Received: 19 May 2025 | Revised: 10 September 2025 | Accepted: 11 December 2025 | Published online: 13 January 2026

RESEARCH ARTICLE

Artificial Intelligence and Applications
2026, Vol. 00(00) 1–5
DOI: 10.47852/bonviewAIA62026187

BON VIEW PUBLISHING

DDoS Attack Detection With and Without SMOTEENN Dataset Balancing Strategy: Deep Learning Approaches

Andualem Girma[1] and Kibreab Adane[1,*]

[1] *Computing and Software Engineering, Arba Minch University, Ethiopia*

Abstract: DDoS attacks flood the target systems with bulky, abnormal traffic, rendering them unavailable to benign users, and could lead to the crash of servers or computing devices, a loss of money, and a loss of productivity in time-dependent sectors like banks, airlines, online shopping, and more. Many researchers have sought to address DDoS attack issues using various techniques despite these attacks continuing to rise. To address these concerns, the study employed PRISMA guidelines to excavate open issues from recent and pertinent research articles to provide viable solutions and employed diverse deep-learning models. Each model was fine-tuned and trained with and without the SMOTEENN dataset balancing strategy using diverse train-test-validation splits. When looking at the models' efficacy, it was evident that all models achieved remarkable accuracy rates on the test dataset following the application of the SMOTEENN dataset balancing strategy. Among others, the combination of the MLP model with SMOTEENN scored the top accuracy of 98.90%. The SMOTEENN technique reduces the FNR and FPR for models like DNN and BiLSTM. While slightly increasing FPR, it lowers FNR for the majority of models, including MLP, LSTM, DCNN, CNN-LSTM, and DNN-AE. Despite significantly improving the accuracy, the SMOTEENN technique did not reduce the model's train-test computational times. The review findings reveal that applying relevant feature selection strategies could reduce model computational time. The study demonstrates the workflow for DDoS attack detection, classification, and mitigation using the proposed model.

Keywords: deep learning techniques, DDoS attack detection, dataset balancing strategy, SMOTEENN, network traffic analysis, DDoS attack types, CICDDoS2019

1. Introduction

Distributed denial-of-service (DDoS) attacks are one of the leading causes of network instability and service disruption. They send bulky illegitimate traffic to the target systems to disrupt online services [1–3]. Because these attacks can originate from a variety of sources, it is difficult to prevent and mitigate. Because more people are accessing the Internet through mobile devices and the Internet of Things, cybercriminals are getting more interested in targeting systems and network infrastructure [4–6].

DDoS attacks have become more complex and frequent in recent years, making it difficult to defend against them using traditional security techniques. Traditional detection techniques, like signature-based and statistical anomaly detection, sometimes fall short in spotting new and changing DDoS attack patterns as attackers use increasingly sophisticated tactics [7]. In recent years, applications of deep learning techniques have contributed to enhanced cybersecurity defense [8]. It excels in relevant feature extraction and handling incomplete data, offering diverse model options, chaining capabilities, pre-trained knowledge, and high-performance rate requirements, and has shown promise in detecting and mitigating DDoS attacks [9–14].

Imbalanced classification is a problem for many real-world applications in general and for machine learning models in particular. This problem arises when there is a bias in predictions toward the majority class due to a skewed distribution of the target variable. There is an urgent need for effective solutions to address this issue in the big data era. Several techniques are discussed in the recent study to address class imbalance issues, such as random undersampling (RUS), random oversampling (ROS), SMOTE, cost-sensitive learning (weighting), and K-medoid [15].

To make the majority class samples equivalent to the minority class samples, the majority class samples are randomly eliminated from the dataset in RUS. This strategy could result in relevant information loss. The minority class samples are randomly replicated to be equivalent to the majority class samples in ROS. This strategy could result in model overfitting due to duplicate samples used for training. The minority class will be given high weight in the cost-sensitive learning method without increasing or decreasing the number of samples. There are some classifiers supporting weighting methods, such as ANN and SVM. K-medoid is used to cluster the samples of the majority class; the number of clusters is equal to the number of minority training samples. Since they are balanced in this instance, the minority class and the medoid—the cluster centers—are used for the training phase [14]. Unlike the ROS method, SMOTE doesn't replicate the minority class dataset; instead, SMOTE generates synthetic samples for the minority class by interpolating current instances of the class. To do this, SMOTE picks a minority class instance with its k-nearest neighbors. While SMOTE focuses solely on oversampling the minority class, SMOTEENN adopts a more holistic approach by incorporating the undersampling of the majority class. This dual strategy affords better handling of varying degrees of class imbalance, thereby making it more adaptable to different datasets and scenarios. By removing noisy majority instances, the SMOTEENN approach allows the model to learn more effectively, leading to improved classification accuracy. Additionally, SMOTEENN not only increases the number of minority

***Corresponding author:** Kibreab Adane, Computing and Software Engineering, Arba Minch University, Ethiopia. Email: kibreab.adane@amu.edu.et

class samples but also ensures that the majority class representation is more relevant. This balance aids in building models that generalize better to unseen data, thereby reducing the risk of over-fitting associated with using SMOTE alone [16–18].

With the use of deep learning techniques, the field of DDoS attack detection has seen notable advancements. Many researchers have sought to address DDoS attack issues using various techniques despite these attacks continuing to rise [19]. This emphasizes the need for beating DDoS attacks using accurate, reliable, and adaptive models. Our study highlights recent studies conducted between 2019 and 2024 on the use of deep learning techniques for DDoS attack detection to identify the gaps, formulate research methodology, and offer workable answers.

The following research questions are meant to be addressed by conducting different experiments:

1. How well do the deep learning models perform with different datasets train, validation, and test splits?
2. How do deep learning models perform both with and without an SMOTEENN dataset balancing strategy in terms of accuracy, false positive rate (FPR), false negative rate (FNR), and train-test computational time?
3. How can incoming network traffic data be categorized using a variety of features utilizing a DDoS attack detection and classification prototype?

2. Literature Review

To retrieve recent and pertinent research works, a suitable keyword-based search strategy like DDoS attack detection OR distributed denial-of-service detection AND deep learning, used to extract these research papers from reputable sources, including the Web of Science, EEE, Springer, and SCOPUS databases, in accordance with PRISMA principles. Boolean operators (AND, OR) were used to successfully combine these search terms. Based on a keyword-based search strategy, 85 papers were retrieved in the first stage, 40 papers were screened in the second stage because they were relevant and included both machine learning and deep learning algorithms, and 24 papers were examined in the third stage because they were appropriate and contained only deep learning algorithms for DDoS attack detection.

The following research questions are meant to be addressed in Sections 2.1–2.5:

1. Which deep learning models are top performing in DDoS attack detection?
2. What are popular datasets used for DDoS attack detection?
3. Did the reviewed studies use balanced datasets for DDoS attack detection?
4. What types of dataset splits used in the reviewed studies?
5. Did the reviewed studies reveal relevant experimental details for the replication or revalidation of their research findings?

The review results are framed based on the following research questions.

2.1. RQ#1: which deep learning models are top performing in DDoS attack detection?

As can be seen from Table 1, the scientific community uses a variety of deep learning algorithms, albeit not all of them are equally effective in identifying DDoS attacks. To save future researchers' time in model selection, the study determined which models performed best in identifying DDoS attacks based on prior studies. Autoencoder (AE) was the least chosen DL model for DDoS attack detection, while long short-term memory (LSTM) and deep neural network (DNN) were among the most popular and well-performed models based on the application on an individual basis. The scientific community has employed several hybrid-based applications, including LSTM-AE, CNN-LSTM, LSTM-RNN, AE-MLP, and RNN_AE, to demonstrate top performance in DDoS attack detection.

2.2. RQ#2: what are popular datasets used for DDoS attack detection?

Commonly utilized datasets for deep learning-based DDoS attack detection includes CICDDoS2019, CICIDS2017, NSL-KDD, ISCX2012, and UNSW 2018. Appearing in 17 out of 24 studies, the

Table 1
Structured summary of the reviewed research works

Evaluation criteria	[6]	[10]	[11]	[13]
Deep learning used	DNN	DNN, LSTM	LSTM	(CNN) 2D, LSTM-DAE, DNN
Used dataset	CICDDoS2019	CICDDoS2019	CICDDoS2019	CICDDoS2019
Data source	CIC	CIC	CIC	CIC
No. of the used dataset	1	1	1	1
Data size	730,355	Unknown	Unknown	186,548
Dataset ratios	Balanced	Imbalanced	Balanced	Imbalanced
Feature selection techniques	Unknown	Unknown	Unknown	Unknown
No. of used features	69	24	71	80
Feature scaling techniques	Applied	Not applied	Applied	Applied
No. of DDoS attack type	14	3	Not mentioned	12
% of train-test-validation split ratio	80/20	60/20/20	Not mentioned	90/10
Model hyperparameter	Revealed	Revealed	Revealed	Revealed
Classification type	Multi-class	Multi-class	Binary	Multi-class
Best-performing detection model	DNN	LSTM	LSTM	LSTM
Accuracy rate	94.57%	99.90%	98%	74.42%

Table 1
(*Continued*)

Evaluation criteria	[2]	[8]	[50]	[28]
Deep learning used	AE-MLP	1DCCN-LSTM	RNN_AE	AE, DCAE, VAE, LSTM
Used dataset	CICDDoS2019	CICDDoS2019	CICDDoS2019	CICDDoS2019, CICIDS2017, NSL-KDD
Data source	CIC	CIC	CIC	CIC
No. of used dataset	1	1	1	3
Data size	70,427,637	1,060,572	230,673	Unknown
Dataset ratios	Imbalanced	Imbalanced	Balanced	Imbalanced
Feature selection techniques	Unknown	Unknown	Autoencoder	Contractive auto encoder &stochastic threshold
No. of used features	78	79	77	CICDDoS2019, 78; CICIDS2017, 77; NSL_KDD, 121
Feature scaling techniques	Applied	Applied	Applied	Not applied
No. of DDoS attack type	5	12	Unknown	Unknown
% of train-test-validation splits	70/20/10	80/10/10	70/20/10	Unknown
Model hyperparameter	Revealed	Revealed	Revealed	Not revealed
Classification type	Multi-class	Binary & multi-class	Binary	Binary
Best-performing detection model	AE-MLP	DCNN-LSTM	RNN_AE	DCAE
Accuracy rate	98%	99.76%	98.8%	CICDDoS2019, 97.58%; NSL-KDD, 96.08%; CICIDS2017, 92.45%

Table 1
(*Continued*)

Evaluation criteria	[21]	[31]	[29]	[32]
Deep learning used	BiLSTM	LSTM	CNN	LSTM-CNN
Used dataset	CIC-IDS2017 CICDDoS2019	ISCX2012	CICDDoS2019 CICIDS2017	UNSW 2018 dataset DDoS attack SDN dataset
Data source	CIC	CIC	CIC	University of New South Wales, Australia Mendeley dataset
No. of the used dataset	2	1	2	2
Data size	Unknown	240,000	Unknown	Unknown
Dataset ratios	Imbalanced	Unknown	Imbalanced	Unknown
Feature selection techniques	GMM	Bayes approach	Geometric metrics	stack autoencoder
No. of the used features	Unknown	Unknown	Unknown	23
Feature scaling techniques	Unknown	Unknown	Applied	Unknown
No. of DDoS attack types used	6	Unknown	CICDDoS2019 :10 CICID2017:No	4
% of train-test-validation split ratio	10-fold	80/20	Unknown	80/20
Model hyperparameter	Revealed	Not revealed	Revealed	Not revealed
Classification types	Binary	Binary	Binary	Binary
Best-performing detection model	BiLSTM	LSTM-BA	CNN-GEO	LSTM-RNN with stack autoencoder
Accuracy rate	94%	98.15%	98%	99.99%

CICDDoS2019 dataset was determined to be the most recent and highly favored dataset by the scientific community. The Canadian Institute for Cybersecurity (CIC) provides this dataset and included the new types of DDoS attacks like UDP-lag and NetBIOS (refer to Table 1).

Performance is significantly impacted by the quantity, quality, and distribution of datasets used to train deep learning algorithms. Despite revealing model accuracy, 9 out of 24 studies still need to reveal the dataset size used for DDoS attack detection; 4 out of 24 studies did not reveal train-test-validation dataset ratios; 6 out of 24 studies did not reveal the optimal model hyperparameters used; 6 out of 24 studies did not reveal the number of top predictive features used for DDoS attack detection; 10 out of 24 studies did not reveal feature scaling/normalization techniques, used to avoid one feature dominating another; and 4 out of 24 studies did not reveal whether the dataset used

Table 1
(*Continued*)

Evaluation criteria	[1]	[7]	[23]	[33]
Deep learning used	CNN-RNN	DNN	DNN	MLP, CNN, LSTM, LSTM-CNN
Used dataset	CICDDoS2019	CICIDS2017	CICDDoS2019	CICIDS2017
Data source	CIC	CIC	CIC	CIC
No. of the used dataset	1	1	1	1
Data size	1,130,650	691395	12,797,829	Unknown
Dataset ratios	Imbalanced	Imbalanced	Imbalanced	Unknown
Feature selection techniques	Random Forest Feature Importance	Unknown	Unknown	Unknown
No. of used features	83	75	11	Unknown
Feature scaling techniques	Unknown	Applied	Unknown	Applied
No. of DDoS attack type used	16	5	3	5
% of train-test-validation split ratio	70/20/10	80/10/10 train-validation-test split	Unknown	Unknown
Model hyperparameter	Revealed	Revealed	Revealed	Not revealed
Classification type	Multi-class	Multi-class	Binary	Multi-class
Best-performing detection model	CNN-RNN	DNN	DNN	LSTM-CNN
Accuracy rate	98.92%	99%	99%	97.16%

Table 1
(*Continued*)

Evaluation criteria	[22]	[24]	[34]	[25]
Deep learning used	DNN	LSTM	CCN-LSTM	DNN and CNN-AE
Used dataset	CICDDoS2019	CICDDoS2019	Synthetic dataset	CICDDoS2019
Data source	CIC	CIC	Unknown	CIC
No. of used dataset	1	1	1	1
Data size	Over 180,000	Unknown	1,000,000	650,000
Dataset ratios	Balanced	Unknown	Imbalanced	Balanced
Feature selection techniques	PCC	Unknown	Unknown	Unknown
No. of the used features	10	Unknown	12	83
Feature scaling techniques	Applied	Unknown	Applied	Applied
No. of DDoS attack class used	10	8	Unknown	Unknown
% of train-test-validation split ratio	70/30	80/20	80/20	Unknown
Model hyperparameter	Revealed	Revealed	Revealed	Not revealed
Classification type	Multi-class	Binary	Binary	Binary
Best-performing detection model	DNN	LSTM	CNN-LSTM	CNN-AE
Accuracy rate	99.66%	99.19%	99%	87%

in the study was balanced or Imbalanced. These practices are against replicating study findings and revalidating models for deployment in actual environments (refer to Table 1).

2.3. RQ#3: did the reviewed studies use balanced datasets for DDoS attack detection?

Most (13 out of 24) studies used Imbalanced datasets (used uneven DDoS and none-DDoS dataset ratios) to train deep learning models or did not apply popular dataset balancing strategy like SMOTE even though models were biased toward the majority classes when using Imbalanced datasets (refer to Table 1). Even though model performance varies when using different train-test split ratios, the majority of reviewed papers only use one dataset split to train deep learning models. According to our review findings, only the study by Jamal et al. [17] employed the SMOTEENN data balancing approach for DDoS attack detection, even though they did not use the popular dataset, CICDDoS2019; did not use deep learning models like MLP, DNN, BiLSTM, and DNN-AE; did not develop a prototype to facilitate evaluation of the top-performing model by experts; and did not assess the train-test computational time of the SMOTEENN on CICDDoS2019 dataset. These authors employed four balancing

Table 1
(*Continued*)

Evaluation criteria	[5]	[35]	[26]	[27]
Deep learning used	MLP	CNN-AE, BiLSTM-AE, & ANN-AE	CNN,LSTM,GRU.	CNN_LSTM
Used dataset	NSL-KDD	ISCXIDS2012 UNSW2018	CICDDoS2019	CICDDoS2019
Data source	University of New Brunswick	Unknown	CIC	CIC
No. of used dataset	1	2	1	1
Data size	148,527	ISCX2012 = 100,000 Unsw2018 = 200,000	50,063,112	138,839
Dataset ratios	Imbalanced	Balanced	Imbalanced	Balanced
Feature selection techniques	Unknown	Autoencoder	RFE (SVM)	XGBoost
No. of used features	41	Unknown	81	10
Feature scaling techniques	Unknown	Unknown	Applied	Applied
No. of DDoS attack classes	22	Unknown	11	3
% of train-test-validation split ratio	60/20/20	60/20/20	80/10/10	Unknown
Model hyperparameter	Not revealed	Revealed	Not revealed	Revealed
Classification types	Binary	Binary	Multi-class	Binary
Best-performing detection model	MLP	BiLSTM-AE	Hybrid of them	CNN-LSTM
Accuracy rate	99.13%	ISCDX2012, 99.35% UNSW2018, 99.95%	89.4%	99.5%

strategies—SMOTE-Tomek, SMOTE-ENN, Borderline SMOTE, and SMOTE—to demonstrate that the MSCAD dataset, with six classes, was representative of the entire population. They found that 99.99% top accuracy was achieved after applying the SMOTEENN with the KNN model. To get beyond the drawbacks of the separate dataset balancing strategies, SMOTEENN integrates the inherent qualities of RUS and ROS approaches. Thus, SMOTE corrects unequal class distributions by boosting the representation of the minority class, while ENN acts as a cleaning mechanism, guaranteeing that the final training set is more coherent and less prone to noise [14]. By eliminating problematic points, ENN protects the quality of the data, which makes SMOTEENN an effective method for enhancing classification performance on highly skewed datasets [16–18, 20]. The aforementioned literature-based evidence was one of the core rationales for selecting SMOTEENN for our study.

2.4. RQ#4: what common types of dataset splits are used in the reviewed studies?

To evaluate model skills on unobserved or unseen data, different dataset split ratios were used by the scientific community. The dataset split ratios of 80%–20% were used in five studies, 60%–20%–20% splits in three studies, 70%–20%–10% splits in three studies, 80%–10%–10% splits in three studies, 90%–10% split in one study, 70%–30% split in one study, and tenfold cross-validation in one study. Most studies used 80%–20% dataset split ratios for DDoS attack detection (refer to Table 1).

2.5. RQ#5: did the reviewed studies reveal relevant experimental details for the replication or revalidation of their research findings?

As seen from Table 1, most studies did not reveal each experimental detail for the reproducibility of research findings. Specifically, most studies used imbalanced datasets and did not reveal dataset size, fine-tuned model hyperparameters, and train-test-validation split ratios. For instance, references [1, 8, 21–27] used the same dataset CICDDoS2019 with our study even though references [10, 11, 21, 24, 28] did not reveal dataset size, references [11, 23, 25, 27–29] did not reveal train-test-validation split ratios, references [24, 29, 30] did not reveal number of top predictive features used, and references [28, 25, 26] did not reveal fine-tuned model hyperparameters. The study is an effort to address these issues.

3. Materials and Methods

Intrinsic properties and better performance in related research works were taken into consideration while choosing deep learning algorithms for experiments. The dataset, that is, CICDDoS2019, was selected based on popularity and recent development. The dataset balancing strategy, that is, SMOTEENN, was selected based on advantages and disadvantages when compared with RUS, ROS, and SMOTE (see Sections 1 and 3.3). The identified research gaps and the advantages of filling them are listed in Table 2.

3.1. Dataset information

The findings of our systematic research indicate that CICDDoS2019, CICIDS2017, NSL-KDD, ISCX2012, and UNSW 2018 are frequently used datasets for deep learning-based DDoS attack detection. The CICDDoS2019 dataset was found to be the latest and most popular dataset by the scientific community, appearing in 17 out of 24 articles. These findings led our study to favor publicly available datasets, such as CICDDoS2019, which was made available by the Canadian Institute for Cybersecurity (CICS). These datasets were arranged in numerous CSV files, each of which represented a distinct attack type in addition to typical traffic data [11]. Downloading this dataset requires filling out a request form to obtain access, thereby ensuring the data is utilized for legitimate research purposes. Each data point within the dataset encompasses over 88 distinct features, incorporating millions

Table 2
Some notable limitations in the related research works

	Limitations in past studies	No. of studies out of 24	Benefits of addressing it
1	Did not disclose fine-tuned model hyperparameters	7	Facilitate reproducibility of research findings
2	Did not disclose DDoS-benign dataset ratios	4	Facilitate reproducibility of research findings
3	Used only one deep learning algorithm	9	Comparing multiple models to select the better one
4	Did not disclose train-test split/cross-validations	7	Facilitate reproducibility of research findings
5	Used Imbalanced datasets/no balancing strategy	13	Better generalization
6	Didn't disclose dataset size	8	To check scalability and make comparisons
7	Focused on binary classification	14	Increase model decision scope
8	Evaluated model before and after dataset balancing strategy	None	To ensure the impacts using dataset balancing techniques on the deep learning models' performance
9	Focused on research publications from 2020 to 2024	None	To see the current DDoS attack detection practices

of records that provide a diverse representation of DDoS attack traffic [36]. This dataset was selected due to its comprehensiveness, well-documented features, and recent development. Its widespread use in other studies allows for valid comparisons with existing research. CICDDoS2019 dataset incorporates the varied types of DDoS attacks categorized into reflection and exploitation [36]. From the exploitation attacks, the study used SYN, UDP-lag, and UDP and from the reflection attacks used TFTP, NTP, MSSQL, and LDAP [22, 36].

Application and transport layer protocols like UDP, TCP, or a combination of the two are used in reflection-based DDoS attacks. While TCP-based protocols include SSDP and MSSQL attacks, UDP-based DDoS attacks include TFTP, NTP, and CharGen. DNS, LDAP, NetBIOS, and SNMP may all be attacked using a combination of TCP and UDP. While UDP-based exploits include UDP-lag and UDP flood, TCP SYN flood is a TCP-based exploitation attempt. In order to create a three-way handshake connection, attackers use TCP SYN flooding to interfere with the victim's authentic connection. Attackers launch a TCP SYN attack by bombarding a server with SYN packets and then failing to reply to the server's response. Sending a large quantity of UDP packets to the remote computer initiates a UDP flood attack. These UDP packets are sent quickly to the target system's random ports. As a result, the system crashes, performance declines, and the network's available capacity is exhausted [2, 27]. An attacker takes advantage of flaws in Microsoft Structured Query Language (MSSQL) by posing as a genuine MSSQL client and sending scripted requests to the MSSQL server using a spoofed IP address, making it look as though they were coming from the target server. DDoS attack uses the Lightweight Directory Access Protocol (LDAP), an application layer protocol that is commonly used to retrieve a human-readable URL (such as google.com), to send queries to a publicly accessible but insecure LDAP server in order to produce a huge number of answers [2]. The detailed descriptions of the dataset features, that is, CICDDoS2019, could be accessed from the recent study by Cil et al. [6].

3.2. Data preprocessing techniques

The CICDDoS2019 dataset is very large, making it challenging to find the necessary hardware for processing such large quantities of data, especially in resource-constrained devices in African countries in general and Ethiopia in particular. It requires GPU-enabled hardware to handle this volume effectively. In this respect, the study concentrated on eight specific classes of DDoS attacks to conduct experiments. These are SYN, UDP, MSSQL, UDP-lag, TFTP, NTP, LDAP, and benign classes, as assessed by their importance for DDoS attack detection in existing studies. The study removed the duplicates using the Python function `drop_duplicates` for each class, which may affect the performance of the models. After removing the duplicated records from each class, the study concatenated the data into a single file, facilitating easier manipulation and analysis. After concatenation, the dataset comprised 797,128 samples. During this process, the study identified and removed certain entries from the dataset that were deemed irrelevant or potentially harmful to model performance.

To mitigate the model overfitting issues, the study opted to remove socket-related features like Source Port, Timestamp, Destination IP, Flow ID, Destination Port, and Source IP. These attributes can exhibit significant variability across different network configurations, which can complicate the training process [2]. Overfitting issues can occur when training the deep learning model by incorporating socket feature values due to the attackers and legitimate users sharing the same IP address [6, 8, 50].

Due to their lack of variability, we removed features such as Bwd Packet Length Std, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, Bwd Avg Bytes/Bulk, ECE Flag Count, Fwd Avg Packets/Bulk, Fwd Avg Bytes/Bulk, FIN Flag Count, PSH Flag Count, Fwd Avg Bulk Rate, Bwd Avg Packets/Bulk, Unnamed, Bwd Avg Bulk Rate, and Similar HTTP [6]. After excluding the zero variance features, the updated dataset consists of 67 features.

One-hot encoding was used to accurately represent the categorical data [16]; StandardScaler was used to transform each feature to have 0 mean and 1 standard deviation for standardizing the features with the highest variance identified in the dataset [37].

3.3. Dataset balancing strategy

Most (13 out of 24) studies used Imbalanced datasets to train deep learning models. This approach could result in biased model outputs. To address these concerns, the SMOTEENN technique combines the advantages of oversampling and data cleaning to better handle imbalanced datasets. SMOTE creates fresh samples for the minority class in the first stage. This method reduces the risk of overfitting that can occur with the naïve replication of uncommon samples and helps address the problem of having too few training cases in the minority class. However, because misaligned synthetic points may obscure class distinctions, including freshly generated data in noisy regions or close to class determination boundaries might occasionally reduce model performance. SMOTEENN follows a hybrid approach to overcome class imbalance. Thus, SMOTE corrects unequal class distributions by boosting the representation of the minority class, while ENN acts as

a cleaning mechanism, guaranteeing that the final training set is more coherent and less prone to noise. By eliminating problematic points, ENN protects the quality of the data, which makes SMOTEENN an effective method for enhancing classification performance on highly skewed datasets [20, 38–40].

Compared to dataset balancing strategies like the synthetic minority oversampling technique (SMOTE), ADASYN, RUS (random undersampling), and ROS, SMOTEENN is a new resampling technique that uses Map-Reduce architecture to combine intelligent undersampling and oversampling and show outperforming results for small- and medium-sized datasets while achieving positive results on large datasets with reduced running times. One useful technique for lowering the overall complexity of the learning process is undersampling. Nonetheless, several studies have demonstrated that hybrid undersampling/oversampling techniques like SMOTEENN that incorporate SMOTE oversampling and ENN undersampling might help raise the predictive capabilities of classifiers [38]. Table 3 illustrates the imbalanced and balanced dataset distributions across multi-class.

3.4. Implemented deep learning algorithms

The study employed seven popular deep learning algorithms, namely, MLP, LSTM, BiLSTM, DNN, deep convolutional neural network (DCNN), CNN-LSTM, and DNN with autoencoder. These deep learning algorithms were chosen due to their ability to leverage various strengths for managing complex, high-dimensional, and sequential data. Each algorithm was trained before and after employing the SMOTEENN dataset balancing techniques for performance comparison. Descriptions of each deep learning algorithm are presented as follows.

3.4.1. Multilayer perceptron (MLP)

MLP is a type of feed-forward neural network; it can handle both linear and nonlinear data, widely used and are trained using backpropagation, and is a supervised learning method [41, 42]. It was applied in numerous domains, such as speech recognition, image compression, financial data prediction, autonomous vehicle control, medical diagnosis, and phishing website detection. MLP supports a nonlinear activation function [43, 44].

As stated in study by Najar and Manohar Naik [5], MLP uses the nonlinear backpropagation algorithm to compute all input neuron's standard sum as shown in Equation (1).

$$\frac{1}{1+e^{-\sum_g WjXj+b}} \tag{1}$$

An example of a common MLP is shown in Equation (1), where **b** is the bias, **Xj** is the neuron input, and **Wj** is the input weight. With a specific number of neurons in each layer, the network is completely connected. The characteristics of the data collection and the quantity of classes determine the size of the incoming and outgoing neurons. m outputs are produced by the model for every M-class categorization. Using the input sequence $Y = (Y_1; Y_2; Y_3; ...; Y_d)^T$ as guide Class g's production is determined by:

$$Y_g(X) = S\left[\sum_{I=1}^{hn}(W_{gi}S(Vi^TX + V_{io})W_{go}\right) \tag{2}$$

$$S\left[\sum_{I=1}^{hn}(W_{gi}h_i + W_{go}\right)$$

hn represents the number of hidden layers, and $\mathbf{W_{gi}}$ and $\mathbf{V_i}$ represent the output and hidden layers' connection weights, respectively. The values for every connection are automatically determined when the MLP classifier is trained on a testing dataset [5].

The MLP scored an accuracy of 99.13% in the study [5], even though the study used binary classifications and imbalanced datasets, and it shows a quick computational time in identifying phishing websites [43]. In our study, the MLP model achieved the highest accuracy (98.9%) in DDoS attack detection utilizing a balanced dataset for multi-class classifications.

3.4.2. Deep neural network (DNN)

DNN mimics the works of the human brain. It is analogous to the traditional MLP algorithm despite DNN having more hidden layers. Increasing the amount of input features and the size of the model hyperparameter can affect the DNN model performance in terms of computational speed. The DNN uses the backpropagation for the learning process [43].

Three steps are involved in the DNN algorithm's model construction. Initially, the number of layers and the neurons for each layer are determined by the model topology along with the relationships between them. Then, using artificial neurons, the forward propagation with its activation function and perceptron classifier is employed. Finally, backpropagation is used in conjunction with the optimizer and loss function [49]. DNN formula is represented in Equation (3)

$$X = \left(\sum_{I=1}^{cn}(H_iW_i + B)\right) \tag{3}$$

where **H** stands for these nodes' values, **cn** for the number of nodes in the layer, **W** for the weights, and **B** for the nodes' bias in Equation (3). Activation functions receive the results. The notion of the activation function is generally based on how a neuron functions in the human brain and is explicitly activated at a certain level, known as the activation potential. It puts the outcome inside a certain range. Between the typical activation functions, which include sigmoid, tanh, softmax, and ReLU, this system used ReLU activation functions in the hidden layers [45].

Table 3
Dataset distributions before and after using the SMOTEENN technique

Dataset	Traffic type	No. of entries before SMOTEENN	No. of entries after SMOTEENN
CICDDoS2019	TFTP	153175	144267
CICDDoS2019	SYN	143725	105477
CICDDoS2019	DrDDoS_LDAP	17265	142932
CICDDoS2019	DrDDoS_UDP	151498	148461
CICDDoS2019	UDP-lag	61370	99213
CICDDoS2019	DrDDoS_NTP	151758	146860
CICDDoS2019	BENIGN	49143	151035
CICDDoS2019	DrDDoS_MSSQL	69194	144846
		Total = 797128	Total = 1083091

The accuracy of the DNN model in our investigation was 98.73%, which is 0.1% better than that of the LSTM model, 4.16% better than the accuracy of the LSTM model in the study by Cil et al. [6], while 0.93% lower accuracy attained by Amaizu et al. [22]. Despite the DNN model train using Imbalanced datasets, it attains 99% accuracy in the study by Asad et al. [7] as well as Sbai and Elboukhari [23]. To attain 98.73% accuracy, our study uses a balanced dataset; the initial three hidden layers of DNN consist of 256, 128, and 128 units, respectively, while the subsequent layers contain 96, 48, 56, and 32 units, with a dropout rate of 0.2. The model utilized the Adam solver with categorical cross-entropy loss function and accuracy as metrics.

3.4.3. Long short-term memory (LSTM)

LSTM is a type of RNN that addresses the vanishing gradient problems, which can occur when training deep networks over long sequences. Its architecture includes an input gate, output gate, and forget gate. These components maintain the flow of signals between the layers, enabling long-term learning dependencies [39, 40]. It is often used in applications like language modeling, intrusion detection, and sentiment analysis [39, 40].

As indicated in the study by Zainudin et al. [27], memory cells are neurons found in the LSTM layer. Three gates are present in every memory cell: input, forget, and output gates. Every gate processes the input features in a different way. Depending on the cell's state, the forget gate, for example, chooses which data should be processed by eliminating irrelevant data. To remove extraneous data, the forget gate first passes the previous cell output Fg−1 and the addition layer output Fg via a sigmoid gate σ. Lastly, the acquired data are multiplied and combined.

The output of the forget gate ft is shown in Equation (4) [27]

$$f_g = \sigma\left(W_f . [f_{g-1}, f_g]^T + b_f\right) \tag{4}$$

where $\mathbf{W}_f$ and $\mathbf{b}_f$ depict the weight matrix and the forget gate offset, respectively.

Consequently, the input gate uses a sigmoid function for the input data control and generating the current state. The processing of the input gate can be shown in Equation (5) [27]

$$i_g = \sigma\left(W_i . [f_{g-1}, f_g]^T + b_i\right) \tag{5}$$

where $\mathbf{W}_i$ and $\mathbf{b}_i$ depict the weight matrix and the forget gate offset, respectively.

Additionally, the input gate creates a vector of the data to be added to the current state using a tanh function. The network calculates the hidden state cg in the following manner using the outputs of the input and forget gates [27] (see Equations (6) and (7)):

$$\mathrm{cg} = tahn\left(W_c . [f_{g-1}, f_g]^T + b_c\right) \tag{6}$$

$$\mathrm{cg} = f_g . \mathrm{cg} - 1 + i_g . \mathrm{cg}\ \ tahn\left(W_c . [f_{g-1}, f_g]^T + b_c\right) \tag{7}$$

Lastly, the output gate chooses helpful characteristics depending on the cell's current state, the preceding cell's outcome, and new data. An expression for this gate's output $\mathbf{o}_g$ function can be shown in Equation (8):

$$o_g = \sigma\left(W_o . [f_{g-1}, f_g]^T + b_o\right) \tag{8}$$

The LSTM layer output (**Fout**) can be shown in Equation (9):

$$\mathrm{Fout} = og . tahn(c_g) \tag{9}$$

The dense layer receives input from the output LSTM layer. Softmax and fully linked layers process the output data after the dense layer in order to categorize each attack type [27].

LSTM scored accuracies of 99.9% [10], 98% [11], and 99.19% [24], even though the study by Khempetch and Wuttidittachotti [10] used Imbalanced dataset, the study by Kumar et al. [11] didn't mention train-test splits used and didn't employ multi-class classification, and the study by Shurman et al. [24] didn't mention dataset size and features. Despite being 0.63% more accurate than the accuracy achieved by the study [11], LSTM scored the lowest accuracy in our analysis (98.63%) compared to MLP, BiLSTM, CNN-LSTM, DNN, DNN-AE, and DCNN models.

3.4.4. Bidirectional long short-term memory (BiLSTM)

BiLSTM networks are particularly effective for handling numerical data, especially in applications like time series forecasting and other sequential tasks. These networks are designed to capture dependencies in both directions, which is especially beneficial for time series data where past and future values can impact predictions [46]. In BiLSTM architecture, two LSTM units are employed, with one processing the input in a forward direction and the other in reverse. This dual approach enhances the model's ability to understand context and improve prediction accuracy. This method differs from the unidirectional LSTM approach by combining sequences from both directions to form a unified representation. This integration allows the model to utilize information from both LSTMs, enhancing its ability to forecast the next temporal output based on the combined results of the two LSTMs [47].

BiLSTM is an improved version of LSTM that combines backward and forward LSTM to analyze traffic data more precisely. This enables a two-way information analysis process that enables more accurate computations. This is how the computation is carried out [48] (see Equations (10), (11), and (12)).

$$K_{tforward} = f(W_{forward} x I_t + W x \mathrm{K}_{t-1forward} + B_{forward}) \tag{10}$$

$$\mathrm{K_{tbackward}} = \mathrm{f(W_{backward} x I_t + W_{backward} x K_{t-1backward} + B_{backward})} \tag{11}$$

$$O_t = g(Ux[K_{tforward}, K_{t-1backward}] + c) \tag{12}$$

$\boldsymbol{K_{tforward}}$ and $\boldsymbol{K_{t-1backward}}$ are the two LSTM layer outputs at time **t** in forward and backward directions, respectively. $\boldsymbol{W_{forward}}$ and $\boldsymbol{W_{backward}}$ are the parameters of hidden layer, $\mathbf{I}_t$ is the input data, $\boldsymbol{B_{forward}}$ and $\boldsymbol{B_{backward}}$ are the offset values, and $\boldsymbol{O}_t$ is the output of the BiLSTM. Generic BiLSTM formula includes the forget gate, the input gate, the previous cell state, the current cell state, the output gate, the previous unit output, the current unit output, the weight of neural network, and the bias value [48].

In order to better comprehend the context and increase the prediction accuracy, our study used the BiLSTM model, which has advantages over LSTM in terms of capturing dependencies in both directions (past and future values) [46, 47]. The BiLSTM model in our investigation detected DDoS attacks with a third-best accuracy (98.8%), behind the MLP and CNN-LSTM combo accuracies, respectively. This accuracy is 4.8% greater than the accuracy achieved in the study by Nguyen et al. [21] by BiLSTM and 0.17% higher than the accuracy of the LSTM model in our study.

To attain the stated accuracy, the first layer of the BiLSTM model contains 128 units, which process input sequences; the next layer contains 64 units to enhance the model's capacity to capture complex temporal patterns with a dropout rate of 0.1. The model also featured a dense layer with 128 units and a ReLU function. To further mitigate overfitting, a second dense layer contains 64 units with a dropout rate of 0.1. Finally, the architecture concluded with a dense output layer

that utilized the softmax activation function to generate probabilistic outputs for multiple classes. The architecture was set up with the Adam solver and a learning rate of 0.00001. For the loss function, categorical cross-entropy was employed, making it appropriate for tasks involving multiple classes.

3.4.5. Deep convolutional neural networks (DCNN)

The CNN has many layers, including input, convolutional, pooling, fully connected, and output layers; the number of layers used determines how deep the CNN is. The general system used by convolutional neural networks is comparable to that of multilayer perceptron. It is designed similarly to a multilayer perceptron. DCNN is capable of automatically learning hierarchical structures and identifying spatial relationships, which makes them well suited for analyzing and understanding structured grid-like data [30].

As indicated in the study by Jamal et al. [17], CNNs encompasses many hierarchical levels and computational procedures which are represented in Equation (13).

$$\text{Output} = Activation(\sum \text{k} = 1^{n}(W_k * Input_k) + b \quad (13)$$

where

1. ***Output*** depicts the features yielded by the layer.
2. ***Activation*** is the function used elementwise.
3. ***n*** is the number of kernels (filters) in the layer.
4. W_k depicts the kth filter's weight.
5. $Input_k$ is the kth input feature.
6. ***b*** depicts the bias term.
7. $*$ indicates the convolution process between the filter and input feature.

In our study, DCNN attains 98.75% accuracy in DDoS attack detection which is better than LSTM, DNN, and DNN-AE accuracy. For DCNNs, we set up a sequential model, a popular way to build neural networks in Keras. The initial 1D CNN utilized 64 filters with a 3x3 kernel size, serving as feature detectors with the ReLU function. The next layer contains 128 filters with a 3x3 kernel size. Max pooling layers were used twice to diminish the data's dimensionality. Max pooling takes the maximum value within windows of size 2, summarizing the most vital variables in a smaller representation. The flattening layer then transformed the multidimensional output from the convolutional layer into single vectors. This step was essential for connecting the data to fully connected layers. We defined two dense layers with 64 and 128 units, respectively, each monitored by a ReLU function to model complex relationships between the features more effectively.

3.4.6. CNN-LSTM

CNN and LSTM were combined to form the hybrid technique in our study. The CNN was served as a feature extractor to feed or provide important features for the LSTM model at its pooling layer. LSTM is a type of RNN that addresses the vanishing gradient problems, which can occur when training deep networks over long sequences or enabling long-term learning dependencies [39, 40].

Because it performs the best in identifying DDoS attacks among the examined studies, with accuracies of 97.16% [33], 99% [34], and 99.5% [27], the CNN-LSTM was selected for our experiment even though the study by Roopak et al. [33] did not include dataset size, number of features, train-test splits, and fine-tuned model hyperparameters, the study by Nugraha and Murthy [34] employed imbalanced datasets and binary classification, and the study by Roopak et al. [27] used binary classification and didn't reveal train-test split ratios; these practices are against the replication or revalidation of the research findings.

In our study, the CNN-LSTM combo achieved the second better accuracy (98.82%) next to the MLP model in DDoS attack detection utilizing balanced dataset and multi-class classifications by fine-tuning the CNN-LSTM model using activation function (ReLU and softmax), hidden layer sizes, optimizer or solver (Adam), batch size, and 3x3 kernel size.

3.4.7. DNN with autoencoder (DNN-AE)

DNN and AE were combined to form the hybrid technique in our study. AE is frequently employed for dimensionality reduction and feature extraction [2]. AE is a feed-forward neural network with one or more hidden layers. These networks may be trained sequentially, layer by layer, reducing the computational resources necessary to develop an effective model [35]. It is a specific kind of artificial neural network primarily utilized for unsupervised learning tasks, especially in deep learning. Its main function is to learn a compact representation of input data by encoding it into a lower-dimensional latent space and subsequently reconstructing it back to its original form. These networks are specially trained to recreate their input. This approach does not require labeled data for training because it functions in a self-supervised fashion. An encoder and a decoder are the two primary parts of an autoencoder. To minimize the reconstruction error—the discrepancy between the original input data and the reconstructed output—the encoder must convert the input data into a lower-dimensional representation. The decoder then uses this compressed representation. This procedure motivates the autoencoder to create effective and educational data representations. One of the key benefits of using this model is its ability to identify significant features of the input by compelling it to learn the essential aspects of the data during the training phase. Dimensionality reduction occurs when fewer nodes exist in each hidden layer of the model [49]. A DNN has multiple hidden layers situated between the input and output layers. The hidden layers enable the network to capture increasingly complex representations of the input data, allowing it to tackle tasks that are more sophisticated [43].

Our study's DNN-AE combo achieves 98.72% accuracy, surpassing the LSTM model's 98.69% accuracy. To attain this result, the DNN-AE model included several dense layers with ReLU activation functions. The initial dense layer comprised 256 neurons, the ReLU function, and the dropout layer with a rate of 0.2 for regularization. The subsequent dense layer contained 128 neurons, also using ReLU activation, followed by another dropout layer with a rate of 0.2. The third dense layer had 64 neurons with ReLU activation, again followed by a dropout layer at a rate of 0.2.

The values of the hyperparameters directly affect the behavior of the trained model, which underscores the importance of choosing optimal settings. Selecting the best hyperparameter values relies on established best practices and human expertise [43]. According to the study by Sindian [49], while underfitting can occur when a model has few parameters and overfitting can occur when a model has many, increasing the depth of a model can also result in a gradient disappearing or gradient explosion. Therefore, it takes a lot of trial and error and careful consideration to create a model that precisely matches the computer resource limitation without sacrificing classification performance. In this respect, our study conducted multiple experiments to fine-tune models for better performance using parameters including the batch size, learning rate, number of layers, filter sizes, and number of neurons. The carefully selected model hyperparameter values are presented in Table 4.

3.5. Model evaluation and validation methods

The dataset is partitioned into multiple subsets to ensure the learning model's generalization capability on unseen data [37, 44]. In this regard, the study utilized various split ratios, including 60/20/20, 70/20/10, and 80/10/10 for training, testing, and validation,

Table 4
Selection of suitable hyperparameters for different deep learning models

Models	Parameter	Learning rate	Batch size	Optimizer	Activation function	Kernel size	Test accuracy (%)
MLP	Values	0.00001	128	Adam	ReLU, softmax	–	98.9
DNN	Values	0.00001	128	Adam	ReLU, softmax	–	98.73
LSTM	Values	0.00001	128	Adam	ReLU, softmax	–	98.69
BiLSTM	Values	0.00001	128	Adam	ReLU, softmax	–	98.8
DCNN	Values	0.00001	128	Adam	ReLU, softmax	–	98.75
CNN-LSTM	Values	0.00001	128	Adam	ReLU, softmax	3x3	98.82
DNN-AE	Values	0.00001 for DNN while 0.0001 for AE	128 for DNN & 64 for AE	Adam	ReLU & softmax for DNN while ReLU & MSE for AE	–	98.72

respectively. These ratios were carefully selected to balance the availability of training data with the necessity of reserving sufficient data for model validation and testing. The effectiveness of classification methods relies not only on the techniques employed but also on how the training and testing data is divided [50]. The appropriate split ratio may also be contingent on the size of the dataset; larger datasets might allow for smaller validation and test sets, while smaller datasets typically necessitate larger proportions to ensure a robust evaluation. Most studies didn't compare different train-test splits. Additionally, introducing randomness into the data-splitting process is essential for guaranteeing that different experimental runs produce varied datasets, which in turn enhances the model's robustness. A random seed is used for reproducibility of experiments, ensuring consistency across runs. The study employed diverse model evaluation metrics like precision, recall, accuracy, and F1 scores.

For code editing, Jupyter Notebook and Spyder were preferred due to their user-friendly interfaces and powerful functionalities, which are particularly suited for Python programming and data science tasks. The study used cloud-based tools to enhance computational efficiency, notably Google Colab, which offered additional graphics processing unit (GPU) resources during the deep learning model training. For data analysis, manipulation, visualizations, balancing, and model implementations, diverse Python libraries were used, namely, Pandas, NumPy, Scikit-learn, TensorFlow, Matplotlib, and Imblearn [43].

To conduct experiments, the study used the following hardware and software as illustrated in Table 5.

4. Experimental Results

This section covers an experimental result on different train-test splits, comparison of deep learning models' performance with and without SMOTEENN dataset balancing strategy, the learning curve of the top-performing model, and workflow of how the proposed model works in detection, classification, and mitigation of DDoS attacks.

Table 5
Experimental setups for implementing deep learning models

No.	Aspects	Details
1	Programming language	Python 3.9
2	Environment	Google Colab environment
3	Client operating system	Windows 10
3	RAM	4GB
4	Processors	Intel Core i3, operating at 2.4 GHz
5	Hard disk	500 GB
6	GPU	T4 provided free Google Colab

4.1. Experiment 1: assessing deep learning models' performance across different dataset train, validation, and test splits

Seven distinct deep learning models were utilized to detect DDoS attacks in the context of multi-class classification. These models include MLP, DNN, DCNN, LSTM, BiLSTM, CNN-LSTM, and DNN-AE. To enhance each model's performance, different tasks were carried out, such as feature scaling, encoding, data cleaning, dataset balancing, model fine-tuning, and partitioning the dataset into training, testing, and validation splits. Various data partitioning strategies were implemented on the proposed model, including splits of 80/10/10, 70/20/10, and 60/20/20, to ensure model capability on unobserved data. Notably, the 70/20/10 split consistently demonstrated optimal performance in the context of multi-class classification as illustrated in Table 6 and Figure 1.

4.2. Experiment 2: assessing deep learning models' performance with and without the SMOTEENN dataset balancing strategy

As depicted in Figure 2, the MLP model outperformed the others, with 98.90% accuracy and a 0.69% improvement over its pre-SMOTEENN results. MLP accuracy (98.90%) was 0.1% better than the third-best accuracy (98.8%) achieved by BiLSTM and marginally (0.08%) higher than the second-best accuracy (98.82%) achieved by CNN-LSTM combo.

Although SMOTEENN technique significantly improved the

Table 6
Performance of the models across different dataset split ratios

	Dataset split ratios (train/test/validation)		
Models	80/10/10 (%)	70/20/10 (%)	60/20/20 (%)
DNN	98.73	98.73	98.71
DNN-AE	98.55	98.72	98.57
CNN-LSTM	98.75	98.80	98.79
LSTM	98.61	98.63	98.58
DCNN	98.71	98.75	98.67
BiLSTM	98.79	98.80	98.76
MLP	98.85	98.90	98.83

Figure 1
Accuracy-based model comparisons across different data splits

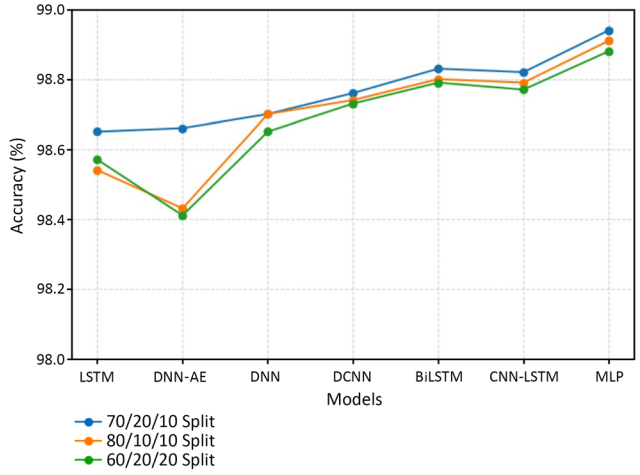
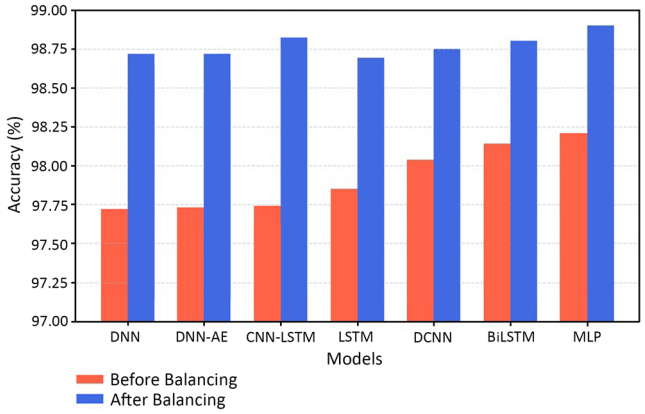


Figure 2
Testing accuracy of the models with and without the SMOTEENN dataset balancing strategy



accuracy of each deep learning model, it did not reduce the train-test computational times for each model, as Table 7 illustrates. The MLP model's train-test computational time was delayed by 802.15 seconds, the LSTM model by 737.66 seconds, the DNN model by 747.66 seconds, the BiLSTM model by 728.63 seconds, and the DCNN model by 797.11 seconds after the SMOTEENN application. The CNN-LSTM and DNN-AE models were delayed by 849.5 and 857.69 seconds, respectively. After applying the SMOTEENN technique, the experimental results show that BiLSTM is comparatively the quickest DDoS attack detection model, whereas CNN-LSTM and DNN-AE models are the slowest when compared to MLP, LSTM, and DCNN models. From these experimental results, it is possible to conclude that because they are hybrid models or combine the computational times of two models, CNN-LSTM and DNN-AE are the slowest models. According to the study by Zainudin et al. [27], applications of relevant feature selection strategies on CICDDoS2019 dataset was contributed to reduce machine learning and deep learning model's computational time.

The model is said to be successful when lowering the false negative rate (FNR) and false positive rate (FPR). In this study, FPR indicates that benign traffic was mistakenly classified as DDoS attacks, whereas FNR indicates that DDoS attack traffic was mistakenly classified as benign traffic. Normal network operations are disrupted by both FPR and FNR. In this context, the SMOTEENN technique reduces the FNR of the MLP model by 0.57% while slightly increasing the FPR of the MLP model by 0.13%; reduces the FNR of the LSTM model by 0.48% while slightly increasing the FPR of the LSTM model by 0.12%; reduces the FNR and FPR of the DNN model by 0.51% and 0.52%, respectively; reduces the FNR and FPR of the BiLSTM model by 0.34% and 0.37%, respectively; reduces the FNR of the DCNN model by 0.32% while slightly increasing the FPR of the DCNN model by 0.08%; reduces the FNR of the CNN-LSTM model by 0.33% while slightly increasing the FPR of the CNN-LSTM model by 0.08%; and reduces the FNR of the DNN-AE model by 0.35% while slightly increasing the FPR of the DNN-AE model by 0.08%. According to these experimental results, the SMOTEENN technique reduces the FNR and FPR for models like DNN and BiLSTM. While slightly increasing the FPR, the SMOTEENN technique lowers FNR for the majority of models, including MLP, LSTM, DCNN, CNN-LSTM, and DNN-AE.

Table 7
Comparative model performance analysis with and without SMOTEENN technique

No.	Model name	With or without SMOTEENN	Accuracy (%)	FPR(%)	FNR (%)	Train-test computational time in seconds
1	MLP	Without	98.21	1.15	1.90	7219.39
		With	98.9	1.28	1.37	8021.54
2	LSTM	Without	97.85	1.25	1.85	6638.93
		With	98.69	1.37	1.37	7376.59
3	DNN	Without	97.72	1.80	1.80	6728.96
		With	98.73	1.28	1.29	7476.62
4	BiLSTM	Without	98.14	1.50	1.50	6557.61
		With	98.8	1.13	1.16	7286.24
5	DCNN	Without	98.04	1.05	1.45	7173.95
		With	98.75	1.13	1.13	7971.06
6	CNN-LSTM	Without	97.74	1.07	1.47	7645.46
		With	98.82	1.14	1.14	8494.96
7	DNN-AE	Without	97.73	1.04	1.48	7719.27
		With	98.72	1.12	1.13	8576.96

As illustrated in Figure 3, the MLP model outperforms the remaining deep learning models such as DNN, DCNN, BiLSTM, LSTM, CNN-LSTM, and DNN-AE in terms of accuracy, precision, recall, and F1 score. For the MLP model, the study constructed the neural network model using the Keras Sequential API. It featured multiple densely connected layers with ReLU activation functions. The architecture began with an input dense layer of 1024 neurons, followed by dropout regularization (rate of 0.1) and batch normalization. The subsequent dense layers had 2024, 2000, 1000, 500, and 200 neurons, each with batch normalization and dropout. The output layer contained eight neurons and an activation function softmax. Training the MLP model involves Adam optimizer, 0.00001 learning rate value, categorical cross-entropy loss, and accuracy metrics, 50 epochs, and 128 batch sizes. It achieved an accuracy of 98.90% and showed consistent performance with precision, F1 score, and recall at 99%.

Classification reports, shown in Table 8, revealed no bias across classes, providing valuable insights into dataset fairness.

Figure 4 shows the accuracy and loss curves of MLP on multi-class classification.

Figure 3
Models' performance results using accuracy, precision, recall, and F1 score

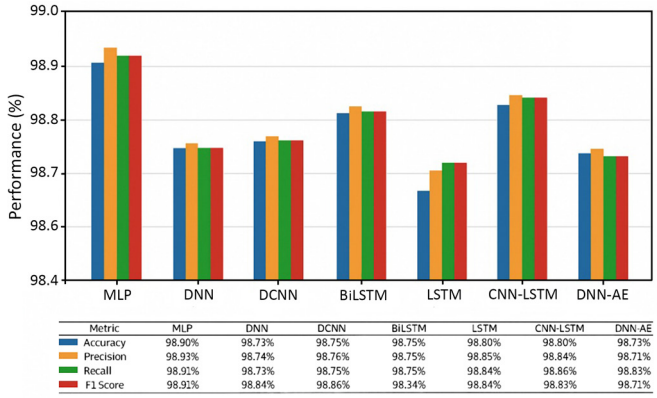


Table 8
MLP model classification results with SMOTEENN technique

Classes	Precision	Recall	F1 score	Support
Benign	1.00	1.00	1.00	30,431
DrDoS_LDAP	0.95	0.98	0.97	28,822
DrDDoS_MSSQL	0.98	0.95	0.96	28,709
DrDDoS_NTP	1.00	1.00	1.00	29,447
DrDDoS_UDP	1.00	1.00	1.00	29,654
Syn	1.00	1.00	1.00	21,086
TFTP	1.00	1.00	1.00	28,801
UDP-lag	0.99	1.00	1.00	19,669
Accuracy			99	216619
Macro Avg	99	99	99	216619
Weighted Avg	99	99	99	216619

4.3. Experiment 3: demonstrating DDoS attack detection and classification using sample workflow

In order to show how the top-performing deep learning model might function in a real-world application, the study tried to demonstrate the workflow for DDoS attack detection, classification, and mitigation. Integrating MLP model into network security devices for DDoS attack detection and classification is a robust approach. Once a DDoS attack is detected and classified, a mitigation strategy can be implemented. Figure 5 illustrates a simple flowchart outlining the steps to follow for the mitigation strategies of identified DDoS attacks in a network-computing environment.

The DDoS detection and classification model is trained to classify incoming network traffic data based on a comprehensive set of 67 features. Users can upload a CSV file, enabling predictions for multi-class classifications, that is, it helps users effectively identify and categorize data into one of eight target classes. For binary classification, it assists users by classifying network traffic as either benign or DDoS. This functionality may enhance the security and reliability of network systems. The MSC thesis served as the source of the manuscript's contents, including the workflow for DDoS attack detection, classification, and mitigation of proposed model illustrated in Figure 5. Both the thesis supervisors and the examining committees evaluated and gave encouraging comments about the workflow during the MSC thesis defense.

5. Discussions on Key Research Findings

The study rigorously reviewed recent research works on detecting DDoS attacks utilizing deep learning approaches, specifically from 2019 to 2024, and used structured guidelines to locate current research gaps and suggest viable solutions. These research works were retrieved using the PRISMA guidelines from reputed academic databases. The dataset used for the study was the CICDDoS2019, provided by

Figure 4
Accuracy and loss curves of the MLP model and SMOTEENN combo

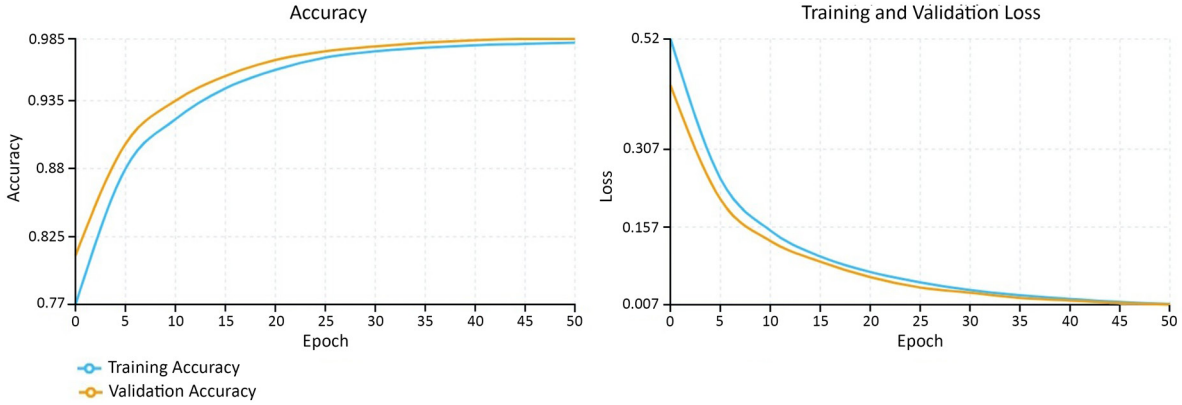
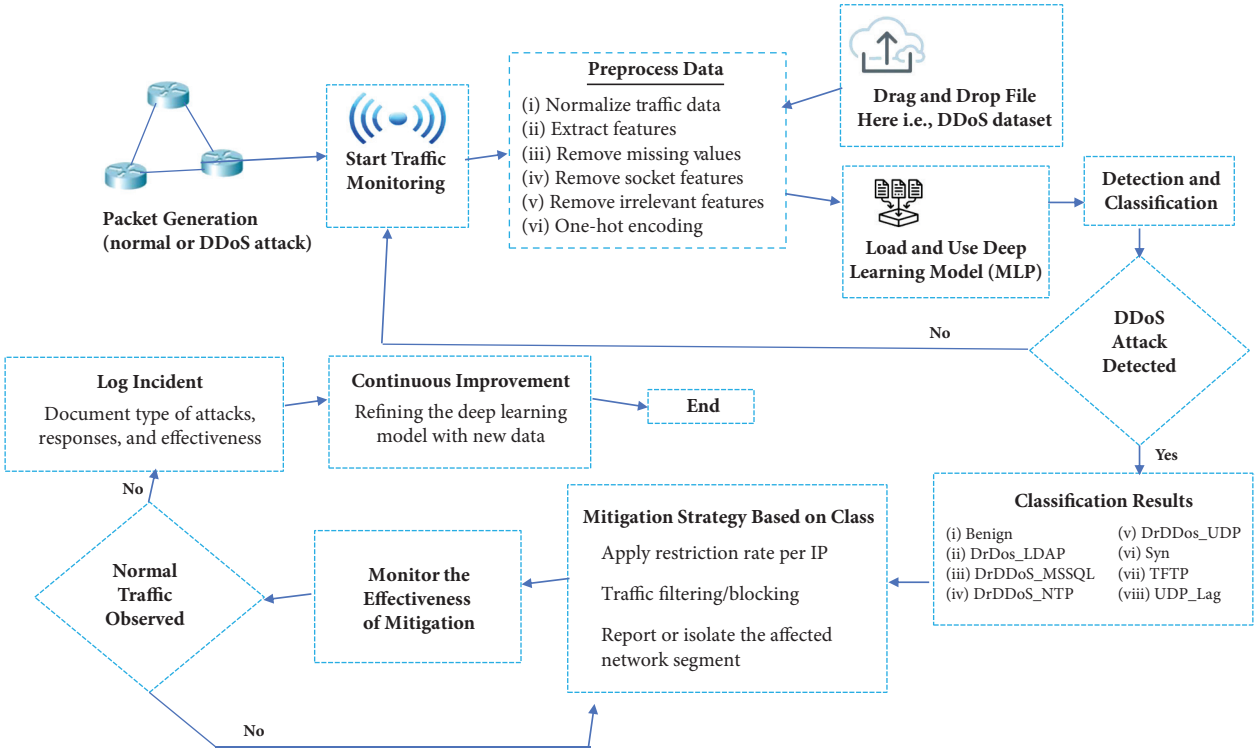


Figure 5
Workflow for DDoS attack detection, classification, and mitigation of the proposed model



the Canadian Institute for Cybersecurity (CIC). It was selected due to its recent release and importance within the intrusion detection system (IDS) field. Its widespread use in other studies allows for valid comparisons with existing research. CICDDoS2019 dataset consists of the various types of DDoS attacks categorized into reflection and exploitation-based attacks [36]. The study included exploitation-based attacks, such as SYN, UDP-lag, and UDP, and reflection-based attacks, such as TFTP, NTP, MSSQL, and LDAP [22, 36].

Seven popular deep learning models were utilized to detect DDoS attacks in the context of multi-class classification. These models include MLP, DNN, DCNN, LSTM, BiLSTM, CNN-LSTM, and DNN-AE. To enhance each model's performance, different tasks were carried out, such as feature scaling, encoding, data cleaning, dataset balancing, model fine-tuning, and partitioning the dataset into training, testing, and validation splits. Various data partitioning strategies were implemented on the proposed model, including splits of 80/10/10, 70/20/10, and 60/20/20, to ensure model capability on unobserved data. Notably, the 70/20/10 split consistently demonstrated optimal performance in the context of multi-class classification.

Although SMOTEENN technique significantly improved the accuracy of each deep learning model, it did not reduce the train-test computational times for each model. Even though MLP and CNN-LSTM achieved the first and second accuracy, 98.9% and 98.82%, respectively, the experimental results show that BiLSTM is comparatively the fastest DDoS attack detection model, with the third-best accuracy of 98.8%, while CNN-LSTM and DNN-AE models are the slowest when compared to MLP, LSTM, and DCNN models. Because CNN-LSTM and DNN-AE are hybrid models or combine the computational times of two models, they become slower. Reducing the computational time of deep learning models requires the application of relevant feature selection strategies [27]. Despite being 0.63% more accurate than the accuracy achieved by Kumar et al. [11], LSTM achieved the lowest accuracy in our analysis (98.63%) when compared to MLP, BiLSTM, CNN-LSTM, DNN, DNN-AE, and DCNN.

The model is said to be successful when lowering the false negative rate (FNR) and false positive rate (FPR). In this study, FPR indicates that benign traffic was mistakenly classified as DDoS attacks, whereas FNR indicates that DDoS attack traffic was mistakenly classified as benign traffic. According to our experimental results, the SMOTEENN technique reduces the FNR and FPR for models like DNN and BiLSTM. While slightly increasing the FPR, the SMOTEENN technique lowers FNR for the majority of models, including MLP, LSTM, DCNN, CNN-LSTM, and DNN-AE.

Numerous studies have been conducted in this field. The review findings reveal that most studies did not reveal each experimental detail for the reproducibility of research findings. Specifically, most studies used imbalanced datasets and did not reveal dataset size, fine-tuned model hyperparameters, and train-test-validation split ratios. For instance, references [1, 8, 21–27] used the same dataset CICDDoS2019 with our study despite references [10, 11, 21, 24, 28] not revealing dataset size, references [11, 23, 25, 27–29] not revealing train-test-validation split ratios, references [21, 24, 29] not revealing number of top predictive features used, and references [25, 26, 28] not revealing fine-tuned model hyperparameters. The study aims to address these issues. When assessing the effectiveness of various models, the proposed deep learning approaches consistently outperform many existing methods. This success can be attributed to the careful selection of methodologies, including data balancing techniques, normalization techniques, missing values, removing irrelevant features, encoding techniques for converting categorical values to numerical values, and model optimization techniques. Based on extensive research, this study presents a comparative analysis of model performance alongside the most relevant preceding studies, highlighting the advancements in DDoS attack detection and classification through deep learning approaches as illustrated in Table 9.

Table 9
Comparative performance analysis with the related works on CICDDoS2019 dataset

Related study	Top-performing model	Domains	Classification types	Dataset	Dataset ratios	Accuracy (%)
[10]	LSTM	IDS	Multi-class	CICDDo2019	Imbalanced	99.4
[11]	LSTM	IDS	Binary	CICDDo2019	Balanced using random sampling techniques	98
[2]	MLP-AE	IDS	Multi-class	CICDDoS2019	Imbalanced	98.34
[6]	DNN	IDS	Multi-class	CICDDoS2019	Balanced by using RUS	94.57
[13]	LSTM	IDS	Multi-class	CICDDo2019	Imbalanced	74.42
[8]	1DCNN-LSTM	IDS	Binary and multi-class	CICDDo2019	Imbalanced	99.76
[23]	DNN	IDS	Binary	CICDDoS2019	Imbalanced	99
Our proposed model	MLP	IDS	Multi-Class	CICDDoS2019	SMOTTEEN	98.90

6. Limitations of the Study and Future Research Directions

Despite its contributions to science and literature, the study has the following limitations. To find research articles from reliable sources, the study used keyword-based search strategies, such as DDoS attack detection OR distributed denial-of-service detection AND deep learning. Different research articles may be retrieved using different keywords. Including model explainability and assessing the model's resistance to adversarial attacks are left for future work. The future work will compare model performances on binary and multi-class classifications. The CICDDoS2019 dataset was chosen due to its popularity in the examined research works, wide and diverse feature set, and recentness. Even though each deep learning model's train-test computational times increased, the SMOTEENN approach was utilized to handle class imbalance issues and significantly improve each model's accuracy. In order to solve problems related to model computational time, future research would employ pertinent feature selection strategies. To guarantee model performance consistency, it is anticipated that the model performance will be compared across several DDoS datasets, including the CICIDS2017, NSL-KDD, and UNSW 2018 datasets.

7. Conclusions

Many researchers have sought to address DDoS attack issues using various techniques despite these attacks continuing to rise. This emphasizes the need for beating DDoS attacks using accurate, reliable, and adaptive models. Our study highlights recent studies conducted between 2019 and 2024 on the use of deep learning techniques for DDoS attack detection to identify the gaps, formulate research methodology, and offer workable answers. The review findings reveal that most studies used imbalanced datasets and did not reveal dataset size, fine-tuned model hyperparameters, and train-test-validation split ratios for replication and revalidation of their research findings. Our research findings reveal that none of the reviewed studies use the SMOTEENN dataset balancing strategy on the CICDDoS2019 dataset, considering its potential benefits. Even though model performance varies when using different train-test split ratios, the majority of reviewed papers only use one dataset split to train deep learning models. The study aims to address these issues.

The study assessed the performances of diverse deep learning algorithms, including MLP, DNN, DCNN, LSTM, BiLSTM, CNN-LSTM, and DNN-AE. To obtain better results, rigorous data preprocessing methods like feature scaling, encoding, removing duplicate features, dataset balancing strategy, proper train-test-validation dataset splits, and fine-tuning of model hyperparameters were conducted. The experimental findings show that all models achieved remarkable accuracy rates of above 98.69% on the test dataset following the application of the SMOTEENN dataset balancing strategy and the combination of the MLP model with the SMOTEENN dataset strategy outperform the remaining deep learning models in detection and classification of the DDoS attack.

Although SMOTEENN technique significantly improved the accuracy of each deep learning model, it did not reduce the train-test computational times for each model. According to our experimental results, the SMOTEENN technique reduces the FNR and FPR for models like DNN and BiLSTM. While slightly increasing the FPR, the SMOTEENN technique lowers FNR for the majority of models, including MLP, LSTM, DCNN, CNN-LSTM, and DNN-AE.

Although not considered in the reviewed research works, our study developed the DDoS attack detection and classification workflows to categorize incoming network traffic data using a wide range of features or classify data into one of eight target classifications. Future researchers from many fields may find it easier to identify research gaps with the help of the structured evaluation parameters we utilized in our study to evaluate various research works.

Acknowledgments

The authors thank all participating editors and anonymous reviewers for their valuable suggestions and comments to make the manuscript up to standard.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Author Contribution Statement

Andualem Girma: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – review & editing, Visualization. **Kibreab Adane:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Supervision.

References

[1] Effah, E. Q., Osei, E. O., Maxwell Dorgbefu Jnr., & Tetteh, A. (2024). Hybrid approach to classification of DDoS attacks on a computer network infrastructure. *Asian Journal of Research in Computer Science*, *17*(4), 19–43. https://doi.org/10.9734/ajrcos/2024/v17i4428

[2] Wei, Y., Jang-Jaccard, J., Sabrina, F., Singh, A., Xu, W., & Camtepe, S. (2021). AE-MLP: A hybrid deep learning approach for ddos detection and classification. *IEEE Access*, *9*, 146810–146821. https://doi.org/10.1109/ACCESS.2021.3123791

[3] Ortet Lopes, I., Zou, D., Ruambo, F. A., Akbar, S., & Yuan, B. (2021). Towards effective detection of recent DDoS attacks: A deep learning approach. *Security and Communication Networks*, *2021*(1), 5710028. https://doi.org/10.1155/2021/5710028

[4] Hossain, M. A., & Islam, M. S. (2024). Enhancing DDoS attack detection with hybrid feature selection and ensemble-based classifier: A promising solution for robust cybersecurity. *Measurement: Sensors*, *32*, 101037. https://doi.org/10.1016/j.measen.2024.101037

[5] Najar, A. A., & Manohar Naik, S. (2022). DDoS attack detection using MLP and random forest algorithms. *International Journal of Information Technology*, *14*(5), 2317–2327. https://doi.org/10.1007/s41870-022-01003-x

[6] Cil, A. E., Yildiz, K., & Buldu, A. (2021). Detection of DDoS attacks with feed forward based deep neural network model. *Expert Systems with Applications*, *169*, 114520. https://doi.org/10.1016/j.eswa.2020.114520

[7] Asad, M., Asim, M., Javed, T., Beg, M. O., Mujtaba, H., & Abbas, S. (2020). DeepDetect: Detection of distributed denial of service attacks using deep learning. *The Computer Journal*, *63*(7), 983–994. https://doi.org/10.1093/comjnl/bxz064

[8] Salih, A. A., & Abdulrazaq, M. B. (2024). Cybernet model: A new deep learning model for cyber DDoS attacks detection and recognition. *Computers, Materials & Continua*, *78*(1), 1275–1295. https://doi.org/10.32604/cmc.2023.046101

[9] Wei, Y., Jang-Jaccard, J., Singh, A., Sabrina, F., & Camtepe, S. (2023). Classification and explanation of distributed denial-of-service (2306.17190) attack detection using machine learning and Shapley additive explanation (SHAP) methods. *arXiv Preprint: 2306.17190*. https://doi.org/10.48550/ARXIV.2306.17190

[10] Khempetch, T., & Wuttidittachotti, P. (2021). DDoS attack detection using deep learning. *IAES International Journal of Artificial Intelligence*, *10*(2), 382. https://doi.org/10.11591/ijai.v10.i2.pp382-388

[11] Kumar, D., Pateriya, R. K., Gupta, R. K., Dehalwar, V., & Sharma, A. (2023). DDoS detection using deep learning. *Procedia Computer Science*, *218*, 2420–2429. https://doi.org/10.1016/j.procs.2023.01.217

[12] Abu Bakar, R., Huang, X., Javed, M. S., Hussain, S., & Majeed, M. F. (2023). An intelligent agent-based detection system for DDoS attacks using automatic feature extraction and selection. *Sensors*, *23*(6), 3333. https://doi.org/10.3390/s23063333

[13] Ahmim, A., Maazouzi, F., Ahmim, M., Namane, S., & Dhaou, I. B. (2023). Distributed denial of service attack detection for the Internet of Things using hybrid deep learning model. *IEEE Access*, *11*, 119862–119875. https://doi.org/10.1109/ACCESS.2023.3327620

[14] Kasture, P. (2023). DDoS attack detection using ML. *International Journal for Research in Applied Science and Engineering Technology*, *11*(5), 6421–6424. https://doi.org/10.22214/ijraset.2023.53133

[15] Pereira, J., & Saraiva, F. (2020). A comparative analysis of Imbalanced data handling techniques for machine learning algorithms to electricity theft detection. In *2020 IEEE Congress on Evolutionary Computation*, 1–8. https://doi.org/10.1109/CEC48606.2020.9185822

[16] Wongvorachan, T., He, S., & Bulut, O. (2023). A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining. *Information*, *14*(1), 54. https://doi.org/10.3390/info14010054

[17] Jamal, M. H., Naz, N., Khattak, M. A. K., Saeed, F., Altamimi, S. N., & Qasem, S. N. (2023). A comparison of re-sampling techniques for detection of multi-step attacks on deep learning models. *IEEE Access*, *11*, 127446–127457. https://doi.org/10.1109/ACCESS.2023.3332512

[18] Peranginangin, R., Harianja, E. J. G., Jaya, I. K., & Rumahorbo, B. (2020). Penerapan algoritma *Safe-Level-Smote* untuk peningkatan nilai *G-Mean* dalam klasifikasi data tidak seimbang [Application of Safe-Level-Smote algorithm for improving G-Mean value in imbalanced data classification]. *METHOMIKA: Jurnal Manajemen Informatika & Komputerisasi Akuntansi*, *4*(1), 67–72. https://doi.org/10.46880/jmika.Vol4No1.pp67-72

[19] Zeebaree, S. R. M., Sharif, K. H., & Mohammed Amin, R. M. (2018). Application layer distributed denial of service attacks defense techniques: A review. *Academic Journal of Nawroz University*, *7*(4), 113. https://doi.org/10.25007/ajnu.v7n4a279

[20] Husain, G., Nasef, D., Jose, R., Mayer, J., Bekbolatova, M., Devine, T., & Toma, M. (2025). SMOTE vs. SMOTEENN: A study on the performance of resampling algorithms for addressing class imbalance in regression models. *Algorithms*, *18*(1), 37. https://doi.org/10.3390/a18010037

[21] Nguyen, T.-T., Shieh, C.-S., Chen, C.-H., & Miu, D. (2021). Detection of unknown DDoS attacks with deep learning and Gaussian mixture model. In *2021 4th International Conference on Information and Computer Technologies*, 27–32. https://doi.org/10.1109/ICICT52872.2021.00012

[22] Amaizu, G. C., Nwakanma, C. I., Bhardwaj, S., Lee, J. M., & Kim, D. S. (2021). Composite and efficient DDoS attack detection framework for B5G networks. *Computer Networks*, *188*, 107871. https://doi.org/10.1016/j.comnet.2021.107871

[23] Sbai, O., & Elboukhari, M. (2022). Deep learning intrusion detection system for mobile ad hoc networks against flooding attacks. *IAES International Journal of Artificial Intelligence*, *11*(3), 878. https://doi.org/10.11591/ijai.v11.i3.pp878-885

[24] Shurman, M., Khrais, R., & Yateem, A. (2020). Dos and DDoS attack detection using deep learning and IDS. *The International Arab Journal of Information Technology*, *17*(4A), 655–661. https://doi.org/10.34028/iajit/17/4A/10

[25] Boonchai, J., Kitchat, K., & Nonsiri, S. (2022). The classification of DDoS attacks using deep learning techniques. In *2022 7th International Conference on Business and Industrial Research*, 544–550. https://doi.org/10.1109/ICBIR54589.2022.9786394

[26] Sayed, M. I., Sayem, I. M., Saha, S., & Haque, A. (2022). A multi-classifier for DDoS attacks using stacking ensemble deep

neural network. In *2022 International Wireless Communications and Mobile Computing*, 1125–1130. https://doi.org/10.1109/IWCMC55113.2022.9824189

[27] Zainudin, A., Ahakonye, L. A. C., Akter, R., Kim, D.-S., & Lee, J.-M. (2023). An efficient hybrid-DNN for DDoS detection and classification in software-defined IIoT networks. *IEEE Internet of Things Journal*, *10*(10), 8491–8504. https://doi.org/10.1109/JIOT.2022.3196942

[28] Aktar, S., & Yasin Nur, A. (2023). Towards DDoS attack detection using deep learning approach. *Computers & Security*, *129*, 103251. https://doi.org/10.1016/j.cose.2023.103251

[29] Shieh, C.-S., Nguyen, T.-T., & Horng, M.-F. (2023). Detection of unknown DDoS attack using convolutional neural networks featuring geometrical metric. *Mathematics*, *11*(9), 2145. https://doi.org/10.3390/math11092145

[30] Haider, S., Akhunzada, A., Mustafa, I., Patel, T. B., Fernandez, A., Choo, K.-K. R., & Iqbal, J. (2020). A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks. *IEEE Access*, *8*, 53972–53983. https://doi.org/10.1109/ACCESS.2020.2976908

[31] Li, Y., & Lu, Y. (2019). LSTM-BA: DDoS detection approach combining LSTM and Bayes. In *2019 Seventh International Conference on Advanced Cloud and Big Data*, 180–185. https://doi.org/10.1109/CBD.2019.00041

[32] Mousa, A. K., & Abdullah, M. N. (2023). An improved deep learning model for DDoS detection based on hybrid stacked autoencoder and checkpoint network. *Future Internet*, *15*(8), 278. https://doi.org/10.3390/fi15080278

[33] Roopak, M., Yun Tian, G., & Chambers, J. (2019). Deep learning models for cyber security in IoT networks. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference*, 0452–0457. https://doi.org/10.1109/CCWC.2019.8666588

[34] Nugraha, B., & Murthy, R. N. (2020). Deep learning-based slow DDoS attack detection in SDN-based networks. In *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks*, 51–56. https://doi.org/10.1109/NFV-SDN50289.2020.9289894

[35] Yaser, A. L., Mousa, H. M., & Hussein, M. (2022). Improved DDoS detection utilizing deep neural networks and feedforward neural networks as autoencoder. *Future Internet*, *14*(8), 240. https://doi.org/10.3390/fi14080240

[36] Szymanski, J. R., Chanda, C. K., Mondal, P. K., & Khan, K. A. (2023). *Energy systems, drives and automations: Proceedings of ESDA 2021*. Singapore: Springer. https://doi.org/10.1007/978-981-99-3691-5

[37] Adane, K., Beyene, B., & Abebe, M. (2023). Single and hybrid-ensemble learning-based phishing website detection: Examining impacts of varied nature datasets and informative feature selection technique. *Digital Threats: Research and Practice*, *4*(3), 46. https://doi.org/10.1145/3611392

[38] Vairetti, C., Assadi, J. L., & Maldonado, S. (2024). Efficient hybrid oversampling and intelligent undersampling for imbalanced big data classification. *Expert Systems with Applications*, *246*, 123149. https://doi.org/10.1016/j.eswa.2024.123149

[39] Sumathi, S., Rajesh, R., & Lim, S. (2022). Recurrent and deep learning neural network models for DDoS attack detection. *Journal of Sensors*, *2022*(1), 8530312. https://doi.org/10.1155/2022/8530312

[40] Muhuri, P. S., Chatterjee, P., Yuan, X., Roy, K., & Esterline, A. (2020). Using a long short-term memory recurrent neural network (LSTM-RNN) to classify network attacks. *Information*, *11*(5), 243. https://doi.org/10.3390/info11050243

[41] Masih, N., Naz, H., & Ahuja, S. (2021). Multilayer perceptron based deep neural network for early detection of coronary heart disease. *Health and Technology*, *11*(1), 127–138. https://doi.org/10.1007/s12553-020-00509-3

[42] Oliveira, T. P., Barbar, J. S., & Soares, A. S. (2014). Multilayer perceptron and stacked autoencoder for internet traffic prediction. In *Advanced Information Systems Engineering: 11th IFIP WG 10.3 International Conference*, 61–71. https://doi.org/10.1007/978-3-662-44917-2_6

[43] Adane, K., Beyene, B., & Abebe, M. (2023). Ml and DL-based phishing website detection: The effects of varied size datasets and informative feature selection techniques. *Journal of Artificial Intelligence and Technology*, *4*(1), 18–30. https://doi.org/10.37965/jait.2023.0269

[44] Adane, K., & Beyene, B. (2022). Machine learning and deep learning based phishing websites detection: The current gaps and next directions. *Review of Computer Engineering Research*, *9*(1), 13–29. https://doi.org/10.18488/76.v9i1.2983

[45] Perumal, K., & Arockiasamy, K. (2023). Optimized deep neural network based DDoS attack detection and bait mitigation process in software defined network. *Concurrency and Computation: Practice and Experience*, *35*(12), e7692. https://doi.org/10.1002/cpe.7692

[46] Zhao, J., Liu, Y., Zhang, Q., & Zheng, X. (2023). CNN-AttBiLSTM mechanism: A DDoS attack detection method based on attention mechanism and CNN-BiLSTM. *IEEE Access*, *11*, 136308–136317. https://doi.org/10.1109/ACCESS.2023.3334916

[47] Zhang, Y., Liu, Y., Guo, X., Liu, Z., Zhang, X., & Liang, K. (2022). A BiLSTM-based DDoS attack detection method for edge computing. *Energies*, *15*(21), 7882. https://doi.org/10.3390/en15217882

[48] Zhou, H., & Ling, J. (2023). A cooperative detection of DDoS attacks based on CNN-BiLSTM in SDN. *Journal of Physics: Conference Series*, *2589*(1), 012001. https://doi.org/10.1088/1742-6596/2589/1/012001

[49] Sindian, S., & Sindian, S. (2020). An enhanced deep autoencoder-based approach for DDoS attack detection. *Wseas Transactions on Systems and Control*, *15*, 716–724. https://doi.org/10.37394/23203.2020.15.72

[50] Elsayed, M. S., Le-Khac, N.-A., Dev, S., & Jurcut, A. D. (2020). DDoSNet: A deep-learning model for detecting network attacks. In *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks"*, 391–396. https://doi.org/10.1109/WoWMoM49955.2020.00072

How to Cite: Girma, A., & Adane, K. (2026). DDoS Attack Detection With and Without SMOTEENN Dataset Balancing Strategy: Deep Learning Approaches. *Artificial Intelligence and Applications*. https://doi.org/10.47852/bonviewAIA62026187