duration	Flow Duration
	Fwd Packet Length Min	Flow Packets/s
	Flow Bytes/s	min_seg_size_forward
	Flow IAT Max	Flow Bytes/s
	Average Packet Size	
Ensemble group 4	Fwd PSH Flags	Min Packet Length
	Bwd PSH Flags	Init_Win_bytes_forward
	Fwd URG Flags	ACK Flag Count
	Bwd URG Flags	Fwd Packet Length Mean
	FIN Flag Count	Flow Bytes/s
	URG Flag Count	Bwd IAT Total
	Subflow Fwd Bytes	Flow IAT Std
	Init_Win_bytes_backward	Flow IAT Min
	Fwd Packet Length Min	Fwd Packet Length Min
	Flow IAT Std	Flow Duration
	Fwd IAT Min	
	Fwd Packets/s	
	Min Packet Length	
	Packet Length Mean	
	Packet Length Std	

$$L(y, \widehat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\left(y_{i,k} * \log\left(\widehat{y}_{i,k}\right)\right), \tag{8}$$

where y and $\hat{y}$ are the target and predicted values, respectively, N is the number of training records, and K is the number of classes.

Lastly, another essential aspect of any deep learning model is the learning rate (LR), which defines how fast the deep neural network learns. Choosing the correct LR (not too high, not too low) is imperative as it helps the model to converge faster and with higher accuracy. CyDDoS employs the cyclical learning rate strategy [24], where boundary values are defined for minimum and maximum values at each layer. Cyclical learning rates allow a deep learning model to achieve improved classification accuracy. Table 3 presents a summary of the parameters used in CyDDoS model's definition.

The DNN is used to train the model because its main advantages over classic machine learning are its ability to extract different abstraction levels at different processing layers without requiring human intervention [5]. Multiple layers allow a deep learning model to automatically discover complex correlations and mapping from the input to output data [25].

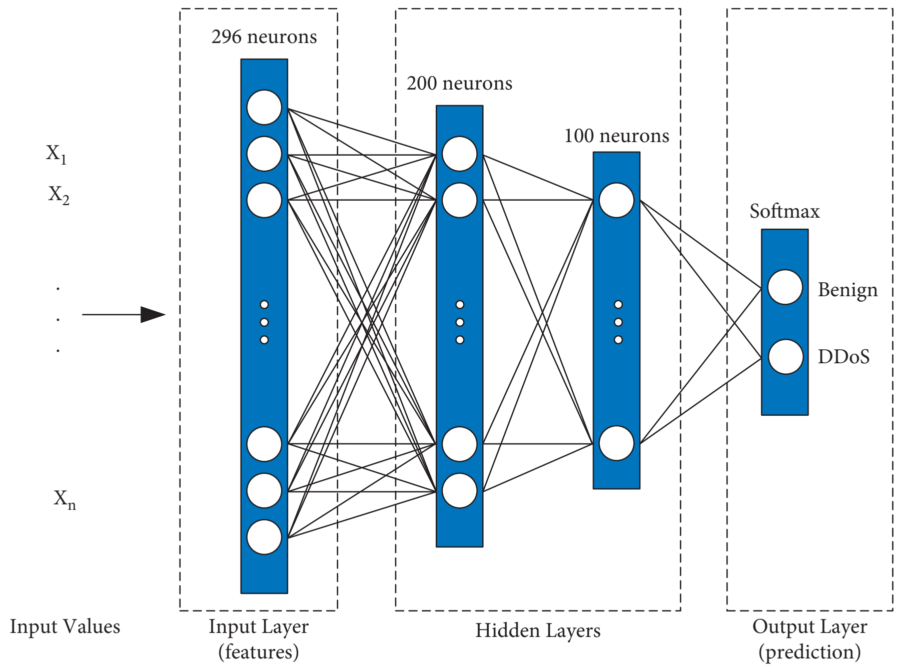


Figure 2: Deep neural network architecture of CyDDoS.

Table 3: CyDDoS model hyperparameters.

Model parameters	Values
Input neuros	296
Output neurons	2
First hidden layer	200
Second hidden layer	100
Optimizer	Adam
Loss function	Cross-entropy
Activation	ReLu
Output layer activation	Softmax
Epoch	5
Bach size	1024
Cyclical learning rates	lr_min = $1e^{-7}$ lr_max = $1e^{-4}$

3.4. Experimental Setup. To evaluate the proposed system, we conduct experiments on an Intel-based system equipped with Core (TM) i9-9900X CPU @3.50 G and 128 GB of RAM running Ubuntu 18.04 -LTS. Also, the host system is equipped with Nvidia TITAN RTX GPU, having 24 GB of RAM. In order to reduce the uncertainty of the final results, we repeated the experiment ten times. We use PyTorch v1.7 and fastai v2. PyTorch is an open-source ML framework developed by Facebook's AI Research lab. It includes an optimized tensor capable of running on both CPUs and GPUs [26]. fastai is a higher-level framework built on top of the PyTorch framework to provide cutting-edge features for building DL models. The primary purpose of fastai is to provide the research community an abstract, easy-to-use, and high-performance framework while preserving the low-level components for flexibility to develop DL-based systems [27].

3.5. Dataset Description. Our study uses the CICDDoS2019 dataset [28, 29], developed and published by the "Canadian Institute for Cybersecurity" in collaboration with the "University of New Brunswick." These institutions have provided high-quality datasets used by the research community for several years [30]. It is available in both CSV and PCAP formats. The dataset contains over 113 thousand normal traffic samples and over 70 million malicious flows, distributed over 13 classes of recent DDoS attacks. The dataset has been developed due to the limitations of previous datasets used by existing studies. It captures data related to application-level DDoS attacks that use TCP and UDP at the transport layer, categorized as reflection-based and exploitation-based taxonomy.

3.5.1. Reflection-Based Attacks. Here, the attacker hides its identity and exhausts the victim resources using a legitimate third-party node as a reflector server. The attacker first sends packets to the reflector server, using the victim's spoofed IP address as the source address. The victim receives the replay of this packet, and its ability to process it is reduced as many reflectors server is used. Table 4 illustrates the different DDoS attacks included on the dataset identifying their respective category. According to the table, DDoS attacks such as SNMP, DNS, NETBIOS, and LDAP can be executed over both TCP and UDP protocols.

3.5.2. Exploitation-Based Attacks. Here, the attacker exploits an application or protocol's specific feature or weakness to excessively consume and exhaust the target device resource and bring it down [31, 32]. The SYN Flood attack, UDP Flood, and UDP-Lag are examples of exploitation DDoS attack categories.

Furthermore, the dataset comprises training and test data files collected on 12 January and 11 March 2019. Training data contains 12 types of DDoS attacks, while the

Table 4: DDoS attack distribution according to category and transport layer protocol used.

Class of attack	Category	Transport layer protocol	DDoS type
DDoS attacks	Reflection attacks	TCP	MSSQL
			SSDP
		TCP/UDP	DNS
			LDAP
			NETBIOS
			SNMP
			PORTMAP
		UDP	CharGen
			NTP
			TFTP
	Exploitation attacks	TCP	SYN Flood
		UDP	UDP Flood
			UDP-Lag

test dataset comprises only seven types of attacks. The test dataset does not include the same number of attack types as the training dataset. However, it contains the port scan attack, which is not part of the training set to study the system detection capability for unseen attacks.

The dataset is highly unbalanced; malicious traffic flow is much higher than the normal ones. We utilize an undersampling technique that randomly removes records from the dominant class to ensure unbiased learning and classification. A balanced dataset is reliable and ensures that the learning model extracts the correct features representation from the input traffic patterns. CyDDoS makes binary classification; it classifies the incoming traffic as normal traffic or DDoS attack. It uses the 80/20 split of the dataset for training and validation purposes; however, it chooses the validation samples randomly. Finally, we employ the test dataset to evaluate the model performance to assure a truthful detection rate as the model does not have to access it during the learning process. Table 5 illustrates the sample distribution used in our experiments.

3.6. Model Training. The base deep learning architecture used to train several models is first created. This model is trained using 34 features obtained by merging all feature lists produced by the ensemble of feature ranking. Extensive experiments are conducted to learn the appropriate values for different parameters. The model achieved the best results using two hidden layers with 200 and 100 neurons and a batch size of 1024 samples when trained for five epochs, with a duration of 10 seconds. Increasing hiding layer parameters provides a very negligible improvement in accuracy compared to the processing overhead.Table 5 illustrates the flow distribution of traffic samples from the dataset used for training, validation, and testing purpose.

4. Results and Discussion

Before discussing the obtained results, we first briefly explain the metrics considered for assessing the appropriateness of our proposed system in detecting DDoS attacks compared to the existing state-of-the-art systems.

Table 5: Flow distribution among datasets.

Dataset	Flow type	Quantity
Training	Normal	45,534
	DDoS	44,250
Validation	Normal	11, 322
	DDoS	11,102
Test	Normal	52,231
	DDoS	51,746

4.1. Performance Metrics. We use different performance metrics to evaluate the appropriateness of our proposed design compared to existing techniques. The confusion matrix is a commonly used metric to measure an IDS's effectiveness [5]. Many other essential evaluation metrics are derived from it. Also, we consider metrics such as precision, accuracy, recall, and F1-Score to argue about the effectiveness of our model against existing ML-based DDoS detection schemes.

4.1.1. Accuracy. is calculated as the ratio of correctly foreseen DDoS and normal flow over total flows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{9}$$

4.1.2. Precision. can be computed as a ratio of true positives over the sum of true positives and false positives:

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

4.1.3. Recall. is calculated as the ratio of true positives over the sum of true positives and false negatives:

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

4.1.4. F1-Score. is determined as the balanced average between precision and recall:

$$F1-\text{Score}=\frac{2^{*}\left(\text{Recall}^{*}\text{Precision}\right)}{\text{Recall}+\text{Precision}}, \tag{12}$$

where true positive (TP) and true negative (TN) denote traffic correctly predicted, while false positive (FP) and false negative (FN) represent misclassified traffic.

4.1.5. Receiving Operating Characteristics (ROC). is a graph used to demonstrate the classification model performance at every classification threshold; it plots the False Positive Rate (FPR) and True Positive Rate (TPR). FPR is assigned to abscissa and represents specificity, while TPR is assigned to ordinate to represent sensitivity. Equations (5) and (6) indicate the formula to calculate FPR and TPR, respectively. From the equation, one can see that TPR uses the same equation as recall. ROC curve provides an improved balanced evaluation method because it considers the tradeoff between positive and negative samples [33]:

$$\begin{aligned} FPR &= \frac{FP}{TN+FP},\\ TPR &= \frac{TP}{TP+FN}. \end{aligned} \tag{13}$$

In addition to the metrics mentioned above, area under the ROC curve (AUC) represents the entire 2-dimensional area under the ROC curve that indicates the accuracy of the proposed system. AUC is calculated using the following equation [34]:

$$AUC=\int_{0}^{1}\frac{TP}{TP+FN}d\frac{FP}{TN+FP}. \tag{14}$$

4.2. Select the Best Minimal List of Features. A set of twelve additional experiments are conducted to select the best set of minimal features list that allows our DL model to achieve the highest classification performance. Table 6 illustrates the accuracy, precision, recall, and F1-Score evaluation results of each feature set. The results reveal that the model trained based on the Top 10 Weight list, produced by five classifiers (ensemble group 2), provides the best classification performance. From the results, our framework is able to reduce 89% of features from the original dataset while giving a classification superior to 99.6% on most of the considered metrics. This model is trained in five seconds and is selected as the best model used for further analysis.

The ROC/AUC graph provides a visual illustration of a balance between the classifier's sensitivity and specificity; a higher AUC value indicates better performance in terms of prediction. Figure 3 shows the CyDDoS's ROC, where the x-axis and y-axis correspond to wrong and correct classification rates, respectively. According to the figure, our proposed solution obtains results near the optimal point, where false positives are zero and true positives are one. This means that CyDDoS can correctly classify about 99.8% of DDoS and normal classes.

4.3. Precision-Recall Curves. Precision-recall curves usually measure the classifier's quality, primarily when the dataset is imbalanced. It depicts the tradeoff between precision (relevancy) plotted on the y-axis and recall (completeness) metrics plotted on the x-axis. Figure 4 reveals that CyDDoS obtains a large area under the curve, indicating that it achieves a high score in both recall and precision. The red dot line indicates the average precision rate, which is 0.999.

4.4. Detection Performance. This experiment calculates the prediction performance based on the different number of samples processed per second on CPU and GPU devices. To the best of our knowledge, no approach was evaluated based on the same testing environment to allow a fair comparison. The evaluation is performed based on test samples from the CiCDDoS2019 dataset.

Figure 5 shows the time required to predict the different magnitude of the dataset in the range of 2 to 100. Results demonstrate that the CPU provides better performance than GPU for a small dataset, having an average inference time of 9.6 milliseconds per batch.

Furthermore, Figure 6 compares the number of samples per second processed by the proposed model using CPU and GPU based on different batch sizes. According to the figure, the detection performance significantly increases with larger batch size. This parameter defines the number of traffic samples processed by CyDDoS simultaneously at each iteration. As the batch size increases, more memory read is required, while the number of iterations needed to process all samples from the test dataset decreases. Performance improvement introduced by GPU is noticed from a batch size of 128 samples. The results show that, using a batch size of 16,384, our model can process around 519,885 samples per second using GPU and 371,346 samples per second when CPU is disabled.

4.5. Comparing CyDDoS Performance with Some Classic ML Algorithms. To validate the efficiency of CyDDoS over classical ML algorithms using the same benchmarking dataset, we executed a new set of experiments to train the four following algorithms: random forest, Logistic Regression, Perceptron, and Gaussian Naïve Bayes. Figure 7 compares our proposed system's accuracy, precision, recall, and F1-Score against the classical ML techniques. As we can see, CyDDoS performs better than any classic technique regarding the considered metrics. Among the classical ML techniques, random forest performs best because it is an ensemble of decision tree algorithm that combines many decision tree models' predictions into a single model.

4.6. Comparison with DL-Based Systems. In addition to classic ML techniques, we compare our proposed model against existing DL-based approaches. Moreover, we consider ROC in addition to precision, accuracy, recall, and F1-Score. Figure 8

Table 6: Classification results achieved by each feature selection strategy within ensemble group.

Ensemble group	Feature selection strategy	Accuracy	Precision	Recall	F1-Score
Ensemble group 1	Threshold-based list	0.9319	0.9355	0.9315	0.9150
	Top 10 Weight	0.9954	0.9956	0.9956	0.9950
	Threshold-based list + Top 10 Weight	0.9950	0.9950	0.9950	0.9950
Ensemble group 2	Threshold-based list	0.9519	0.9540	0.9520	0.9515
	Top 10 Weight	0.9962	0.9965	0.9965	0.9960
	Threshold-based list + Top 10 Weight	0.9941	0.9940	0.9940	0.9940
Ensemble group 3	Threshold-based list	0.9198	0.9200	0.9200	0.9195
	Top 10 Weight	0.9949	0.9950	0.9950	0.9950
	Threshold-based list + Top 10 Weight	0.9910	0.9910	0.9910	0.9910
Ensemble group 4	Threshold-based list	0.9277	0.9325	0.9275	0.9275
	Top 10 Weight	0.9877	0.9880	0.9880	0.9880
	Threshold-based list + Top 10 Weight	0.9940	0.9950	0.9950	0.9940

Receiver operating characteristic/area under the curve (ROC/AUC).

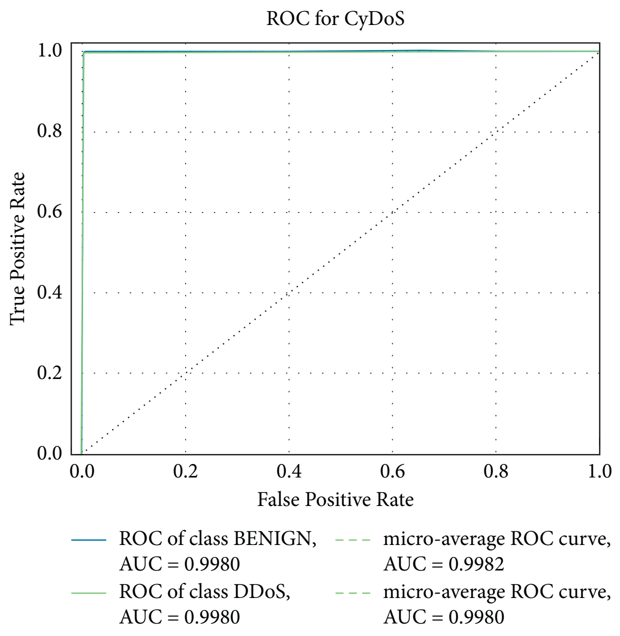


Figure 3: ROC curve.

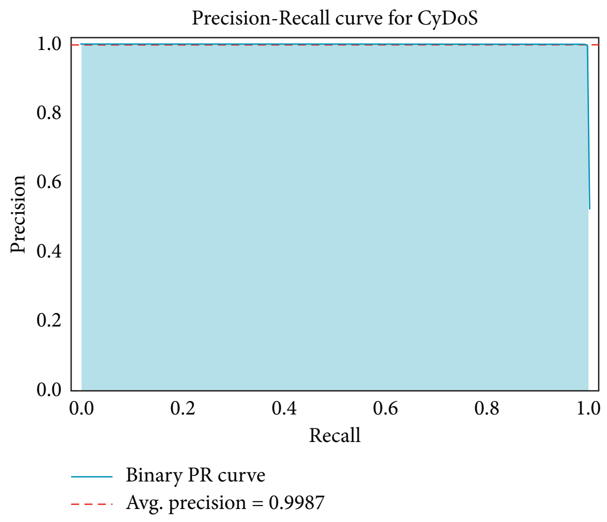


Figure 4: Precision-recall curve.

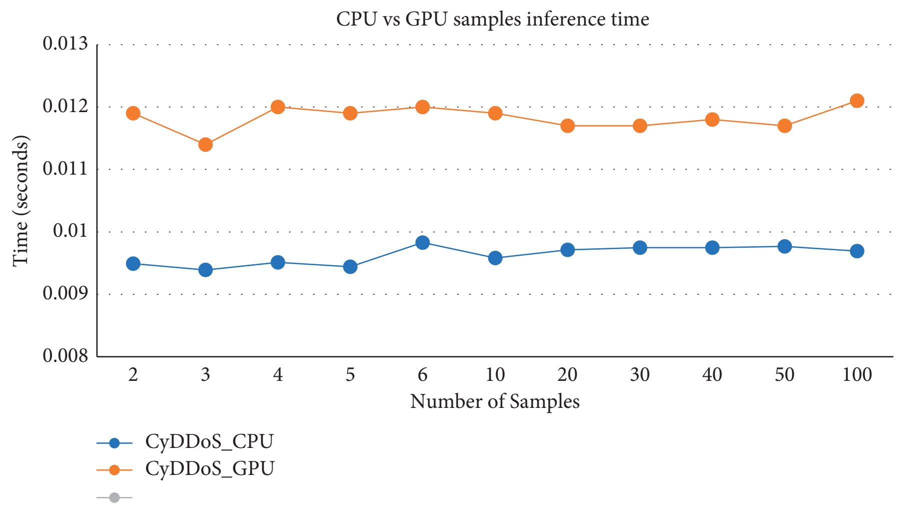


FIGURE 5: Different dataset size inference performance.

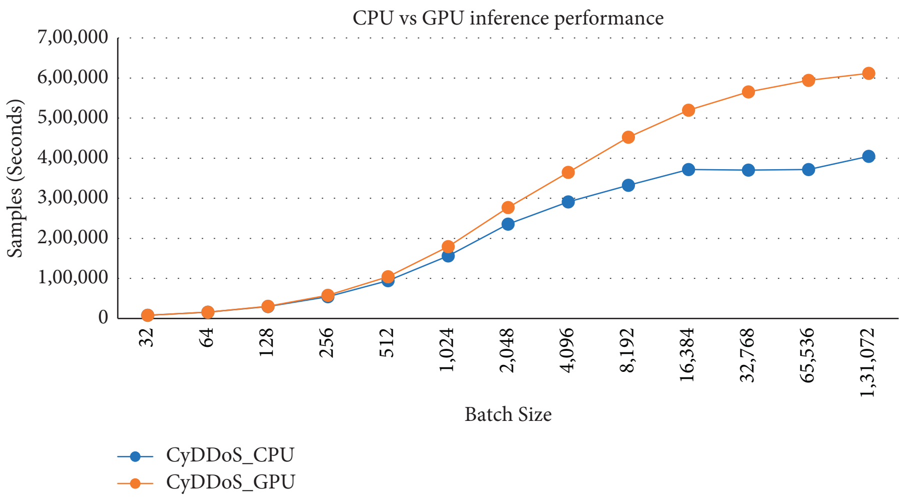
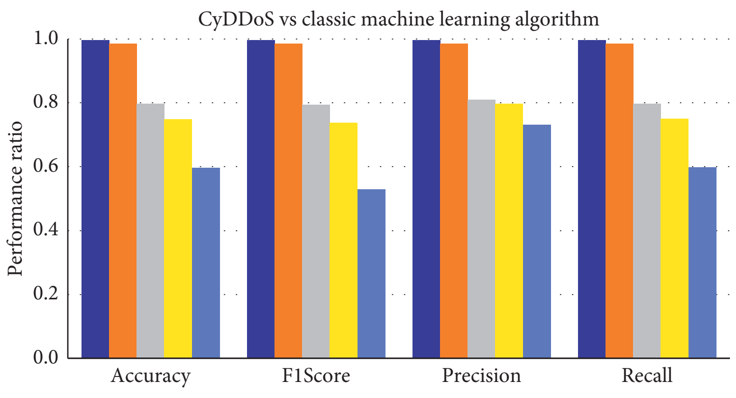
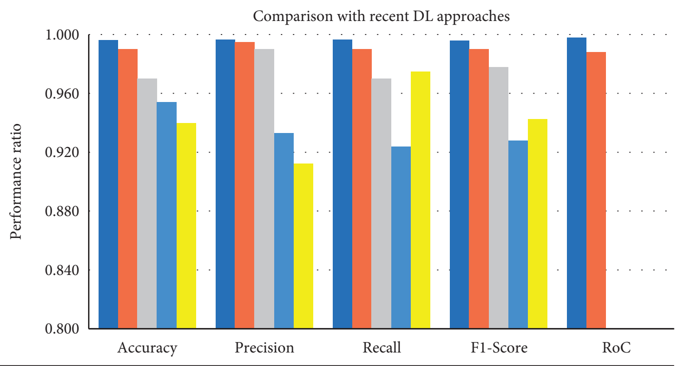


FIGURE 6: Prediction performance on CPU versus GPU.



	Accuracy	F1Score	Precision	Recall
■ CyDDoS	0.996	0.996	0.997	0.997
■ Random Forest	0.985	0.985	0.985	0.985
■ Logistic Regression	0.797	0.795	0.811	0.797
■ Perceptron	0.749	0.738	0.797	0.749
■ Gaussian Naive Bayes	0.596	0.529	0.731	0.598

FIGURE 7: CyDDoS versus classical machine learning.



	Accuracy	Precision	Recall	F1-Score	RoC
🟦 CyDDoS	0.996	0.997	0.997	0.996	0.998
🟥 DDoSNet[14]	0.990	0.995	0.990	0.990	0.988
⬜ Meta-classification/decision jungle [36]	0.970	0.990	0.970	0.978	
🟦 Convolutional Neural Networks[38]	0.954	0.933	0.924	0.928	
🟨 Kalman backpropagation NN[37]	0.94	0.9122	0.9749	0.943	

Figure 8: CyDDoS versus other deep learning models.

clearly shows that CyDDoS outperforms existing state-of-the-art DL-based techniques regarding all considered metrics. DDoSNet [14] discussed in the second section performs almost near our model in terms of precision, as it combines a deep learning technique called Recurrent Neural Network (RNN) with an autoencoder; however, the CyDDoS achieves 1% higher ROC. ROC is a preferable method to compare classifier performance as it represents the classifier's performance over the entire class distribution range [35].

Metaclassification using decision jungle [36] combines different classifiers for binary and multiclass classification. Decision jungle acts as the metalearner that ensembles multiple learners to achieve optimal prediction performance. This proposed approach achieves a precision of 0.99. Kalman backpropagation neural network [37] is proposed to detect DDoS in 5G-enabled IoT networks. This model achieved the highest score of 0.9749 in the recall. Finally, convolution neural network-based IDS [38] obtained the highest value of 0.954 in accuracy metrics.

5. Conclusion

DDoS is a predominant threat to the reliability of online services and is experienced frequently by most service providers worldwide. An effective DDoS classification mechanism is required to prevent resource outages as a result of a DDoS attack. However, very few studies consider an up-to-date dataset containing data related to recent DDoS attacks. Furthermore, the existing solution requires a high processing capacity for model training and making predictions.

This work presents an integrated DL-based IDS framework that effectively uses ensemble feature selection to improve DDoS attack detection performance with low processing overhead. Several ensemble groups are evaluated based on different numbers of classifiers to determine the minimal feature list that can produce the best prediction performance. Our proposed model can reduce 89% of features from the CICDDoS2019 dataset and is trained in five seconds. Overall, the experimental results reveal that our proposed design outperforms its competitors, attaining accuracy of 99.6% and the ROC value nearly to 1, based on the test dataset. CyDDoS can be deployed as an SDN application and be integrated with the controller to provide a highly accurate prediction of DDoS attacks.

CyDDoS performed a detection only of DDoS attacks, which is one limitation of our framework. As future work, we will improve the proposed IDS by incorporating a controller to block DDoS traffic and allows the normal traffic to be forwarded. Additionally, the model will be enhanced by introducing adjustments to allow a multiclass detection of the various attacks category. The adoption of DL models in certain critical AI applications, where the reliability and interpretation of the model's internal behavior are crucial, is limited due to its black-box nature. To address this issue, as future work, we will use the eXplainable Artificial Intelligence (XAI) [39] technique to investigate the trustworthiness and interpretability of CyDDoS and make improvements if required.

Abbreviations

ADAM:	Adaptive Moment Estimator
AE:	Autoencoder
AUC:	Area under curve
CNN:	Convolutional Neural Network
DBN:	Deep belief network
DDoS:	Distributed Denial of Service
DL:	Deep learning
DNN:	Deep neural network
DoS:	Denial of service
DT:	Decision tree
FN:	False negatives
FNN:	Feedforward neural network
FP:	False positives
FPR:	False Positive Rate
GBoost:	Gradient Boost
IDS:	Intrusion detection system
IoT:	Internet of Things
LightGBM:	Light Gradient Boosted Machine

ML:	Machine learning
M2-DAE:	MultiModal Deep AutoEncoder
NN:	Neural networks
ReLu:	Rectified Linear Units
RF:	Random forest
RFE:	Recursive feature elimination
RNN:	Recurrent Neural Network
ROC:	Receiver operating characteristic
SDAE-ELM:	Stacked Denoising AutoEncoder-Extreme Learning Machine
SDN:	Software-Defined Networking
TN:	True negative
TP:	True positive
TPR:	True Positive Rate
XGBoost:	Extreme Gradient Boosting.

Data Availability

Our study uses the CICDDoS2019 dataset [21, 22], developed and published by the "Canadian Institute for Cybersecurity" in collaboration with the "University of New Brunswick." These institutions have provided high-quality datasets used by the research community for several years [23]. It is available in both CSV and PCAP formats. The dataset contains over 113 thousand benign traffic samples and over 70 million malicious flows, distributed over 13 classes of recent DDoS attacks. The dataset has been developed due to the limitations of previous datasets used by existing studies. It captures data related to application-level DDoS attacks that use TCP and UDP at the transport layer, categorized as reflection-based and exploitation-based taxonomy. [21] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," in 2019 International Carnahan Conference on Security Technology (ICCST), CHENNAI, India, Oct. 2019, pp. 1–8, doi: 10.1109/CCST.2019.8888419 [22] "DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB." https://www.unb.ca/cic/datasets/ddos-2019.html (accessed Dec. 09, 2020). [23] V. Kanimozhi and T. P. Jacob, "Artificial Intelligence based Network Intrusion Detection with Hyper-Parameter Optimization Tuning on the Realistic Cyber Dataset CSE-CIC-IDS2018 using Cloud Computing," p. 4.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant no. 61902138, the Key-Area Research and Development Program of Guangdong Province (no. 2019B010139001), and the Shenzhen Fundamental Research Program (JCYJ20170413114215614).

References

[1] Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper, *Cisco*, https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, and E. Bursztein, *Understanding the Mirai Botnet*, p. 19.

[3] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[4] B. M. Rahal, A. Santos, and M. Nogueira, "A distributed architecture for DDoS prediction and bot detection," *IEEE Access*, vol. 8, pp. 159756–159772, 2020.

[5] 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, Article ID 105124, 2020.

[6] 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, Article ID 102767, 2020.

[7] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, pp. 108–116, SciTePress, Funchal, Madeira, Portugal, Jan 2018.

[8] P. J. Criscuolo, *Distributed Denial of Service*, p. 18.

[9] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del-Rincon, and D. Siracusa, "Lucid: a practical, lightweight deep learning solution for DDoS attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[10] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[11] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[12] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[13] X. Yuan, C. Li, and X. Li, "DeepDefense: identifying DDoS attack via deep learning," in *Proceedings of the 2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 1–8, IEEE, Hong Kong, China, May 2017.

[14] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "DDoSNet: a deep-learning model for detecting network attacks," in *Proceedings of the 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pp. 391–396, IEEE, Cork, Ireland, Aug. 2020.

[15] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, *Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection*, http://arxiv.org/abs/1802.09089 Accessed, May 2018.

[16] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "A hierarchical hybrid intrusion detection approach in IoT scenarios," in *Proceedings of the GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pp. 1–7, IEEE, Taipei, Taiwan, Dec. 2020.

[17] R. Doshi, N. Apthorpe, and N. Feamster, “Machine learning DDoS detection for consumer Internet of Things devices,” in *Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW)*, pp. 29–35, IEEE, San Francisco, CA, USA, May 2018.
[18] M. Ge, N. F. Syed, X. Fu, Z. Baig, and A. Robles-Kelly, “Towards a deep learning-driven intrusion detection approach for Internet of Things,” *Computer Networks*, vol. 186, Article ID 107784, 2021.
[19] Z. Wang, Y. Liu, D. He, and S. Chan, “Intrusion detection methods based on integrated deep learning model,” *Computers & Security*, vol. 103, Article ID 102177, 2021.
[20] C. Guo and F. Berkhahn, *Entity Embeddings of Categorical Variables*, p. 9.
[21] M. A. Uthaib and M. S. Croock, “Multiclassification of license plate based on deep convolution neural networks,” *International Journal of Electrical and Computer Engineering*, vol. 11, no. 6, p. 5266, Dec. 2021.
[22] M. Sagen, J. Lu, and J. Li, *EDseteimp aLteinargnUinngcMertoadienltsy*, p. 28.
[23] Y. Ho and S. Wookey, “The real-world-weight cross-entropy loss function: modeling the costs of mislabeling,” *IEEE Access*, vol. 8, pp. 4806–4813, 2020.
[24] L. N. Smith, “Cyclical learning rates for training neural networks,” in *Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, IEEE, Santa Rosa, CA, USA, Mar. 2017.
[25] L. Deng and D. Yu, “Deep learning: methods and applications,” *Foundations and Trends in Signal Processing*, vol. 7, pp. 197–387, 2014.
[26] PyTorch: https://pytorch.org/.
[27] J. Howard and S. Gugger, “Fastai: a layered api for deep learning,” *Information*, vol. 11, no. 2, p. 108, 2020 Feb. 2020.
[28] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy,” in *Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST)* , pp. 1–8, IEEE, CHENNAI, India, Oct. 2019.
[29] *DDoS 2019 Datasets Research Canadian Institute for Cybersecurity UNB*, https://www.unb.ca/cic/datasets/ddos-2019.html accessed.
[30] V. Kanimozhi and T. P. Jacob, “Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-ids2018 using Cloud computing,”vol. 5, pp. 211–214, in *Proceedings of the 2019 International Conference on Communication and Signal Processing (ICCSP)*, vol. 5, pp. 211–214, IEEE, Chennai, India, April 2019.
[31] M. Marvi and A. Arfeen, “A generalized machine learning based model for the detection of DDoS attacks,” *SSRN Electronic Journal*, vol. 200, Article ID 108498, 2020.
[32] N. Z. Bawany and J. A. Shamsi, “SEAL: SDN based secure and agile framework for protecting smart city applications from DDoS attacks,” *Journal of Network and Computer Applications*, vol. 145, Article ID 102381, 2019.
[33] M. Lan, J. Luo, S. Chai, R. Chai, C. Zhang, and B. Zhang, “A novel industrial intrusion detection method based on threshold-optimized CNN-BiLSTM-Attention using ROC curve,” in *Proceedings of the 2020 39th Chinese Control Conference (CCC)*, pp. 7384–7389, IEEE, Shenyang, China, July 2020.
[34] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, “Network intrusion detection based on supervised adversarial variational auto-encoder with regularization,” vol. 8, p. 16, 2020.
[35] J. Jin Huang and C. X. Ling, “Using AUC and accuracy in evaluating learning algorithms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.
[36] S. Rajagopal, P. P. Kundapur, and K. S. Hareesha, “Towards effective network intrusion detection: from concept to creation on azure Cloud,” *IEEE Access*, vol. 9, pp. 19723–19742, 2021.
[37] M. Almiani, A. AbuGhazleh, Y. Jararweh, and A. Razaque, “DDoS detection in 5G-enabled IoT networks using deep Kalman backpropagation neural network,” *International Journal of Machine Learning and Cybernetics*, vol. 12, pp. 3337–3349, 2021.
[38] M. V. O. de Assis, L. F. Carvalho, J. J. P. C. Rodrigues, J. Lloret, and M. L. Proença, “Near real-time security system applied to SDN environments in IoT networks using convolutional neural network,” *Computers & Electrical Engineering*, vol. 86, Article ID 106738, 2020.
[39] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G.-Z. Yang, “XAI—explainable artificial intelligence,” *Sci. Robot.*vol. 4, no. 37, p. 7120, 2019.