September 2025, Vol:3, Issue:2

International Journal of New Findings in Engineering, Science and Technology

journal homepage: https://ijonfest.gedik.edu.tr/

Received:31.05.2025 Accepted:19.09.2025 Pablished:29.09.2025

DDoSGedik30K: A Unique Dataset with FFNN and LSTM-Based Deep Learning Models for Detecting DDoS Attacks

Şenay Kocakoyun Aydoğan[a*], Turgut Pura[b], Zeki Çıplak[c], Anıl Yıldız[d]

[a]*Department of Information Security Technology, Istanbul Gedik University, Türkiye, senay.aydogan@gedik.edu.tr (*Corresponding author)*

[b]*Department of Software Engineering, Istanbul Gedik University Türkiye, turgut.pura@gedik.edu.tr*

[c]*Department of Computer Technologies, Istanbul Gedik University, Türkiye, zeki.ciplak@gedik.edu.tr*

[d]*Department of Information Computer Technologies, Istanbul Beykent University, Türkiye anilyildiz@beykent.edu.tr*

Abstract

The rapid advancements in network technologies, along with the increasing volume and scope of data transmitted over networks, have led to a rise in both the intensity and complexity of cyber threats and attacks. One of the most prominent and destructive types of cyberattacks threatening network and system security is the Distributed Denial of Service (DDoS) attack. This study examines the use of deep learning techniques to develop an effective detection mechanism against the growing number of DDoS attacks today. For this purpose, a dataset called DDoSGedik30K, which includes real-world attack scenarios, was created. Using this dataset, a total of 12 models were developed based on Feedforward Neural Network (FFNN) and Long Short-Term Memory (LSTM) deep learning architectures. The fact that all models exceeded a 99.9% accuracy rate proves that the proposed dataset is highly effective in detecting DDoS attacks. Additionally, cross-validation (CV) was employed in the model evaluation, and the experimental results are consistent with the obtained performance metrics, demonstrating the reliability and robustness of the proposed approach. This study introduces the DDoSGedik30K dataset and deep learning models that significantly improve DDoS attack detection. These contributions enhance cybersecurity research and provide actionable insights for future advancements.

Keywords: DDoSGedik30K, DDoS Dataset, DDoS Detection, DDoS Analysis, Deep Learning.

1. INTRODUCTION

The rapid increase in digitalization has led to a parallel growth in cybersecurity threats. A Denial of Service (DoS) attack is a cybersecurity threat that causes a resource or service to become unavailable or unresponsive for legitimate users. The resource could be a single server, a group of machines (such as a dedicated server pool), or a network. If an attacker successfully renders the targeted resource inaccessible to legitimate users, the DoS attack is considered successful. This type of attack can be carried out through various methods and at different layers of the OSI and TCP/IP models [1].

Attackers often lack the computing power to launch a DoS attack, especially when large amounts of traffic are required. This need for high computational power has led to the rise of a related attack known as a Distributed Denial of Service (DDoS) attack. Executing a DoS attack with significant computational resources is a unique challenge. Today, DDoS attacks are considered one of the most common and destructive types of cyberattacks threatening network and system security [2-4]. DDoS attacks aim to overwhelm a system or network, causing it to become unavailable and leading to significant financial losses and service disruptions. For this reason, early detection and effective prevention of such attacks have become a critical necessity in the field of cybersecurity.

https://doi.org/10.61150/ijonfest.2025030201

Each type of DDoS attack has its own unique execution technique, driven by various factors such as the software tool used to generate the attack traffic, the communication layer, the target protocol, and more. The ultimate goal of the attacker is to render the target resource inaccessible to legitimate users. Although various protection mechanisms can secure critical resources, system vulnerabilities are often exploited by hackers. Table 1 provides information on different types of DDoS attacks.

Table 1. Overview of different DDoS attack types along with their targets and associated vulnerabilities [5].

Types of DDoS Attacks	Targets	Exploited Vulnerabilities
Data Overflow	Network bandwidth or the capacity of networks or servers.	Limited network bandwidth / Limited server capacity to process requests.
Attack on Network Devices	Network devices/hardware such as routers, switches, and firewalls.	Vulnerability/error in device firmware.
Protocol Attack	Protocol Services.	Protocol limitations and vulnerabilities such as ARP poisoning, IP spoofing, and TCP SYN flooding.
Application Attack	Application Services.	Application Limitations and Vulnerabilities.
Operating System Attack	Operating System Services.	Vulnerability/error in operating system software.

A key feature of DDoS attacks is that they are executed simultaneously from multiple devices, rather than a single source. These devices are typically computers that have been compromised by malware, commonly referred to as a botnet [6]. The objective here is to prevent legitimate users from accessing the system, network, or service. The targeted system can be a website, a server, a gaming platform, or another online service. The attack method overwhelms the target by flooding it with fake requests, exhausting its resources such as CPU, memory, and bandwidth [7]. As a result, the target system may slow down, malfunction, or completely crash. The effects of the attack can include the targeted service or network becoming completely unavailable, business losses due to service interruptions, customer dissatisfaction, and damage to reputation.

A DDoS attack is established by organizing a network composed of connected machines. These machines are referred to as "Zombies," and they form a part of the network known as a "Botnet." A botnet consists of many computers that have been compromised and are remotely controlled by cyber attackers [8]. This traffic consumes the bandwidth, processing power, or other resources of the target system, leading to its slowdown, crash, or complete inaccessibility [9]. The mechanism for controlling the network is established by assigning the Command and Control (C&C) function to special high-capacity resources that give direct instructions on behalf of the attacker. The next layer consists of "Handlers" selected by the C&C function to transmit commands and receive responses. Beneath each handler is a series of zombies used to send attack traffic directly to the target (victim). They also relay information about the victim back to the relevant handlers through the C&C function [10]. Therefore, client-server technology is used for communication within the botnet.

There are several methods to protect against DDoS attacks. The use of continuous monitoring and analysis tools allows for the detection of unusual spikes in network traffic, enabling the identification of early signs of DDoS attacks [8]. Security Information and Event Management (SIEM) tools collect and analyze network traffic and security events, facilitating quick responses by detecting anomalies [11].

With the rapid advancement of Artificial Intelligence (AI) technologies, particularly deep learning techniques used in many fields, these methods can also serve as a precise and robust solution for detecting DDoS attacks [12]. AI and Machine Learning (ML) based tools can analyze traffic to identify abnormal behaviors and enable automated responses. Machine learning represents a promising approach in predicting and simulating human behavior with computational intelligence, having been successfully applied to widespread real-world problems [13]. For ML-focused detection of DDoS attacks, evaluating datasets that encompass DDoS attacks is widely accepted for building ML models. In this study, a dataset containing a mixture of specific attack types was created, and deep learning-based models were developed for attack detection using this dataset.

1.1. Contribution of Study

This study provides significant contributions to the literature by comprehensively examining the effectiveness of deep learning models developed to detect DDoS attacks in the field of cybersecurity. These contributions can be summarized in several points:

- A high-quality dataset for cybersecurity research (DDoSGedik30K) has been contributed to the literature through a system created in the laboratories of Istanbul Gedik University, using real network attacks and normal connections. The relevant dataset has been developed to mimic real-world scenarios through various firewall configurations. It includes vulnerabilities that could be encountered in the real world, particularly focusing on changes in firewall configurations and intentionally created vulnerabilities.
- The effects of different architectures, epoch counts, learning rates, and optimization algorithms on model performance have been analyzed in detail. These analyses guide the selection of the most suitable deep learning model for different scenarios.
- This study illuminates future research aimed at developing real-time and effective models for DDoS attack detection. It emphasizes that deep learning techniques can also be utilized in cybersecurity solutions, laying an important foundation for the integration of these techniques into future security solutions. As cybersecurity becomes increasingly critical, such efforts will play a key role in protecting networks and systems.

2. RELATED WORKS

Many studies have been conducted on DDoS attacks and their detection. An examination of these studies reveals the use of various methods, resulting in different outcomes depending on the approach.

Erhan and Anarım [14] investigated the statistical modeling of network traffic observed during DDoS attacks. To conduct this analysis, they utilized 16 traffic attributes, employing the BOUN DDoS dataset, which is derived from Boğaziçi University's (BOUN) network and includes TCP SYN flood attacks. The first method used was to create a statistical dataset for both attack and non-attack scenarios from the training dataset. This statistical dataset was then used to detect attacks within the experimental distributions of the test data. The second method employed was the K-Means clustering algorithm. The results showed an accuracy rate of 98% for the attack model and 97% for the normal model.

Asarkaya et al. [15] examined the prediction of DDoS attacks using machine learning classification algorithms. They used the DDoS Attack Network Logs dataset downloaded from Kaggle, which contained a total of 902,186 entries classified into 'Normal', 'Http-Flood', 'SIDDOS', 'Smurf', and 'UDP-Flood' categories, using 27 different attributes. The methods employed included K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), and Random Forest (RF) algorithms. The model created using KNN achieved an accuracy of 93%, SVM reached 98%, MLP achieved 99%, and RF also obtained 98%.

Sharif and Beitollahi [16] focused on the critical need for improved DDoS detection, aiming to minimize false alarms while accurately detecting both known and unknown DDoS attacks. They utilized multiple datasets (CICIDS2017 [17] and CICDDoS2019 [18]) for this purpose. Additionally, Genetic Algorithms (GA) were adopted for automatic hyperparameter optimization to ensure efficient and effective DDoS detection. In this study, the RF classifier achieved an accuracy rate of 99.9%, with precision, recall, and F1 score values of 100%, 99.8%, and 99.9%, respectively.

In another study, Ahmed et al. [19] addressed application layer DDoS attacks by analyzing the characteristics of incoming packets, including the size of HTTP frame packets, the number of sent Internet Protocol (IP) addresses, continuous port mappings, and the number of IP addresses using proxy IPs. Standard datasets such as the CTU-13 dataset-2011, real web logs from organizations-2019, and datasets experimentally created from DDoS attack tools were used. The MLP algorithm was utilized to evaluate the effectiveness of metric-based attack detection, including Slow Loris, Hulk, Golden Eyes, and Xerex. The simulation results demonstrated that the proposed MLP classification algorithm had an efficiency rate of 98.99% in detecting DDoS attacks.

Sharif et al. [20] aimed to fill the gap by investigating the impact of the accessibility of DDoS attack tools on the frequency and severity of attacks. They proposed a machine learning solution to detect DDoS attacks using a feature selection technique that improved speed and efficiency, resulting in a significant reduction in the feature subset. The CICIDS2017 dataset was used, in which the abstract behaviors of 25 users were profiled according to HTTP, HTTPS, FTP, SSH protocols, and emails. After feature selection, six selected features were placed into a three-layer MLP. By using a deliberate approach for feature

selection, they found that the model's effectiveness was greatly enhanced. The provided evaluation criteria indicated that the model achieved a high accuracy level of 99.9%, with 96% precision, 98% recall, and an F1 score of 97%.

Kareem et al. [21] proposed a detection system called eXtreme Gradient Boosting (XGB-DDoS), which utilizes a tree-based ensemble model to detect application layer DDoS attacks. The XGB-DDoS detection model outperformed other models used in the study (KNN, SVM, and PCA hybridized with XGBoost). The proposed method was observed to be effective in detecting application layer DDoS attacks. The CICIDS2017 dataset, created by the Canadian Cybersecurity Institute, was used for the training and testing of the most successful model. The performance results of the model were significantly high, with accuracy, precision, recall, and F1 scores of 0.999, 0.997, 0.995, and 0.996, respectively.

In their study, Salama et al. [22] used the CICDDoS2019 dataset to train and evaluate various machine learning algorithms, including Stochastic Gradient Boosting (SGB), Decision Tree (DT), KNN, Naive Bayes (NB), SVM, and Logistic Regression (LR). The results of the study demonstrated that all machine learning algorithms were capable of effectively detecting DDoS attacks with high accuracy, precision, and recall. However, it was observed that the SVM algorithm achieved the highest accuracy rate (0.99) compared to the others.

Teeb Hussein [23] proposed a new approach for DDoS attack detection using Denoising AutoEncoder (DAE) and Convolutional Neural Network (CNN) for feature selection and classification. The NSL-KDD dataset was used to evaluate the performance of this new model through three main steps (Data Preprocessing, Hyperparameter Optimization, and Classification). After applying the methods, the obtained accuracy, precision, recall, and F1 score metrics were measured at 97.7, 98.1, 97.7, and 97.8, respectively. The detection rate for DoS was found to be 100%.

Manaa and colleagues [24] utilized three significant datasets: UNSW-NB 15, UNSW-2018 Botnet, and Edge IIoT, in an Anomaly-Based Intrusion Detection System (AIDS) to detect and mitigate DDoS attacks. The proposed study used methods such as RF, SVM, LR, MLP, Deep Artificial Neural Network (ANN), and Long Short-Term Memory (LSTM) to identify DDoS attacks. These methods were compared against the fact that the database stored trained signatures. As a result, RF exhibited promising performance with 100% accuracy and minimal false positives when testing both the UNSW-NB 15 and UNSW-2018 Botnet datasets.

Najar, Sugali, Lone, and Nazir [25] proposed a new framework for detecting and classifying DoS attacks with high accuracy and low computational cost. Random sampling was used to address class imbalance, and feature selection techniques such as low information gain, semi-fixed elimination, and principal component analysis were employed for optimal feature selection and reduction. The proposed CNN-based model achieved outstanding performance with 99.99% accuracy for binary classification and 98.44% for multi-class classification.

In their study, Zekri et al. [26] discussed DDoS attack detection in cloud environments using machine learning algorithms. In this research, a simple dataset of five features was analyzed using the C4.5 decision tree algorithm that works on the gain ratio as a splitting criterion. The detection module was also supported by signature-based detection for improved results. For comparison purposes, the C4.5 algorithm was evaluated alongside Naive Bayes and K-Means algorithms. The attack traffic was simulated using the Hping3 tool. Snort, an open-source Intrusion Detection System (IDS), was utilized for signature-based detection. It was found that the C4.5 algorithm yielded better results compared to the other two algorithms, with the highest accuracy recorded at 98.8%.

Yuan et al. [30] tested the application of deep learning in detecting DDoS attacks using Recurrent Neural Networks (RNN), referred to as DeepDefense. In this study, 20 network traffic features were sampled from the ISCX'12 dataset. To accelerate the training process, batch normalization layers were also added to the RNN layers. Four different types of RNNs, including LSTM and 3LSTM, were analyzed and compared with each other. Additionally, a specific RNN parameter tuning was compared with the RF classifier. The highest accuracy was recorded at 98.41% using the 3LSTM algorithm.

3. DATASET

The dataset utilized in this study was generated through detailed operations conducted on the Pentest servers located in the computer laboratories of Istanbul Gedik University [31]. This dataset was specifically developed for the purpose of detecting DDoS attacks, taking into account the nature and complexity of these attacks.

The primary aim in creating this dataset is to enhance the performance of deep learning algorithms in the field of cybersecurity and to develop models that are suitable for real-world scenarios. In this context, the number of variables used for detecting DDoS attacks can directly influence the complexity of the model and its processing speed [32, 33]. Working with a limited number of variables may facilitate faster training of the model and improve the speed of real-time detection processes. This situation presents a practical advantage, particularly in scenarios with resource constraints. Thus, the objective is to achieve high accuracy rates in attack detection while maintaining the processing efficiency of the system.

The creation of the dataset involved executing attacks from both internal networks (such as controlled networks protected by security protocols, like corporate networks) and external networks (which are generally open to the internet and harbor greater threats). These attack scenarios simulated various threats that the system might face, enabling a comprehensive collection of data across a wide range. This methodology allows machine learning models to operate effectively not only against specific types of attacks but also against diverse attack vectors. Over a four-month period, various attack techniques and scenarios were implemented, resulting in the collection of tens of thousands of data records. These records have facilitated the assessment of the generalization capabilities of machine learning models and their resilience to real-world scenarios.

The generated dataset has been named "DDoSGedik30K" and has been introduced to the literature as a valuable resource for training machine learning algorithms. The security of the system is maintained through a complex network structure comprising firewalls, servers, and computers [34]. Firewall systems are a crucial component that enhances security levels by monitoring network traffic and detecting potential threats [35].

In the system established for this study, two firewalls were utilized: Fortigate [36] and Mikrotik [37]. A representative image of the system in operation is shown in Figure 1.

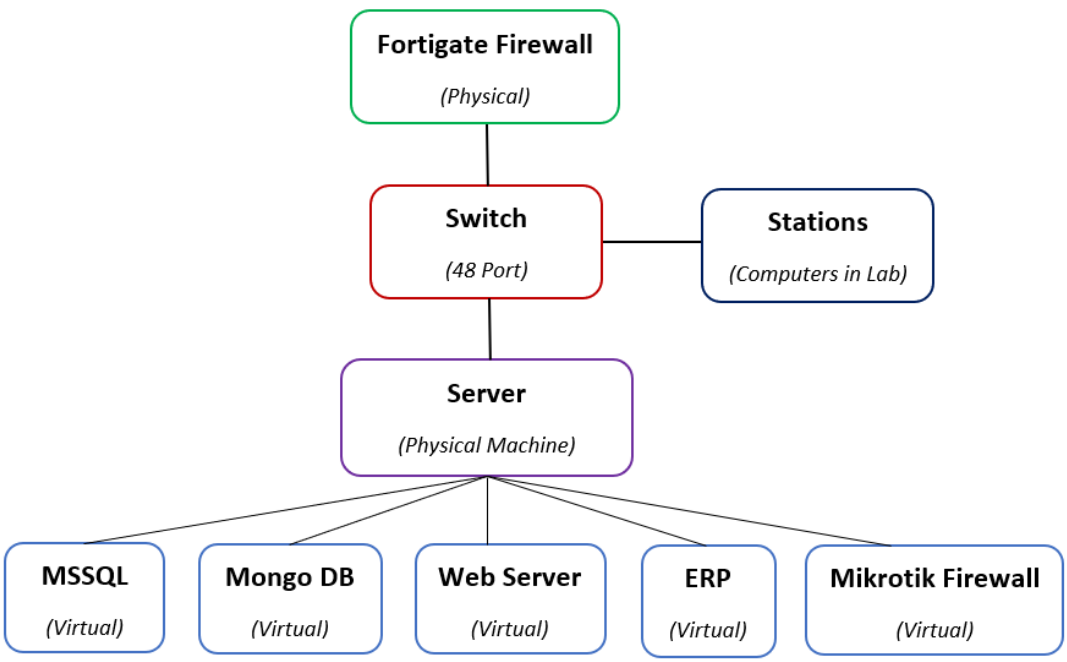

Figure 1. Representative diagram of the laboratory network setup used for dataset creation, showing the firewall, switch, server, stations, and virtualized services deployed for DDoS attack simulations.

The Fortigate operated physically, while the Mikrotik functioned on a virtual machine. These two firewalls were used together for a very short time. Generally, the Fortigate firewall was active, while the Mikrotik firewall was kept in passive mode. The rules in the firewalls used were not kept the same for the sake of data diversity. Occasionally, some rules were intentionally set to passive, creating deliberate vulnerabilities. The primary goal of these vulnerabilities was to diversify the

data and produce the most realistic datasets by mimicking system vulnerabilities or errors that might be encountered in real life.

The 48-port smart switch used in the system ensured the management of all network devices and optimized network traffic. The switch enabled logical separation and management of devices on the network through VLAN (Virtual Local Area Network) configuration, although only a single VLAN was used in this study. Using a single VLAN facilitated access for all devices to the server over the internal network. This configuration was intentionally applied, especially for testing cyber-attack scenarios and examining the security vulnerabilities of the system.

The logs generated in the system were not limited to firewall logs; various programs that could monitor and analyze network traffic were also used. Among these programs, CICFlowmeter [38] and Wireshark [39] stand out.

CICFlowmeter is a tool that analyzes network traffic and extracts flow-based features. This tool provides detailed information about each connection by monitoring data flows on the network, and this information serves as valuable input for machine learning algorithms. Wireshark, on the other hand, is a widely used network protocol analyzer around the world. This program helps understand the nature and propagation of attacks with its ability to monitor network traffic in real-time. The logs obtained through these programs were combined with the firewall logs and used in the creation of the DDoSGedik30K dataset. The DDoSGedik30K dataset consists of 31,458 records. The dataset contains seven variables, which specify the amount of bytes and packets sent in one second, the response time of the sent requests, and the total amount of data packets. Descriptions of all the variables are presented in Table 2.

Table 2. List of Features and Their Corresponding Descriptions in the DDoSGedik30K Dataset.

Feature	Description
time	Response time of the sent request
bytes	Average byte amount per second
packets	Average packet amount per second
total_bytes	Total amount of data packets in bytes
total_packets	Total amount of data packets
label	Type of attack (DDoS, SQL, Port Scan, or None)
class	Attack or Harmless

4. DATA PREPROCESSING

In this study, various deep learning models have been developed to detect DDoS attacks, all of which focus on binary classification problems. Binary classification means that each record in the dataset is labeled as either "attack" or "not an attack" (Harmless). In this context, the "class" variable in the DDoSGedik30K dataset has been designated as the target variable for the model, and all models have been trained based on this variable. The "label" variable in the dataset has been excluded from model training. The primary reason for this is that the "label" variable contains information that could overshadow the importance of the other variables. For example, the "label" variable can directly indicate whether a connection is an attack, and in this case, the model could classify based solely on this information without needing the other independent variables. However, this situation is not expected in real-world scenarios. In real-time network traffic analysis, whether a request is an attack is not information contained within the request itself. Therefore, to enable the model to make more complex and reliable predictions, the "label" variable has been removed from the training dataset. The other independent variables in the dataset represent various features of network traffic. These variables are factors that define the data flow over the network and are used to identify a potential attack. In this study, model design has been carried out based on a total of 6 variables, 5 of which have been used as independent variables. These variables have been carefully selected to ensure that deep learning models achieve high accuracy rates in attack detection.

The rationale behind our dataset relying on only 6 features is that our goal was to create a dataset that could work even on small, fast, and weaker configurations. In this context, we focused only on the basic features. This is because adding more features could increase the complexity of the model and affect its operating speed. Additionally, reducing the size of the dataset provided a significant advantage in terms of training time and system requirements. By choosing this approach in our work, we aimed to create a flexible dataset that could be tested on faster and lower-capacity systems. The DDoSGedik30K

dataset contains a total of 31,458 data records, each consisting of an equal amount of attack and non-attack records. This balance is critical for ensuring that the model learns both classes equally well. The balanced distribution in the dataset enables the model to accurately detect attacks while avoiding misclassifying normal network traffic as false positives. Information regarding the quantities of attack and non-attack values in the dataset is shown in Table 3.

Table 3. Distribution of Values in the 'Class' Variable of the DDoSGedik30K Dataset.

Class	Data Amount
Harmless (0)	15729
Attack (1)	15729

Before proceeding to the training phase of the model, the "Harmless" values in the class variable were recoded as "0," while the "Attack" values were recoded as "1." All variables, except for the target variable class, were normalized using the Z-Score Normalization (Standardization) method [40]. It is a well-known fact that the standardization processes applied to datasets generally enhance model performance [41, 42].

5. METHODOLOGY

In this study, a total of 12 models were produced using two different architectures: Feedforward Neural Network (FFNN) and Long Short-Term Memory (LSTM). Experiments were conducted with different hyperparameter values for both deep learning algorithms.

FFNN is one of the most basic and widely used types of artificial neural networks [43-45]. These types of neural networks transfer information in only one direction, from the input layer to the output layer. Data is passed forward and processed through each layer, starting from the input layer. The neurons in each layer multiply the inputs from the previous layer by weights, add a bias value, and apply an activation function. This process continues until the output layer is reached, producing a final output.

LSTM is an advanced type of recurrent neural networks (RNN) designed specifically for working with time series data and sequential data [46-49]. LSTM layers address the "long-term dependency" problem experienced by traditional RNNs, allowing the model to remember important events from the past without forgetting them. LSTM operates with a structure called "cell state," which enables the model to carry significant information over long periods. This feature provides a critical advantage in analyzing long-duration data flows, such as DDoS attacks.

By recognizing important features in sequential data, LSTM makes more accurate predictions and can effectively detect sudden changes in the data flow. This is particularly beneficial in scenarios where attacks may intensify over a certain period. Data is passed forward through LSTM cells, starting from the model's input layer. At each time step, LSTM updates the cell state and produces an output. The training of LSTM models is conducted using a special backpropagation algorithm for time series data, known as "backpropagation through time". Half of the models produced in this study are based on the FFNN architecture, while the other half are based on the LSTM architecture. The number of layers and neurons in both architectures are values determined after a series of tests. To ensure that the accuracy rate is not adversely affected, care was taken to use as few layers and neurons as possible for the models to operate more efficiently. A representative image of the FFNN architecture is shown in Figure 2.

The FFNN models have an input layer with 5 input variables. These 5 input variables represent the features obtained from the dataset and define the characteristics of each data point to be processed by the model. This layer contains 30 neurons, meaning there are 30 separate units, each responsible for processing one feature. The activation function used for this layer is ReLU (Rectified Linear Unit) [50, 51]. ReLU is commonly preferred in deep learning models because it provides a nonlinear relationship while enabling the model to learn quickly and effectively. The second layer has 20 neurons, which process the 30-dimensional output vector from the input layer and pass it to the next layer. The activation function used in the hidden layer is again selected as ReLU. ReLU is widely preferred in neural networks because it largely prevents the vanishing gradient problem [52] and allows effective learning even in deeper layers of the model. The third layer is a Dropout layer, which is added to enhance the model's generalization ability. Dropout allows for randomly disabling neurons during the training process, preventing the model from overfitting to a specific dataset [53]. In this model, the Dropout rate is set to 0.5, meaning 50% of the neurons are randomly disabled during each training step. This makes the model less sensitive to noise in the dataset

and helps it perform better with new, unseen data. The final layer of the FFNN architecture contains only 1 neuron. This layer serves as the output layer and is used to detect whether a DDoS attack is present. The reason for having 1 neuron in the output layer is that the model solves a binary classification problem. The activation function used in this layer is the sigmoid function. The sigmoid function [54] produces an output between 0 and 1, representing the probability of each input belonging to a specific class.

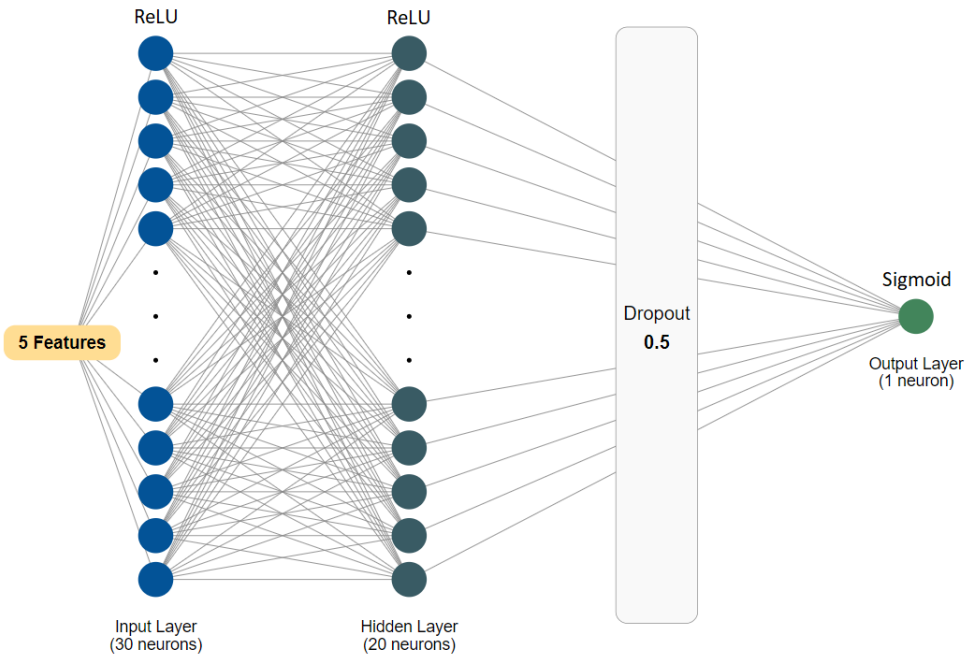

Figure 2. Layers and neurons of the FFNN architecture.

The FFNN models have an input layer with 5 input variables. These 5 input variables represent the features obtained from the dataset and define the characteristics of each data point to be processed by the model. This layer contains 30 neurons, meaning there are 30 separate units, each responsible for processing one feature. The activation function used for this layer is ReLU (Rectified Linear Unit) [50, 51]. ReLU is commonly preferred in deep learning models because it provides a nonlinear relationship while enabling the model to learn quickly and effectively. The second layer has 20 neurons, which process the 30-dimensional output vector from the input layer and pass it to the next layer. The activation function used in the hidden layer is again selected as ReLU. ReLU is widely preferred in neural networks because it largely prevents the vanishing gradient problem [52] and allows effective learning even in deeper layers of the model. The third layer is a Dropout layer, which is added to enhance the model's generalization ability. Dropout allows for randomly disabling neurons during the training process, preventing the model from overfitting to a specific dataset [53]. In this model, the Dropout rate is set to 0.5, meaning 50% of the neurons are randomly disabled during each training step. This makes the model less sensitive to noise in the dataset and helps it perform better with new, unseen data. The final layer of the FFNN architecture contains only 1 neuron. This layer serves as the output layer and is used to detect whether a DDoS attack is present. The reason for having 1 neuron in the output layer is that the model solves a binary classification problem. The activation function used in this layer is the sigmoid function. The sigmoid function [54] produces an output between 0 and 1, representing the probability of each input belonging to a specific class.

The first layer of the LSTM models, the LSTM layer, contains 10 memory cells (neurons) and accepts 5 variables as input size. This layer learns the relationships between past data points and the current data, allowing it to capture changes over time. A representative image of the LSTM architecture is shown in Figure 3.

The LSTM layer allows the model to remember past information from time series data and utilize this information to predict future events. The strong memory structure of LSTM makes it an ideal solution, especially for long-term and complex

attack scenarios. This feature of LSTM is particularly beneficial for detecting DDoS attacks, as attacks often occur based on specific patterns within a defined time frame. In this layer, the ReLU activation function is used.

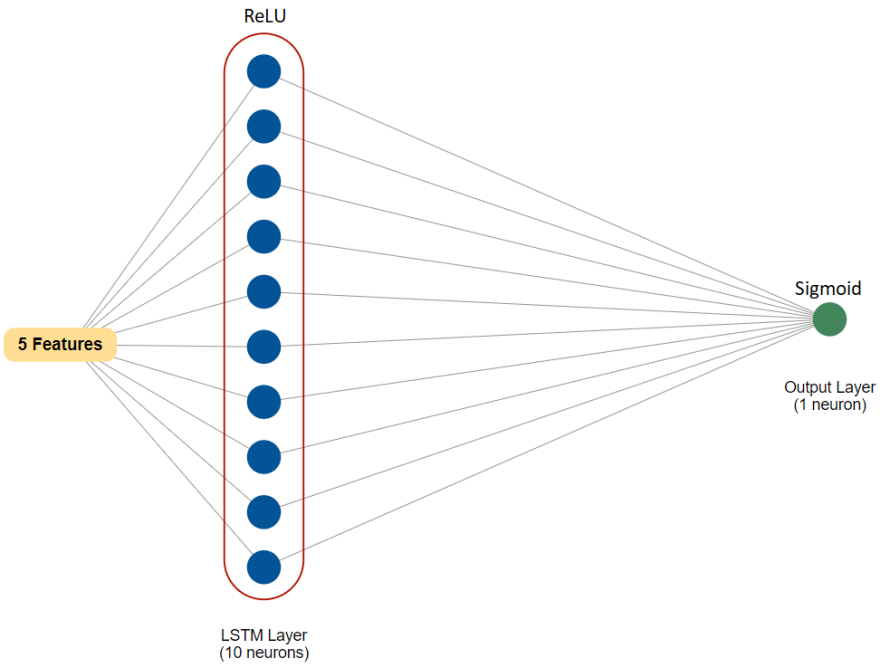

Figure 3. Layers and Neurons of the LSTM Architecture

The LSTM layer allows the model to remember past information from time series data and utilize this information to predict future events. The strong memory structure of LSTM makes it an ideal solution, especially for long-term and complex attack scenarios. This feature of LSTM is particularly beneficial for detecting DDoS attacks, as attacks often occur based on specific patterns within a defined time frame. In this layer, the ReLU activation function is used.

The final layer of the architecture contains only 1 neuron. This layer processes the outputs from the LSTM layer and converts them into a result. Its purpose is to classify each data point as either an "attack" or "not an attack." Therefore, there is only one neuron in the output layer, and the activation function used in this layer is sigmoid.

The primary device used for training all models is a work laptop provided by Istanbul Gedik University, which has the following specifications: Windows 10 operating system, Intel Core i7-12650H CPU at 2.30 GHz, 16GB RAM, 512GB SSD, and NVIDIA GeForce RTX 3060 Laptop GPU.

The development environment used is Jupyter Notebook v6.5.2, and the programming language preferred is Python v3.11.3. To create deep learning models, the following libraries that support usage in Python were utilized: TensorFlow [55], Keras [56], NumPy [57], Pandas [58], and Scikit-learn [59].

6. RESULTS and DISCUSSION

In this study, experiments were conducted to detect DDoS attacks using a total of 12 different models built with two different deep learning architectures. Each model was trained with various parameters. The impact of these parameters on the results was investigated. The differences in the models were interpreted based on the findings obtained. The characteristics of all models produced in the study, along with their hyperparameters and the accuracy values achieved, are shown in Table 4.

Table 4 shows that the first and most striking result is that all models produced very successful results. While the accuracy rates of all models were above 0.999, their loss values remained quite low, demonstrating the effectiveness of the developed deep learning models in detecting DDoS attacks. Here, the test accuracy values show the results determined based on data outside the training data. This represents how each model performs on a data set that did not participate in the training process.

The accuracy of the results obtained in the study and their consistency with cross-validation (CV) were carefully evaluated. When comparing test accuracies with K-fold cross validation (K=5 and K=10) values, both validation types are highly consistent with the accuracy of the models. This indicates that overfitting has been prevented and that the models accurately reflect their overall performance. Using CV is a critical step to ensure that the model does not depend solely on a specific data set and exhibits more generalized performance. Cross-validation for K=5 and K=10 increased the reliability of the model by using different training and test data sets, while also reducing the risk of overfitting.

Table 4. Performance Results of the Models Evaluated in the Study, Including Architecture, Hyperparameters, and Accuracy.

Model	Architecture	Optimizer	Epoch	Elapsed Time	Learning Rate	Loss	Accuracy	CV (K=5)	CV (K=10)
1	FFNN	Adam	50	44.566	0.0001	0.003	0.99937	0.99936	0.99942
2			20	18.987	0.0003	0.003	0.99936	0.99946	0.99933
3			10	10.221	0.0010	0.002	0.99941	0.99939	0.99942
4		SGD	50	40.326	0.0001	0.007	0.99925	0.99942	0.99952
5			20	16.117	0.0003	0.004	0.99925	0.99942	0.99955
6			10	8.524	0.0010	0.006	0.99937	0.99955	0.99939
7	LSTM	Adam	200	212.609	0.0001	0.014	0.99904	0.99977	0.99984
8			100	102.209	0.0003	0.007	0.99925	0.99980	0.99990
9			40	43.477	0.0010	0.008	0.99947	0.99974	0.99965
10		SGD	200	208.580	0.0001	0.009	0.99872	0.99980	0.99996
11			100	11.236	0.0003	0.007	0.99936	0.99971	0.99984
12			40	43.280	0.0010	0.007	0.99925	0.99965	0.99974

The Adam optimization algorithm [60, 61] has generally provided better performance in models. This is because Adam provides faster convergence and can dynamically adjust the learning rate. Stochastic Gradient Descent (SGD), on the other hand, has a deterministic structure and is a simpler algorithm, which may require more epochs [62, 63]. It may also have had a negative impact on the accuracy of the model.

When examining all models in terms of loss value, Model 3 stands out. “Elapsed Time” is a value that indicates the training duration of the model in seconds. A shorter training time demonstrates that the model is trained quickly and, therefore, is more efficient. This becomes especially important when working with large datasets and in real-time applications. Model 6 has the fastest training time at 8,524 seconds.

The most successful model in the study is model number 3 in Table 4. It achieved a very high accuracy of 99.41% according to the accuracy metric and also obtained a result of 99.42% in CV (K=10). Furthermore, the low number of epochs suggests that the model can be trained quickly and has high generalization ability. The model with the lowest loss value is also model number 3 (0.002). This result also indicates that the model's predictions are accurate, and the error rate is very low. Figure 4 shows the Train and Validation Loss graph for Model 3.

In Figure 4, it can be observed that the losses of Model 3 have successfully decreased throughout the training and validation processes, indicating that the model is effectively learning. Additionally, it is evident that the number of epochs was chosen correctly, allowing the model to maintain good generalization performance without overfitting. As shown in the graph, both training loss (red line) and validation loss (blue line) decrease as the number of epochs increases. This indicates that the model is learning better at each epoch and reducing the error rate.

The stability of the validation loss at low levels shows that the model is not overfitting, meaning it maintains its overall performance without excessively fitting the training data. The low values of both training and validation losses suggest that the model performs effectively on both training and test data. This implies that the model is successful in detecting DDoS attacks and has good generalization ability. It was observed that losses could potentially decrease further with an increased number of epochs; however, at this point, it became clear that the losses were stabilizing and that more epochs might be unnecessary. This approach has been applied in the same way to all other models.

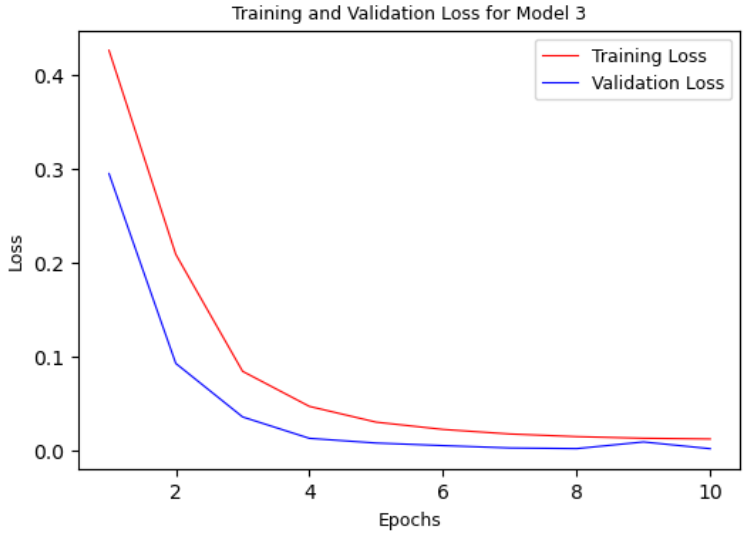

Figure 4. Comparison of Training and Validation Loss for Model 3 throughout the Epochs.

To better analyze Model 3, it is also beneficial to examine the Receiver Operating Characteristic (ROC) curve [66]. The ROC curve is an important tool for evaluating the performance of a classification model. In particular, it provides in-depth information about how well the model classifies by comparing the true positive rate and false positive rate.

The AUC (Area Under the Curve) value of the ROC curve is a metric that indicates the overall accuracy level of the model, with an AUC value of 1.0 indicating a perfect model. The success of the model increases with the number of correct classifications, while a low number of misclassifications is also important. Therefore, the ROC curve allows us to make a more comprehensive assessment than accuracy metrics alone. A model with a high AUC value demonstrates the ability to make accurate predictions despite the diversity of the data.

The shape of a ROC curve can also help determine whether a model is overfitting. If the model has overfitted to the training data, the ROC curve may start with very high accuracy and quickly approach the upper right corner. However, if the model's ability to generalize to real-world data is weak, the curve will generally not be as steep, and the false positive rate will increase. Therefore, examining the ROC curve is crucial for understanding the model's generalization capacity and reliability. Figure 5 shows the ROC curve graph for Model 3.

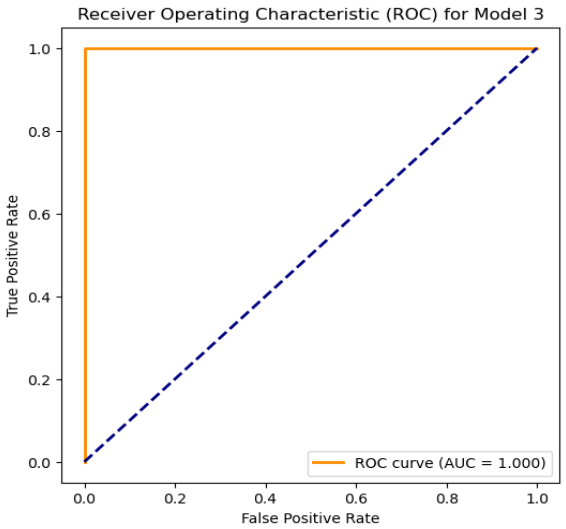

Figure 5. ROC graph for Model 3.

Looking at the ROC curve in Figure 5, the AUC (Area Under the Curve) value is indicated as 1.000. This indicates that the model has excellent discrimination power. In other words, the model is highly effective at making correct classifications, with a very low false positive rate and a very high true positive rate. This result shows that the model provides high accuracy in correctly detecting DDoS attacks. The steep shape of the ROC curve proves that the model performed excellently on the training data and is consistent with the test data. When the AUC value is this high, the likelihood of the model overfitting on the test data is low. The model appears to have a high generalization capacity and does not suffer from overfitting. The confusion matrix for Model 3 also supports this view. Figure 6 presents the confusion matrix for Model 3.

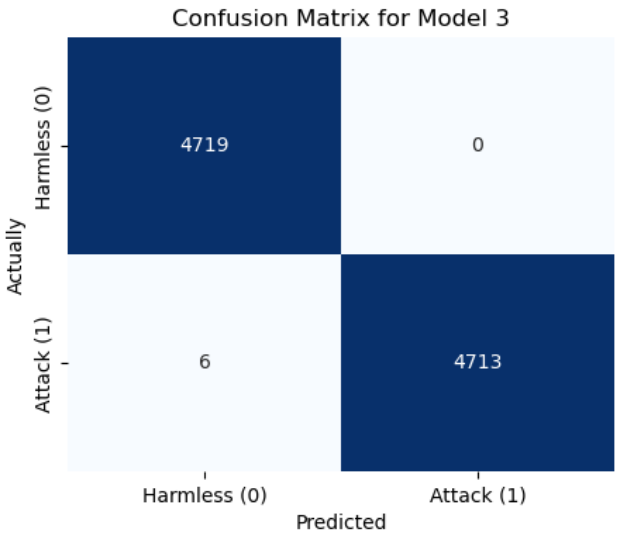

Figure 6. Confusion matrix for Model 3.

Figure 6 shows that the model is quite successful. In particular, the false positive count being 0 indicates that the model correctly distinguishes harmless data and does not generate false alarms. On the other hand, the false negative count being only 6 indicates that the model mostly correctly detects data in the attack class, but very rarely fails to correctly detect some attacks. This matrix confirms that the model has high generalization ability, avoids overfitting, and is quite successful in correctly classifying the data. To better understand the model's performance, it must also be evaluated in terms of metrics other than accuracy. Table 5 presents the Accuracy, Precision, Recall, and F1 score values for all models in the study.

Table 5. The metrics of all models in the study.

Model	Accuracy	Precision	Recall	F1 Score
1	0.99937	1.0	0.99894	0.99947
2	0.99936	1.0	0.99872	0.99936
3	0.99941	1.0	0.99872	0.99936
4	0.99925	1.0	0.99851	0.99925
5	0.99925	1.0	0.99851	0.99925
6	0.99937	1.0	0.99894	0.99947
7	0.99904	1.0	0.99809	0.99904
8	0.99925	1.0	0.99851	0.99925
9	0.99947	1.0	0.99894	0.99947
10	0.99872	1.0	0.99745	0.99872
11	0.99936	1.0	0.99872	0.99936
12	0.99925	1.0	0.99851	0.99925

According to the metrics presented in Table 5, all models have generally performed quite well. The results in metrics other than Accuracy, which also deserve attention, reflect the effectiveness of the models more comprehensively. It is possible to say that FFNN models generally performed well. Although LSTM is an architecture designed to learn long-term dependencies, it stood out in this study with long training times and high loss values. This result caused LSTM models to perform weaker than FFNN models. Since most models had Precision, Recall, and F1 Score values close to 1.0, it is possible to say that each model performed excellently in accurately detecting DDoS attacks.

The FFNN architecture has generally attracted attention with its high accuracy and low loss values. Models 3 and 6 are the models with the best accuracy and F1 Score values, and these models have achieved quite good results with precision, recall, and F1 score values close to 1.0. This result shows that the false positive and false negative rates of the models are very low. This means that they show high sensitivity in correctly identifying DDoS attacks.

Although the LSTM architecture, as in Model 7, offers an approach to learning long-term dependencies, in this study, it generally stood out with long training times and high loss values. Although Model 7 achieved the lowest accuracy with an accuracy value of 99.04%, its precision and recall values are still very high. However, the slight drops in the F1 Score imply that the model cannot generalize effectively enough and that the training process is too long.

While there is no significant difference between FFNN and LSTM architectures, it has been observed that training times and optimization algorithms affect model performance. In particular, obtaining lower loss values in models using the Adam optimization algorithm demonstrates that this algorithm is an effective option for tasks such as DDoS attack detection.

On the other hand, models using SGD took longer training times, and performance required more epochs to reach the same accuracy level. These findings provide important insights into how various factors, from architectural choices to optimization algorithms, affect model performance. This study has yielded more efficient results compared to many other studies previously found in the literature. Table 6 presents a comparison of this proposed study and previous studies.

Table 6. Comparative Performance Analysis of Our Study and Previous Studies Using Various DDoS Datasets and Techniques.

Study	Dataset	Technique	Accuracy
Our Study	***DDoSGedik30K***	FFNN LSTM	***0.999***
[14]	BOUN DDoS Dataset	K-Means	98%
[15]	Kaggle DDoS Attack Network Logs Dataset	MLP	99%
[16]	CICIDS2017 & CICDDoS2019	RF	99.9%
[19]	Combined dataset	MLP	98.99%
[20]	CICIDS2017	MLP	99.9%
[21]	CICIDS2017	XGB-DDoS	0.999

[22]	CICDDoS2019	SVM	0.99
[23]	NSL-KDD	CNN	97.7%
[24]	UNSW-NB 15 & UNSW-2018 Botnet	RF	100%
[26]	Simulation data with Hping3	C4.5	98.8%
[30]	ISCX'12	3LSTM	98.41%

When examining Table 6, it becomes evident that this study has several advantageous aspects from various perspectives. Evaluating the datasets, it can be observed that while datasets commonly used in other studies, such as Kaggle, CICIDS2017, CICDDoS2019, and NSL-KDD, are popular, the DDoSGedik30K dataset provides a unique diversity as it is derived from various firewall configurations in real-world scenarios. Compared to other publicly available datasets, DDoSGedik30K stands out as a more original dataset.

Regarding the techniques employed, it is noted that other studies predominantly utilize methods like K-Means, MLP, RF, CNN, and SVM. It has been observed that RF and MLP achieved high accuracy rates in other studies. However, it is also evident that the LSTM and FFNN architectures used in this study easily reached similar accuracy levels. The richness of the DDoSGedik30K dataset and the correct hyperparameter optimizations have resulted in competitive outcomes compared to other studies. This success highlights the power of deep learning models in detecting complex attacks when compared to results obtained using traditional machine learning algorithms in other datasets.

In examining the work of Erhan and Anarım [14], it is seen that the BOUN DDoS dataset includes only TCP SYN flood attacks. This indicates that the study focuses solely on a single type of DDoS attack. In contrast, the DDoSGedik30K dataset encompasses a broader spectrum of attack types (DDoS, SQL, Port Scan). Traditional algorithms, such as K-Means, used in the relevant study may prove to be much less effective compared to the deep learning models employed in this study. Since network traffic involves a continuous data stream, models capable of learning sequential data, such as LSTM, may perform better than methods like K-Means or SVM. Lazaris and Prasanna's work [64] explains that LSTM yields better results compared to algorithms like K-Means and SVM. In other studies, models capable of time series analysis, such as LSTM, are sometimes either not used or less optimized. In the study conducted by Asarkaya et al. [15], various machine learning algorithms (KNN, SVM, MLP, RF) were tested using the DDoS dataset available on Kaggle. Although the Kaggle dataset is large, its common usage means its structure is generally well-known, and potentially over-optimized models may have been tested on this dataset. The DDoSGedik30K dataset, however, is original and based on real-world scenarios, incorporating simulation attacks, which enhances the model's generalization capability. While KNN, SVM, and MLP can demonstrate strong performance in many cases, they are not suitable for time series analysis. An architecture like LSTM performs better with sequential data such as network traffic. Therefore, the traditional methods used in the work by Asarkaya et al. may be inadequate in understanding time-dependent DDoS attacks.

Zekri et al. [26] achieved a 98.8% accuracy using the C4.5 decision tree algorithm in a cloud environment. However, this study has some weaknesses. The C4.5 algorithm was tested on data simulated through Hping3, but this dataset may not accurately reflect real-world network attacks. In contrast, the DDoSGedik30K dataset has been enriched with attack scenarios and vulnerabilities that closely resemble real-world conditions. Additionally, while decision trees may succeed in simpler data structures, deep learning models are better suited to learning complex attack patterns and time-dependent data. According to Table 5, the C4.5 algorithm does not possess as strong a learning capacity as LSTM and FFNN. Usmani et al.'s work also supports this assertion [65], showing that while decision trees may be faster, LSTM models achieve better accuracy in detecting complex attack patterns.

When examined for vulnerability simulation and realistic attack scenarios, this study is superior to the other studies compared. During the dataset creation, deliberately disabling firewall rules at times to create vulnerabilities was one of the most distinguishing factors of DDoSGedik30K. While most studies collect their datasets in static and secure environments, this study includes vulnerabilities that one might encounter in real life. This approach has increased the resilience of deep learning models against more diverse and realistic attacks.

In conclusion, this study provides significant superiority over other DDoS detection studies due to factors such as the originality, realism, and richness of the DDoSGedik30K dataset, the high performance of deep learning techniques like FFNN and LSTM, careful hyperparameter optimization, the use of models suitable for time series analysis, and the inclusion of a wider range of attack types.

7. CONCLUSION

This study examines deep learning models developed to detect DDoS attacks. It clearly demonstrates that all models achieved an accuracy rate exceeding 99.9% and performed effectively in detecting DDoS attacks. This success is of significant importance, especially in a critical area such as cybersecurity, as it aids in ensuring network security and in the early detection of potential threats. Since the accuracy rates are at the same level, metrics such as loss value, training time, optimization algorithm, and learning rate were evaluated to distinguish the models. These results serve as an important guide for selecting the most suitable model for various scenarios in security applications.

While there is no significant difference between FFNN and LSTM architectures, it has been found that LSTM requires longer training times due to its more complex structure. The ability of LSTM models to better learn long-term dependencies has balanced the need for more epochs. However, the longer training times of these models stand out as a factor that must be considered, especially in real-time applications.

The DDoSGedik30K dataset used in the study is considered an important resource for cybersecurity research. This dataset has been carefully prepared to cover various scenarios related to DDoS attacks, allowing the models to demonstrate performance close to real-world conditions. The high quality and diversity of the dataset have enhanced the effectiveness of the models and enabled more realistic results to be obtained during the training processes.

Future work may propose further expanding the dataset and diversifying it to include different types of attacks. The development of real-time DDoS detection systems and the integration of these systems into existing cybersecurity infrastructures should be one of the focal points of future studies. In conclusion, this study provides a significant contribution that demonstrates the effectiveness of deep learning techniques in the field of cybersecurity and offers new approaches for the detection of DDoS attacks. With its contributions to the literature and the dataset it presents, this study will serve as a guide for future research.

Acknowledgements

This work was supported by Scientific Research Projects Coordination Unit of Istanbul Gedik University, Project number "GDK202207-32".

Conflict of Interest

The authors declare no conflicts of interest.

Authors' Contributions

No	Full Name	ORCID ID	Author's Contribution
1	Şenay Kocakoyun Aydoğan	0000-0002-3405-6497	1, 4, 5
2	Turgut Pura	0000-0002-4108-8518	4
3	Zeki Çıplak	0000-0002-0086-3223	3, 4, 5
4	Anıl Yıldız	0000-0003-4607-6660	2
1- Study design 2- Data collection 3- Data analysis and interpretation 4- Manuscript writing 5- Critical revision			

References

[1] Aamir, M., Zaidi, S.M.A., 2019. DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation. International Journal of Information Security, 18: p. 761-785.
[2] Büyüktanır, B., et al. 2025. DDoS_FL: Federated Learning Architecture Approach against DDoS Attack. Pamukkale University Journal of Engineering Sciences, 31(6), 0-0.
[3] Carl, G., et al., 2006. Denial-of-service attack-detection techniques. IEEE Internet computing, 10(1): p. 82-89.
[4] Anli, Y.A., et al., 2024. DDoS detection in electric vehicle charging stations: A deep learning perspective via CICEV2023 dataset. Internet of Things, 28: p. 101343.
[5] Mitrokotsa, A. Douligeris, C., 2007. Denial-of-service attacks. Network Security: Current Status and Future Directions, p. 117-134.

[6] Özocak, G., 2012. DDoS Saldırısı ve Failin Cezai Sorumluluğu. Bilişim, 28: p. 23.
[7] Feily, M., Shahrestani, A., Ramadass, S., 2009. A survey of botnet and botnet detection. in 2009 Third International Conference on Emerging Security Information, Systems and Technologies. IEEE.
[8] Dayanandam, G., et al., 2019. DDoS attacks—analysis and prevention. in Innovations in Computer Science and Engineering: Proceedings of the Fifth ICICSE 2017. Springer.
[9] Zhu, Z., et al., 2008. Botnet research survey. in 2008 32nd Annual IEEE International Computer Software and Applications Conference. IEEE.
[10] Zhang, L., et al., 2011. A survey on latest botnet attack and defense. in 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications. IEEE.
[11] Efe, A., 2021. Yapay Zekâ Odaklı Siber Risk ve Güvenlik Yönetimi. Uluslararası Yönetim Bilişim Sistemleri ve Bilgisayar Bilimleri Dergisi, 5(2): p. 144-165.
[12] Mittal, M., K. Kumar, S. Behal, 2023. Deep learning approaches for detecting DDoS attacks: A systematic review. Soft computing, 27(18): p. 13039-13075.
[13] Atasever, S., Özçelık İ., Sağiroğlu, Ş. 2020. An Overview of Machine Learning Based Approaches in DDoS Detection. in 2020 28th Signal Processing and Communications Applications Conference (SIU). IEEE.
[14] Erhan, D., Anarım, E., 2020. Istatistiksel Yöntemler Ile DDoS Saldırı Tespiti DDoS Detection Using Statistical Methods. in 2020 28th Signal Processing and Communications Applications Conference (SIU). IEEE.
[15] Asarkaya, S., et al., 2021. DDOS SALDIRILARININ MAKİNE ÖĞRENİMİ ALGORİTMALARIYLA TESPİTİ. Tasarım Mimarlık ve Mühendislik Dergisi, 1(3): p. 221-232.
[16] Sharif, D.M., Beitollahi, H., 2023.Detection of application-layer DDoS attacks using machine learning and genetic algorithms. Computers & Security, 135: p. 103511.
[17] Stiawan, D., et al., 2020. CICIDS-2017 dataset feature analysis with information gain for anomaly detection. IEEE Access, 8: p. 132911-132921.
[18] Priya Devi, Johnson Singh, A. K., 2021. A machine learning approach to intrusion detection system using UNSW-NB-15 and CICDDoS2019 datasets. in Smart Computing Techniques and Applications: Proceedings of the Fourth International Conference on Smart Computing and Informatics, Volume 1. Springer.
[19] Ahmed, S., et al., 2023. Effective and efficient DDoS attack detection using deep learning algorithm, multi-layer perceptron. Future Internet, 15(2): p. 76.
[20] Sharif, D.M., Beitollahi, H., Fazeli, M., 2023. Detection of application-layer DDoS attacks produced by various freely accessible toolkits using machine learning. IEEE Access, 11: p. 51810-51819.
[21] Kareem, M.K., et al., 2023. Efficient model for detecting application layer distributed denial of service attacks. Bulletin of Electrical Engineering and Informatics, 12(1): p. 441-450.
[22] Salama, A.M., Mohamed, M.A., AbdElhalim, E., 2024. Enhancing Network Security in IoT Applications through DDoS Attack Detection Using ML. Mansoura Engineering Journal, 49(3): p. 10.
[23] Hussein, T., 2024. Deep Learning-based DDoS Detection in Network Traffic Data. International journal of electrical and computer engineering systems, 15(5): p.407-414.
[24] Manaa, M.E., et al., 2024. DDoS attacks detection based on machine learning algorithms in IoT environments. Inteligencia Artificial, 27(74): p. 152-165.
[25] Najar, A.A., et al., 2024. A novel CNN-based approach for detection and classification of DDoS attacks. Concurrency and Computation: Practice and Experience, 36(19): p. e8157.
[26] Zekri, M., et al. DDoS attack detection using machine learning techniques in cloud computing environments. in 2017 3rd international conference of cloud computing technologies and applications (CloudTech). 2017. IEEE.
[27] Ahda, A., et al., 2023.Information security implementation of DDoS attack using hping3 tools. JComce-Journal of Computer Science, 1(4).
[28] Koziol, J., 2003. Intrusion detection with Snort. Sams Publishing.
[29] Liao, H.-J., et al., 2013. Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications, 36(1): p. 16-24.
[30] Yuan, X., Li, C. X. Li., 2017. DeepDefense: identifying DDoS attack via deep learning. in 2017 IEEE international conference on smart computing (SMARTCOMP). IEEE.
[31] Dalalana Bertoglio, D. Zorzo, A.F., 2017. Overview and open issues on penetration test. Journal of the Brazilian Computer Society, 23: p. 1-16.
[32] Kim, Y.-E., Y.-S. Kim, and H. Kim, 2022. Effective feature selection methods to detect IoT DDoS attack in 5G core network. Sensors,. 22(10): p. 3819.
[33] Kumari, P., et al., 2024. Towards Detection of DDoS Attacks in IoT with Optimal Features Selection. Wireless Personal Communications, p. 1-26.
[34] Lyu, M.R., Lau, L.K., 2000. Firewall security: Policies, testing and performance evaluation. in Proceedings 24th Annual International Computer Software and Applications Conference. COMPSAC2000. IEEE.
[35] Khraisat, A., Alazab, A. 2021. A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. Cybersecurity, 4: p. 1-27.
[36] Fabbri, R., Volpe, F., 2013. Getting started with fortigate. Packt Publishing Ltd.
[37] Towidjojo, R., 2023. Mikrotik Kung Fu: Kitab 4. Jasakom.

[38] Ali, B.H., et al. 2022. Ddos detection using active and idle features of revised cicflowmeter and statistical approaches. in 2022 4th International Conference on Advanced Science and Engineering (ICOASE). IEEE.
[39] Lamping, U., E. Warnicke, 2004. Wireshark user's guide. Interface, 4(6): p. 1.
[40] Patro, S. and K.K. Sahu, Normalization: A preprocessing stage. arXiv preprint arXiv:1503.06462, 2015.
[41] Cabello-Solorzano, K., et al., 2023. The impact of data normalization on the accuracy of machine learning algorithms: a comparative analysis. in International Conference on Soft Computing Models in Industrial and Environmental Applications. Springer.
[42] Demircioğlu, A., 2024. The effect of feature normalization methods in radiomics. Insights into Imaging, 15(1): p. 2.
[43] Bebis, G., Georgiopoulos, M.,1994. Feed-forward neural networks. Ieee Potentials, 13(4): p. 27-31.
[44] Fine, T.L., 2006. Feedforward neural network methodology. Springer Science & Business Media.
[45] Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. in Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings.
[46] Graves, A., Graves, A., 2012. Long short-term memory. Supervised sequence labelling with recurrent neural networks, p. 37-45.
[47] Hochreiter, S., Schmidhuber, J.,1997. Long short-term memory. Neural computation, 9(8), p. 1735-1780.
[48] Medsker, L.R., Jain, L., 2001. Recurrent neural networks. Design and Applications, 5(64-67): p. 2.
[49] Medsker, L., Jain, L.C., 1999. Recurrent neural networks: design and applications. CRC press.
[50] Agarap, A.F., 2018. Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.
[51] Hara, K., Saito, D., Shouno, H., 2015. Analysis of function of rectified linear unit used in deep learning. in 2015 international joint conference on neural networks (IJCNN). IEEE.
[52] Hochreiter, S., 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02): p. 107-116.
[53] Srivastava, N., 2013. Improving neural networks with dropout. University of Toronto, 182(566), p. 7.
[54] Yin, X., et al., 2003. A flexible sigmoid function of determinate growth. Annals of botany, 91(3): p. 361-371.
[55] Abadi, M., et al., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.
[56] Ketkar, N., Ketkar, N., 2017. Introduction to keras. Deep learning with python: a hands-on introduction, p. 97-111.
[57] Bressert, E., 2012 .SciPy and NumPy: an overview for developers.
[58] Unpingco, J., 2021. Pandas, in Python Programming for Data Analysis. Springer. p. 127-156.
[59] Kramer, O., Kramer, O., 2016. Scikit-learn. Machine learning for evolution strategies, p. 45-53.
[60] Jais, I.K.M., Ismail, A.R., Nisa, S.Q., 2019. Adam optimization algorithm for wide and deep neural network. Knowl. Eng. Data Sci., 2(1), p. 41-46.
[61] Reyad, M., Sarhan, A.M., Arafa, M., 2023. A modified Adam algorithm for deep neural network optimization. Neural Computing and Applications, 35(23), p. 17095-17112.
[62] Jentzen, A., et al., 2021.Strong error analysis for stochastic gradient descent optimization algorithms. IMA Journal of Numerical Analysis, 41(1): p. 455-492.
[63] Bottou, L., 1991. Stochastic gradient learning in neural networks. Proceedings of Neuro-Nımes,. 91(8), p. 12.
[64] Lazaris, A., Prasanna, V.K., 2019.An LSTM Framework For Modeling Network Traffic. in 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM).
[65] Usmani, M., et al. 2022. Predicting ARP spoofing with Machine Learning. in 2022 International Conference on Emerging Trends in Smart Technologies (ICETST).
[66] Fawcett, T. (2006). An introduction to ROC analysis. Pattern recognition letters, 27(8), 861-874.