engineering proceedings

MDPI

Proceeding Paper

Transforming Educational Attendance Systems Through Advanced Facial Recognition and Machine Learning †

Issa Kamar [1] and Galina Momcheva [2,*]

[1] Department of Computer Science, Varna Free University, 9007 Varna, Bulgaria; issa.kamar@gmail.com
[2] Department of Computer Science, Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, 1113 Sofia, Bulgaria
* Correspondence: gmomcheva@math.bas.bg
† Presented at the 15th International Scientific Conference TechSys 2026—Engineering, Technologies and Systems, Plovdiv, Bulgaria, 14–16 May 2026.

Abstract

These days, learning management systems have been instrumental in changing the face of online education. Nonetheless, a significant shortcoming in distance education is the attendance reports. This paper addresses the need to streamline attendance control and administration in institutions, particularly the education sector, by introducing a novel real-time facial recognition-based tracking system. The facial recognition mechanism at the heart of the system is a state-of-the-art technology that enhances attendance tracking. The system can use sophisticated algorithms and machine learning approaches to achieve accurate identification and detection. Together with important features like dynamic charts and real-time insights, the system also includes a dashboard that is used to centralize real-time records for better tracking and managing attendance. This enhanced data visibility enables the administrator to make well-informed decisions. Additionally, because the dashboard can be dynamically altered to suit the requirements of administrators and institutions, admininistrators can quickly become accustomed to the system thanks to dashboard modification. Thus, a significant advancement in attendance management has been made with the PHP 8.3 and Python 3.12 based facial recognition attendance system. Because of its user-friendly interface and instantaneous insights, it can be an invaluable tool for educational institutions looking to improve the effectiveness of their daily operations and optimize attendance monitoring procedures.

Keywords: learning management systems; online learning; attendance; facial recognition; machine learning; Python; PHP

Check for updates

Academic Editors: Nikola G. Shakev, Valyo N. Nikolov, Nikolay R. Kakanakov and Sevil A. Ahmed-Shieva

Published: 3 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The field of machine learning (ML) technology is developing quickly in the digital age. Extensive studies conducted in this area demonstrate its broad applicability across numerous fields. ML, for instance, enables machines to learn in a variety of ways, including recognizing patterns in voice, images, and videos, identifying flaws in power transmission, and identifying patterns in communications networks [1]. Moreover, improvements in Internet of Things (IoT) technology make use of internet connections to facilitate autonomous wireless data transmission. This technology is widely used in single board computer gateways that are linked to cloud services and communications systems that use microcontrollers to transmit sensor data [2]. As a result, the fusion of ML with IoT allows

for the development of intelligent solutions, such as real-time attendance tracking, that could enhance efficiency and practicality across a range of industries [3].

Regular attendance is an essential prerequisite in a multitude of educational establishments, including schools, universities, and professional environments. Therefore, determining student presence in online classrooms is a crucial aspect to be considered when applying long distance and virtual learning [4]. Proxy attendance is a pervasive concern within educational environments, as students are recorded as present despite their absence, hence causing inaccurate attendance records for some individuals [3]. Likewise, accidental inaccuracies in the documentation can manifest in both educational and corporate settings [5]. Nevertheless, conventional approaches for recording attendance, such as verbally announcing names or recording signatures on paper, are laborious and monotonous for instructors [6]. Thus, implementing an artificial intelligence (AI) system that leverages facial recognition technology could be advantageous in tackling these difficulties. Such systems aim to autonomously detect and authenticate persons upon their appearance in the classroom, precisely record their presence throughout sessions, and securely store this data in a database for subsequent utilization [7].

In spite of the presence of various biometric security types, such as voice, fingerprints, and eye recognition, facial recognition has gained attention in today's technology advancements [8]. The distinctiveness inherent in an individual's facial features renders it a very suitable foundation for differentiating between individuals [5]. The implementation of facial recognition technology for attendance purposes in the educational industry presents notable advantages. For instance, it diminishes the probability of human errors, conserves time and resources by eliminating the necessity of manual attendance recording, and saves time for instructors and professors to focus more on educational duties than on administrative ones [3,5,9]. Therefore, by incorporating real-time facial recognition technology, the implementation of a machine learning-based attendance system significantly improves the efficacy, accuracy, and reliability of attendance management within educational institutions [5].

The present work focuses on the development of an attendance system that professors and administrators in educational institutions can use to recognize the presence or absence of students during online sessions without being obligated to spend much time on manual attendance tasks and administrative work. To do so, the development of this system will be utilizing Python and PHP, with a focus on facial recognition technology. The Python section includes the tasks of face detection, data augmentation, and data storage, while the PHP section is responsible for database management and the provision of a web-based dashboard for monitoring attendance statistics.

After presenting an introduction of the topic, followed by the motivations guiding the development of this work, the subsequent sections of this article will be composed of various parts, including a brief literature review focusing on different machine learning algorithms used to develop face recognition-based systems, in order to explore solutions proposed by other authors and to assess the limitations of their methodologies. Further, a detailed methodology of the proposed system will be presented, followed by the implementation of the face recognition system and the results of the performed experiments. Lastly, a thorough discussion of the results will be held, where recommendations and potential solutions will be presented for future work in this field.

In summary, the main objectives of this work are the following:

- Implement facial recognition technology, based on machine learning algorithms, especially Python and PHP, to accurately identify individuals' attendance in the educational online learning context.
- Test the accuracy and efficiency of the developed system.

https://doi.org/10.3390/engproc2026150101

- Develop an admin dashboard that helps the visualization of students' attendance.

1. Related work

Prior to the adoption of facial recognition systems for student attendance, various other technologies were employed to enhance the accuracy and efficiency of monitoring student presence in educational settings. One such technology was detailed in the study of Bhise et al. [10], who explored the use of "near field communication" (NFC) technology integrated with an embedded camera on mobile devices. NFC, a type of short-range wireless communication that involves an active device generating an electromagnetic field to interact with passive devices, was utilized to facilitate the attendance process [10]. In their system, students were provided with uniquely programmed NFC tags containing their identification numbers. During class, lecturers equipped with NFC readers would require students to swipe their tags near the reader to register their attendance, which was then automatically updated in the school's database. However, despite this NFC-based system having drawbacks such as the risk of impersonation where one student could potentially swipe in for another [10], this system has become inapplicable in the era of online and virtual learning.

Consequently, the proliferation of virtual education through diverse learning management platforms has spurred scholarly interest in leveraging machine learning frameworks for automated facial detection. In this regard, Sinha et al. [11] introduced a monitoring infrastructure utilizing facial recognition, emphasizing that conventional biometric hardware is frequently inefficient, cumbersome to manage, and susceptible to the pervasive issue of proxy attendance registration.

In addition, Raj & Vadivel [6] developed a computer-based attendance management system using computer vision technology capable of accessing over 2500 optimized algorithms for various tasks, including similarity searches. They introduced an OpenCV 4.9.0-based face recognition methodology that integrates several components: a camera to capture input images, an algorithm to detect and encode faces from these images, and a process to mark attendance directly into a PDF spreadsheet. The system is underpinned by a training database consisting of authorized students' faces. These faces are stored as cropped images in the database alongside respective labels. The "local binary pattern histogram" (LBPH) algorithm is employed to extract features from these images for accurate recognition. Once operational, the system can identify authorized students in real-time by matching detected faces with the database entries. Attendance is recorded by marking "present" for recognized faces and "absent" for unrecognized ones directly into a spreadsheet. Raj & Vadivel concluded that this system surpasses traditional attendance methods in terms of efficiency and accuracy, positing it as a valuable tool for any organizational context, including educational industries [6].

In the same regard, Al-Mihaidhri & Hussain [12] developed a face recognition machine learning-based system for attendance using Raspberry pi 3B+ with OpenCV 4.9.0 4.9.0/Python libraries and the recognizer algorithm. Notably, the system showed an accuracy rate of 98.5% [12]. Moreover, Deepan et al. [9] deployed "convolutional neural network" (CNN) for face recognition and attendance monitoring, utilizing hidden image elimination, highlighting a high accuracy and efficiency [9]. Moreover, Patil et al. [13] developed an automatic system for students' attendance using facial recognition. In this model, the Viola–Jones technique was used for students' face detection, whereas "linear discriminate analysis" (LDA), "k-nearest neighbors" (KNN) and "support vector machine" (SVM) were used for facial recognition. The experimentations showed a higher accuracy than other existing systems [13].

https://doi.org/10.3390/engproc2026150101

In summary, as shown in this section, there are a variety of algorithms that can be used for the development of facial recognition machine learning-based systems, each characterized by its own advantages and drawbacks, as well as its accuracy and efficiency.

2. Methodology

The methodology of this study focuses on the development and implementation of a facial recognition-based attendance system, integrating Python and PHP to streamline the process of recording and managing attendance using advanced computing and web technologies. It involves a sequence of steps.

In the first step, face detection and landmark saving is done. This step consists of the addition of a new student name to the dataset. After camera opening, face detection is done using OpenCV 4.9.0. By the deployment of Haar cascades, a resizing of the images collected is done. The collected data is then sent to PHP to save the landmarks on the database. To note, this phase is a one-time phase that is performed when an administrator is willing to add a new student to the classroom.

Phase two consists of several steps. The first step includes the compilation of a comprehensive dataset of facial images using the Python 3.12 OpenCV 4.9.0 library that works on real-time recognition through activating the camera and collecting facial landmarks along with the respective student's name. The dataset comprises photos depicting individuals that the system anticipated to detect and identify, including diverse lighting conditions, stances, expressions, and background in order to augment its resilience. Subsequently, the images, detected through Haar cascades, undergo preprocessing procedures aimed at enhancing their quality and consistency by resizing them and capturing multiple frames per individual to build the dataset. These procedures include resizing, normalization, and noise reduction.

After the preprocessing stage, a face detection algorithm is utilized to accurately identify and isolate certain facial regions present in the images. Following the detection of faces, the process of feature extraction is conducted. This includes local data storage and transmission managed by libraries such as NumPy and Pickle. The retrieved data is then transmitted and stored in a PHP script via a POST request for database insertion. The aforementioned database serves as a point of reference for the recognition system.

Additionally, recognition algorithms such as OpenCV 4.9.0 and Scikit-learn, which deploys a pre-trained KNN classifier that performs real-time face recognition, are utilized to conduct a comparison between the characteristics of an input image and those stored in the database. The process aids in either verifying or identifying the features. Thus, the MediaPipe library is employed to provide holistic detection capabilities. Attendance data captured is then forwarded to the PHP side for processing.

The processing phase includes the usage of an admin dashboard interface, connecting to a MySQL dataset. This dashboard is specifically engineered to dynamically populate with the names of students and present attendance statistics. The dashboard utilizes Chart.js to create dynamic charts that display real-time data visualization. These charts are generated depending on the attendance data obtained from the database. This script plays an important role in retrieving attendance data according to predetermined criteria, which can be either a specific student or all students. This data is then processed and returned in JSON format, enabling dynamic changes on the dashboard.

The following developed framework as shown in Figure 1 summarizes the different steps aforementioned:

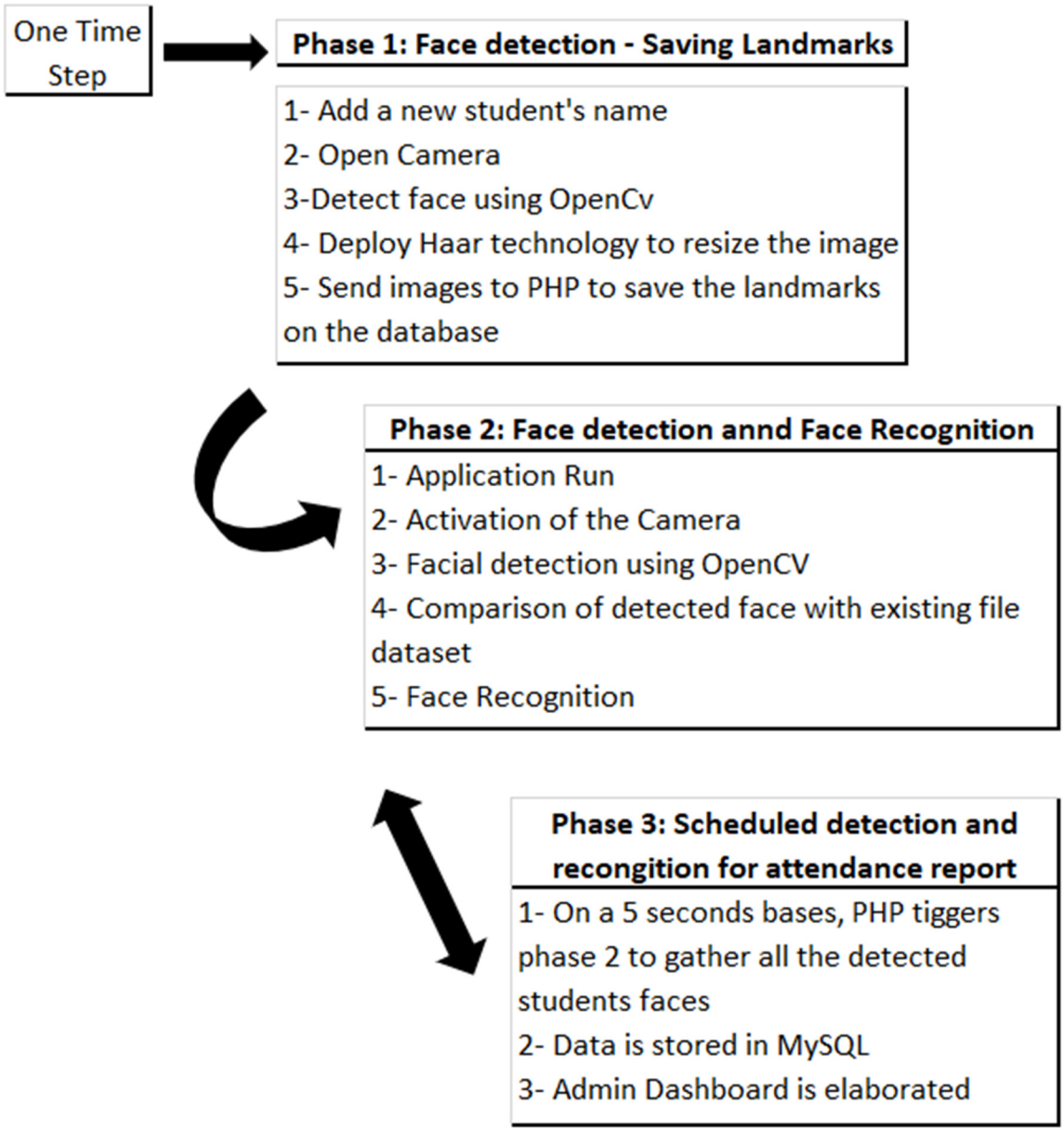


Figure 1. Framework of the development of the attendance system. Curved Arrow (Phase 1 → Phase 2): Initialization Flow—Shows the transition from one-time setup (saving landmarks) to live execution. Double-Headed Arrow (Phase 2 ↔ Phase 3): Recurring Trigger Loop—Shows Phase 3 periodically triggering Phase 2 every 5 seconds for attendance logging.

3. Results and Discussion

The implementation phase results in the development of an admin dashboard (Figure 2) that serves as a centralized platform for managing and monitoring the attendance system. It mainly comprises several components that dynamically adapt based on the values of the URL parameter. As shown in Figure 2, the feature of a student selection dropdown allows the user to choose a specific student or to include the data of all students. The dropdown is populated dynamically from the MySQL database, ensuring accurate and up-to-date student information. Therefore, as the administrator changes the selection of students, the values displayed in the chart vary accordingly.

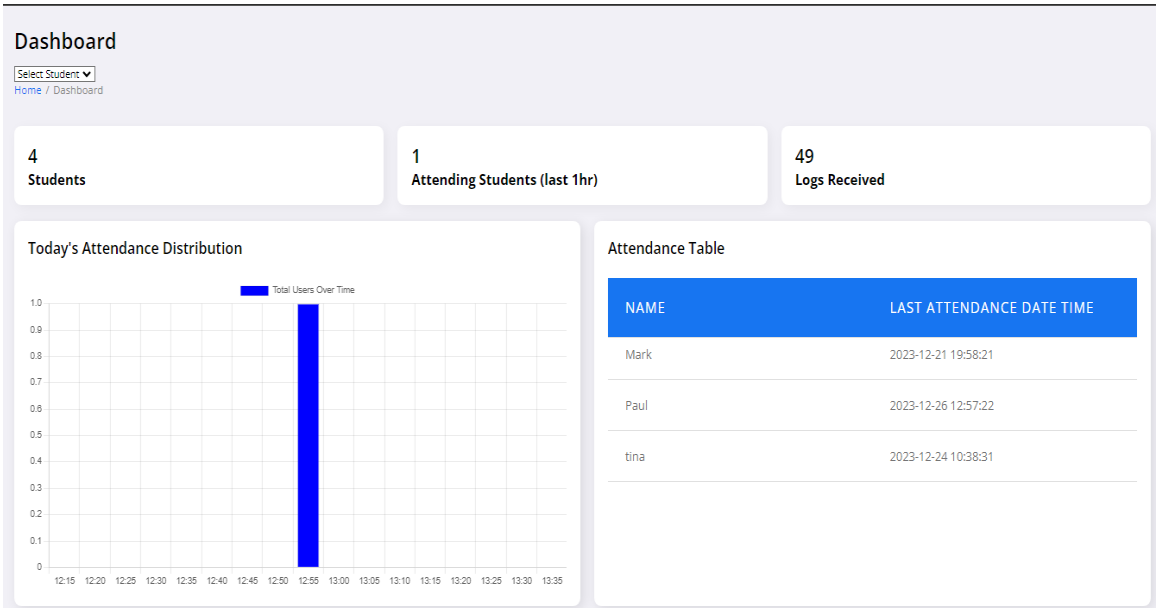


Figure 2. Visualization of the dashboard of the admin.

Additionally, as shown in Figure 2, the dashboard created visualizes the real-time statistics, the total number of students, the attending students within the last hour and the total number of attendance logs received. This shows the dynamic behavior of the system because the values shown are retrieved from the MySQL database and updated in real-time. Hence, the statistics are influenced by the selected student in the dropdown.

Another feature of this dashboard is the daily attendance distribution chart that provides a visual representation of the attendance distribution throughout the day (Figure 3). This chart, which adapts its data based on the selected student in the dropdown, dynamically updates its information and fetches data from the MySQL database.

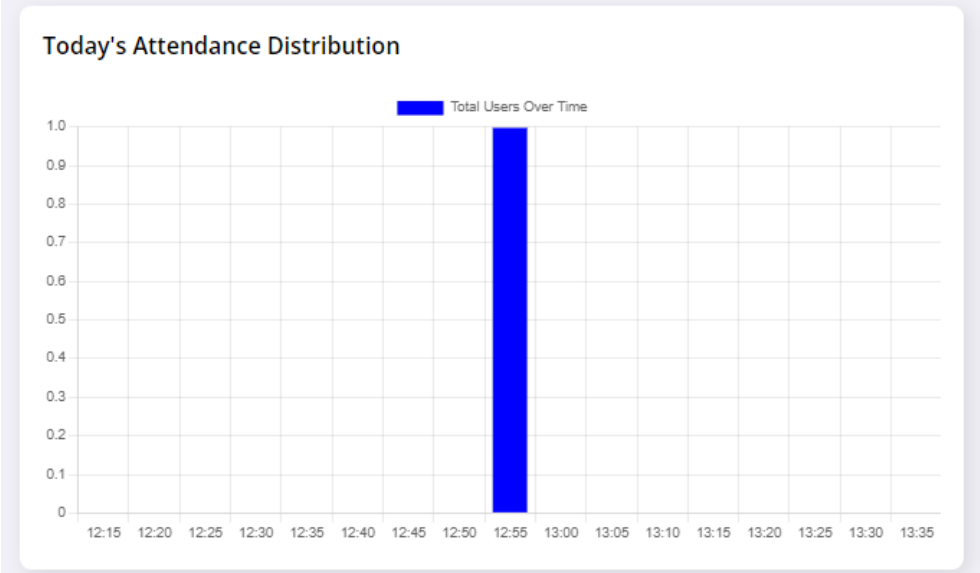


Figure 3. Today's attendance distribution chart.

Subsequently, the dashboard displays a table showing the last attendance date and time for each student, which is also dynamically generated based on the selected student. Thus, the table adapts its content according to the selected student in the dropdown (Figure 4).

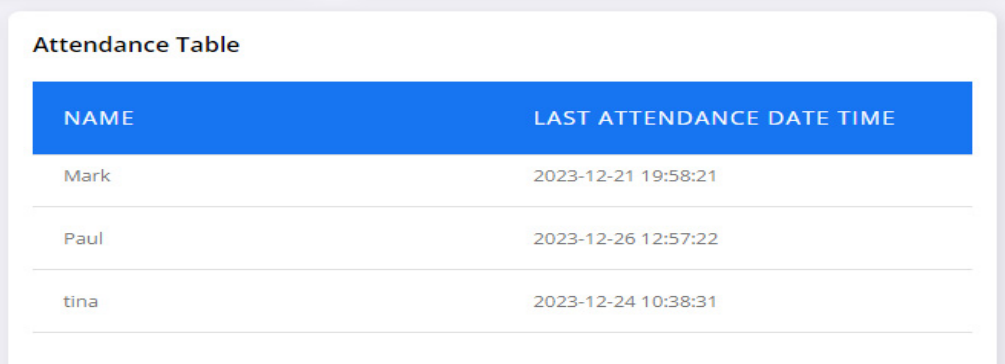


Figure 4. Attendance Table.

In summary, the development and implementation of the attendance system using Python and PHP helps provide the educational institutions with a real-time system that helps them decrease their administrative tasks in a very easy way. Furthermore, the admin dashboard provides a user-friendly interface for monitoring attendance statistics, because its components dynamically adjust on the selected student, providing real-time insights and facilitating effective attendance management. In comparison with traditional methods used for students' attendance, such as calling out names and recording signatures on papers and so on, this system seems to be time-saving and efficient, helping professors focus more on their educational duties than on the administrative tasks they are assigned to do.

4. Conclusions

Introducing facial recognition technology into LMS marks a major leap forward in overcoming long-standing challenges of tracking attendance in educational settings. In this work, we have developed an attendance system utilizing Python and PHP, with a focus on facial recognition technology. The attendance management system is thus an important innovation for many institutions and educational organizations that still rely on traditional methods for tracking attendance. As shown throughout this work, this attendance system is user-friendly, time saving and provides a real-time visualization for the presence of students, not only in the first few minutes of the session but throughout the entire educational session given. The development of this system presents various other advantages as well, as it is secure, more reliable, and accessible, which benefits educators and students and decreases the burden of administrative work. Moreover, it does not require sophisticated or specialized hardware for its implementation. This evolution promises to improve operational efficiency and enrich the educational experience, allowing educators and administrators to focus more on student engagement.

2. Related Work and Research Contributions

As we have already discussed, the topic of DDoS detection has seen a lot of interest in the application of ML and DL. This section highlights the key contributions of this work in comparison with the existing solutions and sheds light on the most well-known ML/DL-based DDoS detection systems.

Gunawan et al. [14] worked on a lightweight directory access protocol (LDAP) attack, which is just one type of DDoS attack on the application layer beside the hypertext transfer protocol (HTTP) and DNS applications. By using a CICDDoS 2019 dataset that includes both fake and real-time DDoS attacks and applying two DL approaches to resolve data imbalances for classifying and DDoS/DoS attacks, these approaches are the adaptive synthetic (ADASYN) approach and the synthetic minority oversampling technique (SMOTE), and the comparison results between ADASYN and SMOTE show that the F1 value for

benign traffic detection was 0.9983 and for attack traffic detection was 0.9983, while SMOTE shows that the F1 benign traffic detection was 0.9997 and attack traffic detection was 0.9997.

Lee & Kim [15] presented a method that can differentiate between App DDoS attacks and regular traffic without considering the order of the sequences. They extracted attributes from web page request sequences, which represent pages of interest, user activity and the breadth and intensity of their interests. These attributes were then organized in a matrix format and analyzed using principal component analysis (PCA) to model browsing patterns. The criterion for distinguishing App DDoS attacks from traffic was based on the reconstruction error of the PCA models. To validate their approach, they utilized web logs from three websites: a shopping site, a community website and an educational website. The proposed method achieved a detection rate (DR) of 86.7% for App DDoS attacks.

Xie B et al. [16] developed a communication model for detecting application layer distributed denial of service (AL DDoS) attacks. The model is based on the explicit duration recurrent network (EDRN). The approach involves training the model, then using it to detect AL DDoS attacks. To evaluate its effectiveness, they conducted experiments using datasets from a campus network and the CICDDoS2019 dataset. The study's findings demonstrate that, in terms of accuracy, F1, recall, and loss values, EDRN outperforms a number of well-known recurrent neural networks. On the CICDDoS2019 dataset, the proposed model detects HTTP DDoS assaults with an accuracy of 0.996, F1 of 0.992, recall of 0.993, and loss of 0.041.

The authors of Bhayo J. et al. [17] demonstrated an ML method to identify DDoS attacks in an SDN-WISE IoT controller for software-defined networking. They put up a testbed environment to simulate the generation of DDoS attack traffic, inserted a machine learning-based detection module into the controller, and transformed the data into a dataset. The SDN-WISE controller's ML DDoS detection module classifies SDN-IoT network packets using the NB, DT, and SVM algorithms. For NB, SVM, and DT, the suggested framework had accuracy rates of 97.4%, 96.1%, and 98.1%, respectively.

In Salama et al. [18] the authors used the Canadian Institute of Cybersecurity dataset CICDDOS2019 and analyzed it with K-NN, DT, LR, and NB, then each method's accuracy and precision value were calculated. According to the results, the highest accuracy and precision values were registered by the LR algorithm (99.9%) and the lowest accuracy and precision values were registered by the DT algorithm (99.8%). Therefore, relative to K-NN, DT, and NB, the most effective way is to classify the DDoS attacks by using logistic regression.

Estupiñán Cuesta et al. [19] worked on application layer DDoS attacks by analyzing the characteristics of incoming packets, including the number of IP addresses sent, the size of HTTP frame packets, the number of IP addresses using proxy IP and constant mappings of ports. They analyzed client behavior in public attacks using the CTU-13 dataset, real weblogs' private datasets, and experimentally created datasets from DDoS attack tools like Xerex, Slow Lairs, Golden Eyes and Hulk. A multilayer perceptron (MLP) and a deep learning algorithm were used to evaluate the effectiveness of metrics-based attack detection. Results show that the proposed MLP classification algorithm has an efficiency of 98.99% in detecting DDoS attacks.

Chen et al. [20] proposed a method for DDoS attack detection based on the LSTM model, using two different datasets, the CIC-DDoS2019 and the CIC-IDS2017 dataset. The detection method gave a high accuracy level in detecting DDoS attacks, but when they tested the same method with various adversarial DDoS attacks generated using GAN, the results show the inefficiency of the LSTM-based detection method. The suggested detection model is effective and precise in recognizing GAN-generated adversarial DDoS traffic, with a detection ratio ranging between 91.75% and 100%, according to their experimental findings.

Machine learning models were developed by Osa & Alhassan [21] using the CICIDS 2017 dataset to train and test attack detection. The same dataset was used to test all three techniques, and it was discovered that while linear regression was the fastest, it was also the least accurate. While random forest is the fastest of the three, it is also the most accurate. Finally, decision trees are the best method for these kinds of issues since they offer a fair mix between speed and accuracy.

This article contributes to the research work in IDSs by:

- Creating a special labeled dataset based on webservers' access logs by using eight of the most public and most used DDoS attack tools.
- Creating other types of attack datasets, including SQL injection, broken authentication, sensitive data exposure, broken access control, security misconfiguration, cross-site scripting (XSS) and insecure deserialization.
- Applying different ML and DL algorithms in training the model to classify website visits as benign or DDoS attack with an accuracy of 99% registered by the LSTM.
- The LSTM model also registers an accuracy of 90.99% in classifying website visits as benign traffic or as other types of attack traffic.

In summary, we have surveyed different types of ML and DL approaches used to detect DDoS attacks and different datasets used to train the models and the accuracy scores registered by each proposed method. This section provides a comprehensive overview of recent studies in the field of DDoS detection using ML and DL techniques. Gunawan et al. [14] addressed LDAP attacks using DL methods, demonstrating high F1 values for benign and attack traffic detection. Lee & Kim [15] introduced a novel approach to differentiate App DDoS attacks from regular traffic with an 86.7% detection rate. Xie B et al. [16] developed an effective communication model based on EDRN, outperforming other recurrent neural networks in terms of accuracy on the CICDDoS2019 dataset. Bhayo J. et al. [17] applied ML to SDN-WISE IoT controllers, achieving high accuracy rates using NB, DT, and SVM algorithms.

Salama et al. [18] analyzed the CICDDOS2019 dataset with various algorithms, with logistic regression showing the highest accuracy and precision. Estupiñán Cuesta et al. [19] employed MLP and deep learning algorithms for efficient DDoS attack detection, with a 98.99% efficiency rate. Chen et al. [20] proposed an LSTM-based detection method with high accuracy, although it exhibited some inefficiency in recognizing GAN-generated adversarial DDoS traffic. Overall, this section highlights the diversity of approaches and datasets used in DDoS detection and contributes by creating specialized labeled datasets and achieving high accuracy in classifying website visits as benign or DDoS attacks using LSTM models.

3. Dataset and Data Preprocessing

3.1. Dataset

DDoS attacks that target the application layer and that bypass the traditional firewalls and IDS are registered in the webservers' access log. For this we created a lab consisting of an Apache webserver, domain name, and two devices that have two different IPs and equipped with different DDoS attack tools like: (1) Goldeneye, (2) HULK, (3) Raven Storm, (4) Torshammer, (5) CC-Attack, (6) MHDdos, (7) DDosX, and (8) Eagle Eye. Then for two days we ran the DDoS attack using the HTTP requests against the webserver, and simultaneously during the attack period the domain name was receiving normal benign traffic from different resources. As a result, we got a log file containing 344,054 records and 7 columns; 62.8% of the records are for DDoS attack traffic and 37.2% of the records are for benign traffic. And the seven columns that any HTTP request registers in the access log file as shown in Figure 5 are:

	ip	time	request	status	size	referer	user_agent
0	2.22.202.102.98	2023-07-11 09:07:42+03:00	GET / HTTP/1.1	200	17406	http://yoursite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)
1	2.22.202.102.98	2023-07-11 09:07:42+03:00	GET /wp-includes/css/dist/block-library/style.css	200	12736	http://yoursite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)
2	2.22.202.102.98	2023-07-11 09:07:42+03:00	GET /wp-includes/css/classic-themes.min.css?ver=	200	210	http://yoursite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)
3	2.22.202.102.98	2023-07-11 09:07:42+03:00	GET /wp-content/themes/jannah/assets/css/base.css	200	8564	http://yoursite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)
4	2.22.202.102.98	2023-07-11 09:07:42+03:00	GET /wp-content/themes/jannah/assets/css/helper.css	200	7754	http://yoursite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)

Figure 5. Sample data from webserver access log file.

The data includes the IP address of the client that made the request, (2) the date and time of the request, (3) the request type and resource being requested, (4) the HTTP response status code, (5) the size of the object returned to the client, (6) the HTTP referrer, which represents the address from which the request for the resource originated and (7) the user agent, which identifies information about the browser that the client is using to access the resource.

3.2. Data Preprocessing

Checking the log records, we find that we have two columns containing null values. The size field has 13,129 (3.8%) records with null values and we filled these null values by the mean of the size column. The second field is the referrer field that has 121,307 (35.25%) records with null values and we filled it by the fixed expression "unknown referrer".

To be able to apply supervised ML algorithms on this dataset, we changed the source IP address field into the label column where we changed the two IP addresses that we used in the DDoS attack into value 1 and the other IP address that generates normal traffic into value 0, and we changed the column name from IP to "is_attack", so the new dataset we obtained has the shape shown in Figure 6.

	ip	time	request	status	size	referer	user_agent
0	203.0.113.42	2024-03-14 09:22:05+03:00	GET /wp-includes/css/dist/block-library/style.min.css HTTP/1.1	200	13021	http://yourwebsite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)
1	185.220.101.7	2024-03-14 10:22:12+03:00	GET /wp-content/themes/astra/assets/css/base.min.css HTTP/1.1	200	9102	http://yourwebsite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)
2	203.0.113.42	2024-03-14 10:23:21+03:00	GET /wp-content/uploads/2023/06/banner.jpg HTTP/1.1	200	45210	http://yourwebsite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)
3	192.168.1.150	2024-03-14 10:22:23+03:00	GET / HTTP/1.1	200	17842	http://yourwebsite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)
4	185.220.101.7	2024-03-14 10:24:41+03:00	GET /wp-content/themes/astra/assets/css/base.min.css HTTP/1.1	200	9102	http://yourwebsite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)
5	45.134.26.11	2024-03-14 10:25:00+03:00	GET /wp-content/uploads/2023/06/banner.jpg HTTP/1.1	200	45210	http://yourwebsite.info/	Mozilla/5.0 (Windows NT 10.0; Win64; x64)

Figure 6. Sample data from the labeled dataset.

Different techniques have been applied to change the non-numeric fields like the time, request, referrer and the user_agent into numeric fields, and then the StandardScaler function that helps to ensure that the features have similar scales and units as shown in Figure 7 is used to normalize the features of a dataset because it plays a crucial role in improving the performance of many ML algorithms.

And the last step in data preprocessing was splitting the dataset into training, testing and validation portions, where 72% of the dataset was used for training (248,578 records), 15% of the dataset was used for testing (51,609 records) and the remaining 13% (43,867 records) was used for validation, we applied data splitting by taking into consideration the necessity of stratified sampling or stratified splitting of a dataset that is used to ensure that the distribution of classes (or labels) in the training and testing subsets closely resembles the distribution of classes in the original dataset; thus, we obtained the following distribution of datapoints among the training (156,191 attack records, 92,387 benign

records), testing (32,428 attack records, 19,181 benign records), and validation (27,566 attack records, 16,301 benign records) parts.

There are 344,054 rows and 7 columns in this data frame

	is_attack	time	request	status	size	referer	user_agent
0	0	−1.161599	−0.812800	−0.221011	−0.761575	−1.052026	−0.157382
1	0	−1.161599	1.556488	−0.221011	−0.862339	−1.051839	−0.157382
2	0	−1.161599	1.556410	−0.221011	−1.132612	−1.051839	−0.157382
3	0	−1.161599	1.554349	−0.221011	−0.952358	−1.051839	−0.157382
4	0	−1.161599	1.554427	−0.221011	−0.969836	−1.051839	−0.157382

Figure 7. Sample data from the normalized dataset.

3.3. Features Correlation Heatmap

For the best understanding of our dataset features we used the correlation heatmap with colors representing the strength and direction of correlations between features. Positive correlations are shown in warmer colors (reds), while negative correlations are shown in cooler colors (blues), a value close to 1 indicates a strong positive correlation, close to −1 indicates a strong negative correlation, and close to 0 indicates a weak or no correlation. The heatmap graph shown in Figure 8 can help us identify patterns and relationships between different features in our dataset.

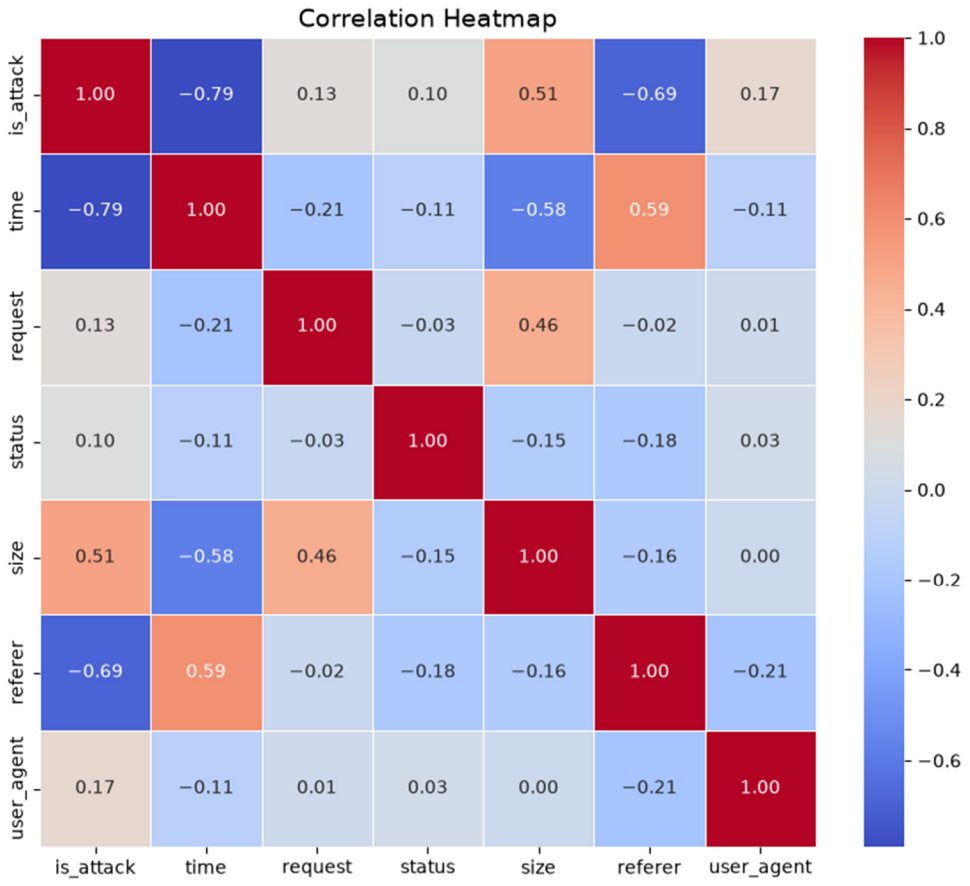


Figure 8. Features correlation heatmap.

4. Training the Models Using ML and DL Classification Algorithms

4.1. ML Algorithms

Machine learning classification algorithms are a type of supervised learning method used for predicting categories or labels for input data based on their features. Classification is one of the most common tasks in machine learning, and there are several algorithms available for this purpose. For this purpose, we used six of the most widely used classification algorithms like SVM, NB, LR, KNN, DT and RF, and the general accuracy score, accuracy score in detecting the benign traffic, accuracy score in detecting the DDoS traffic, precision, recall, and F1-score of the six algorithms are summarized in Table 1.

Table 1. Algorithm results comparison table.

	Algorithm	Accuracy Score	Benign Accuracy	Attack Accuracy	Precision	Recall	F1-Score
1	Random Forest	98.4	96.37	99.77	97.72	99.78	98.74
2	Decision Tree	97	96.47	97.56	97.73	97.5	97.61
3	Naïve Bayes	91.05	87.76	93.04	92.52	93.3	92.91
4	SVM	94.22	85.45	99.86	91.69	99.85	95.59
5	Logistic Reg	94.06	86.81	98.92	92.2	98.92	95.44
6	KNN	98.44	96.25	99.9	97.65	99.93	98.77
Average		95.53	91.52	98.18	94.92	98.21	96.51
Max		98.44	96.47	99.9	97.73	99.93	98.77
Min		91.05	85.45	93.04	91.69	93.3	92.91

We find among the ML algorithms that the RF and KNN algorithms registered the best accuracy score of 98.4%; furthermore, we noticed that the NB algorithm registered the worst accuracy score of 90.99%.

4.2. DL Algorithms

Deep learning classification algorithms are a subset of machine learning algorithms that specifically use deep neural networks for solving classification tasks. Deep neural networks are composed of multiple layers of interconnected nodes (neurons) and have shown remarkable success in various classification problems. In this research we used two structures of neural networks to create a classification model to classify webserver traffic as normal, benign, or DDoS attack traffic.

4.2.1. Simple NN

The first neural network structure is a simple neural network consisting of an input layer, three hidden layers with dropout and batch normalization, and an output layer for binary classification. The ReLU activation function is used in the hidden layers to introduce non-linearity, and dropout and batch normalization are applied to improve generalization and training stability. This structure as shown in Figure 9 is commonly used in deep learning for various classification tasks.

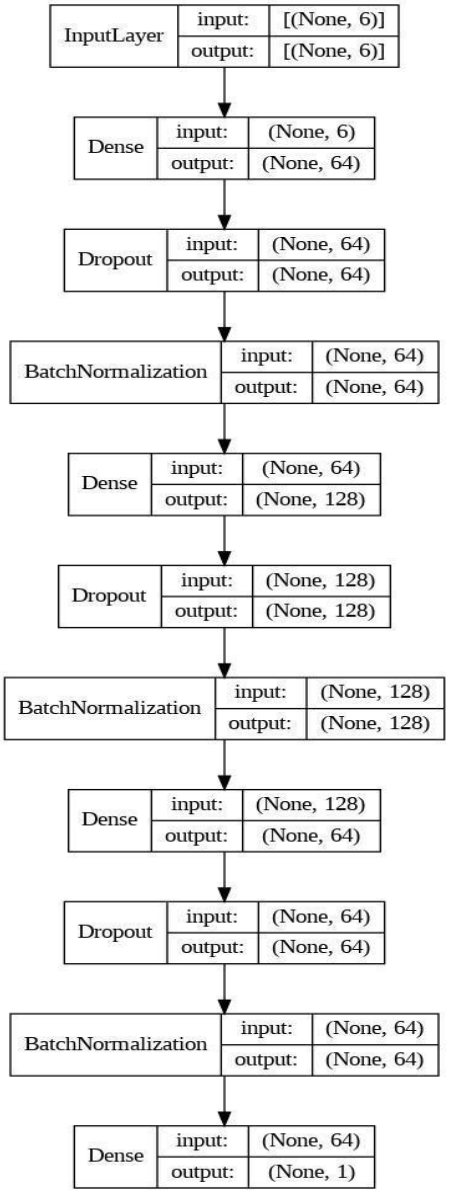
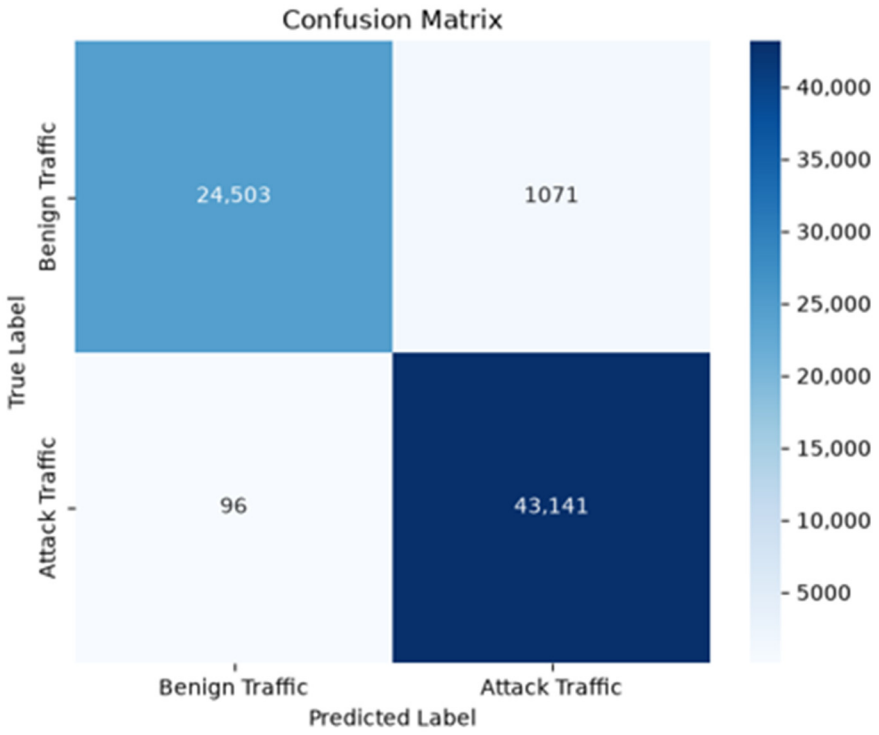


Figure 9. Simple neural network model summary.

Running this simple NN model gives a detection accuracy, precision, recall, and F1-score of 98%, which is somewhat similar to the results we obtained from the RF and KNN algorithms, as shown in Figure 10.

In the model training we applied the issue of early stopping that monitors the model's performance on a validation set during training and stops when the performance stops improving to avoid overfitting and, as a result, we find that the model stops the training process when it reached epoch 34, and instead of using a fixed learning rate, we implemented a learning rate scheduling to adapt the learning rate during training and it can help the model converge faster and improve performance, starting with a lr = 0.001, and at epoch number 11 the lr changed automatically into 0.00001, as shown in the graph in Figure 11.



Classification Report Table:

	precision	recall	f1-score	support
Benign Traffic	1.00	0.96	0.98	25,574
Attack Traffic	0.98	1.00	0.99	43,237
accuracy			0.98	68,811
macro avg	0.99	0.98	0.98	68,811
weighted avg	0.98	0.98	0.98	68,811

Figure 10. Confusion matrix and classification report for the simple NN model.

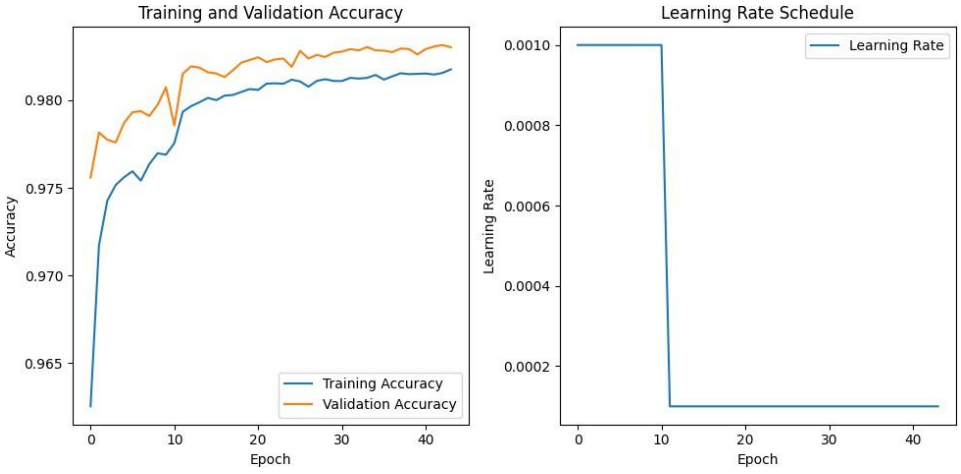


Figure 11. Training and validation accuracy learning rate for the simple NN model.

4.2.2. LSTM Model

LSTM, which stands for long short-term memory is a type of recurrent neural network (RNN) that is particularly effective in dealing with data that is sequential or time-based. Also LSTMs can be helpful when dealing with datasets that contain mixed numeric and text fields. Unlike simple networks, LSTM networks are equipped with specialized units known as "memory cells", enabling them to effectively capture and retain information over extended sequences. These memory cells consist of three gates: the input gate, output gate and forget gate. They regulate the flow of information within the network. As a result, LSTM networks possess the ability to handle dependencies that span ranges within the data. For this purpose, in our research, we developed an LSTM model with a structure that is shown in Figure 12, where this model's architecture combines the strengths of LSTM for text data, Conv1D for local feature extraction, and numeric inputs for a mixed data classification task. It is a comprehensive model designed to handle both text and numeric data effectively for binary classification.

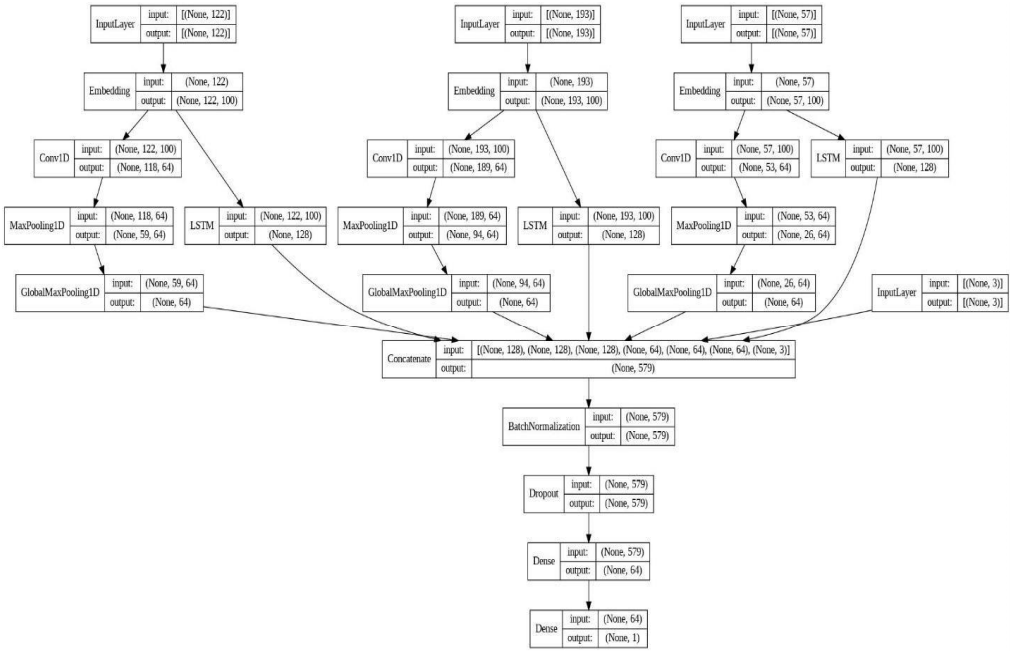


Figure 12. LSTM model summary.

Before creating the model architecture, we changed the text fields into numeric by using the Tokenizer function, which is typically used in natural language processing (NLP) and text processing tasks. The primary purpose of the Tokenizer is to preprocess and tokenize text data, breaking it down into individual words or tokens so that the Tokenizer takes a text corpus as input and separates it into individual tokens. In addition to tokenization, the Tokenizer also builds a vocabulary. It keeps track of all unique tokens encountered in the text corpus and assigns a unique integer (token ID) to each token. This vocabulary can be later used for various NLP tasks, such as text classification or sequence generation. And at the end, we applied the text to sequence conversion, knowing that the Tokenizer can convert a sequence of text data into a sequence of integers, where each integer corresponds to a token's ID in the vocabulary. This is a crucial step for training ML models, as they typically work with numerical data.

Here is a summary of the key components and operations in the model:

- Embedding layers:
  Three separate embedding layers are defined for three text inputs, referrer, request, and user_agent. Each embedding layer converts integer-encoded words into dense vectors of length 100.
- LSTM layers:
  Three LSTM layers are used for the text data; each LSTM layer has 128 units (or cells). LSTM layers are designed to capture sequential patterns in the text data.
- Conv1D layers:
  For each text input, a Conv1D layer is added to capture local patterns in the sequences. Each Conv1D layer has 64 units and uses a kernel size of 5. The activation function used is ReLU.
- Pooling layers:
  MaxPooling1D layers are applied to the outputs of the Conv1D layers for dimensionality reduction. The pooling size is set to 2.
- Global MaxPooling:
  Global MaxPooling1D layers are used to extract the most important information from the pooled data, reducing it to a fixed-size representation.
- Numeric input:
  Numeric features are incorporated into the model using a separate input layer with shape 3.
- Concatenation:
  All the output tensors from the LSTM, Conv1D, and Global MaxPooling layers are concatenated along with the numeric input. This creates a combined feature representation.
- Batch normalization:
  Batch normalization is applied to the concatenated features. This technique helps stabilize and accelerate the training process by normalizing the inputs during each mini-batch.
- Dropout:
  A dropout layer with a dropout rate of 0.5 is added after batch normalization. Dropout randomly sets half of the input units to 0 during training, which helps prevent overfitting.
- Fully connected layers:
  Two fully connected dense layers are added for classification. The first dense layer has 64 units with ReLU activation, and the final output layer has 1 unit with a sigmoid activation function for binary classification.
- Model compilation:
  The model is compiled using the Adam optimizer and binary cross-entropy loss function, which is typical for binary classification tasks. We efficiently trained the machine learning model while employing two crucial callbacks: early stopping to prevent overfitting and restore the best weights, and a learning rate schedule to optimize the convergence of the training process. These techniques are common practices in training neural networks to achieve better results and save training time. We find that the model stopped the training process when it reached the 2nd epoch; we started with lr = 0.001 and we found that the training process stopped automatically by the early stopping parameter before it changed the learning rate, as shown in Figure 13.

Running this LSTM model gives a detection accuracy, precision, recall, and F1-score of 99%, which is the best score we get among all the classification algorithms mentioned above, as shown in Figure 14.

https://doi.org/10.3390/engproc2026150101

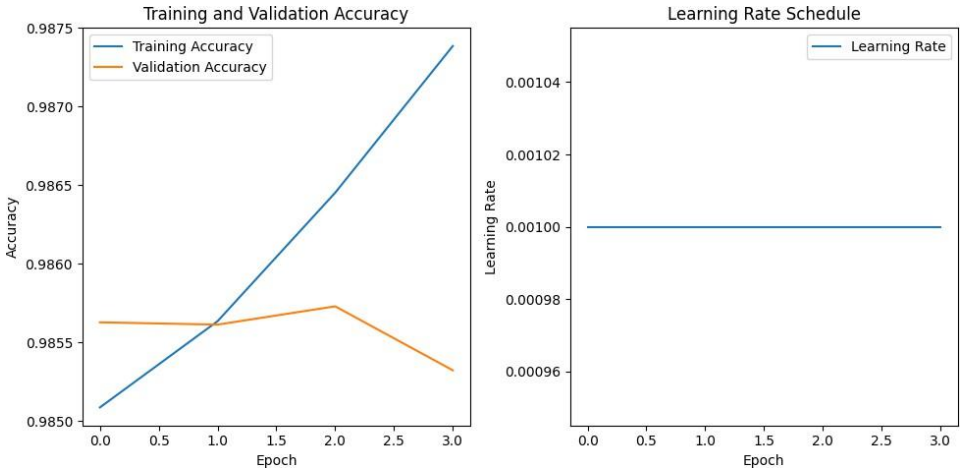
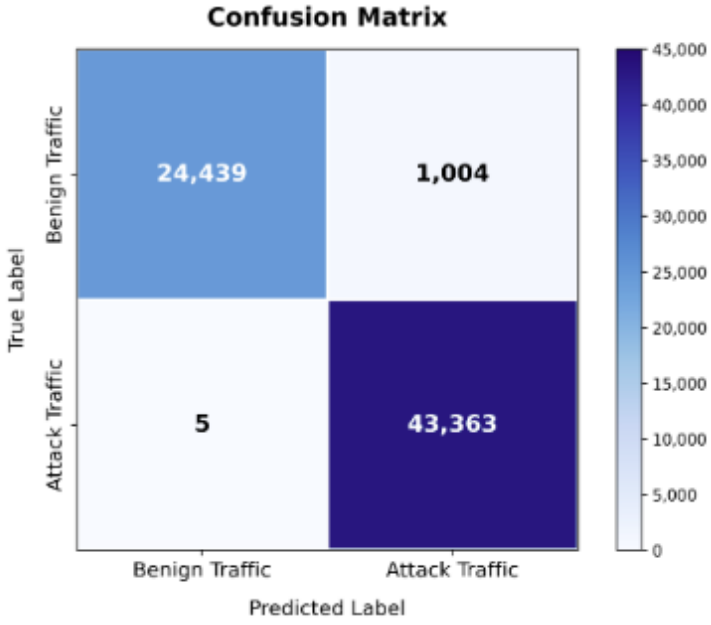


Figure 13. Training and validation accuracy learning rate for LSTM model.



Classification Report

	Precision	Recall	F1-Score	Support
Benign Traffic	1.00	0.96	0.98	25,443
Attack Traffic	0.98	1.00	0.99	43,368
Accuracy			0.99	68,811
Macro Avg	0.99	0.98	0.98	68,811
Weighted Avg	0.99	0.99	0.99	68,811

Figure 14. Confusion matrix and classification report for the LSTM model.

5. Validate the Model in Detecting Other Types of Attacks

The above ML and DL models were able to classify the webservers' traffic as benign or DDoS traffic, with an accuracy reaching 99%, but webservers actually are vulnerable to a wide range of attacks, according to the BreachLock 2022 Pentest Intelligence Report.

Many types of attacks still exist and affect webservers on a daily basis, like SQL injection, sensitive data exposure, function-level access control missing, arbitrary file upload, time-based blind SQL injection (Blind SQLi), authentication bypass, privilege escalation, remote code execution, information disclosure, and cross-site request forgery (CSRF), as shown in Figure 15.

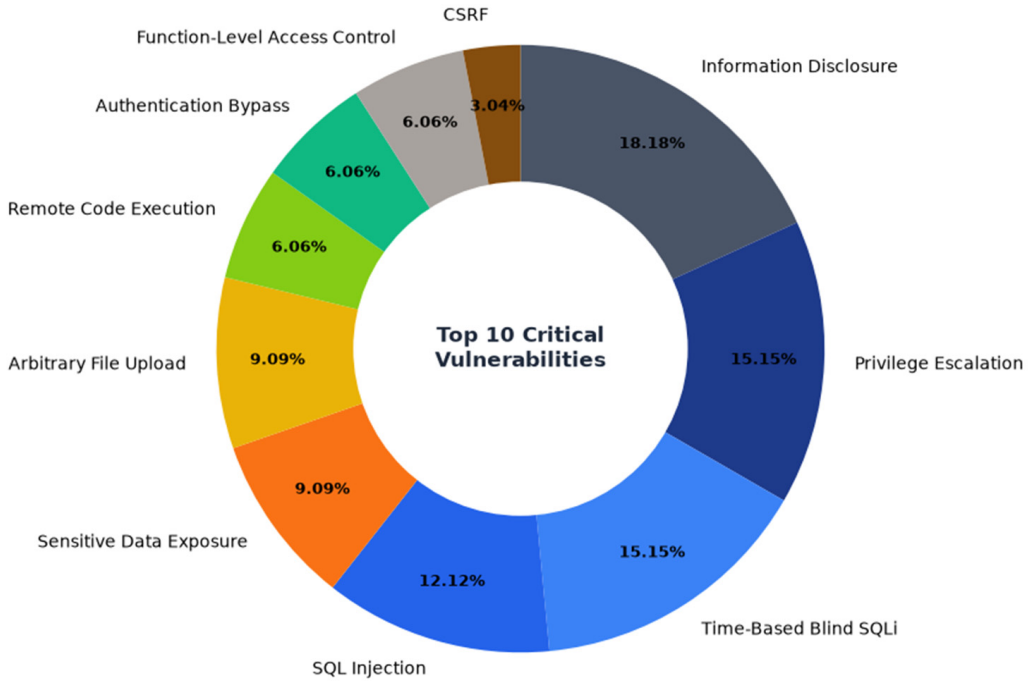


Figure 15. Top 10 critical vulnerabilities.

For this purpose, we created a special dataset by using an open-source web application security scanner, OWASP Zed Attack Proxy (OWASP ZAP), to generate different web application attack traffics like (1) SQL injection, (2) broken authentication, (3) sensitive data exposure, (4) broken access control (5) security misconfiguration, (6) cross-site scripting (XSS), (7) insecure deserialization, (8) components with known vulnerabilities and other attacks; these attacks were registered in the webserver access log and the dataset consists of 5632 attack records.

Then we applied the same data preprocessing steps that we applied on the training and testing portions of the dataset, as mentioned above. Later we loaded the saved models and revalidated them by detecting other types of attacks and the results are summarized in Table 2, showing the following accuracy scores among different algorithms: LSTM (92.89%) and it is the best accuracy score registered, SVM (76.91%), NB (75.58%), LR (68.81%), KNN (39.77%), DT (30.34%), ANN (28.17%), and RF (19.54%) and it is the worst accuracy score registered. So, to conclude, we found that both the ML and DL models registered an excellent percentage in detecting DDoS attacks (98%), but the ML models had poor performance in detecting other types of attacks, while LSTM was good in detecting other types of attacks (92.89%). This issue was expected because the models and the dataset used in training these models were designed to just detect DDoS attacks and no other types of attacks, but by using a complex model such as LSTM has proven to be a powerful tool for classification, especially since some of our dataset fields are text (referrer, request and user agent), with the ability to handle sequential data and capture long-term dependencies.

Table 2. Other types of attack detection accuracy.

	Algorithm	Other Types of Attack Detection Accuracy
1	Random Forest	19.54
2	Decision Tree	30.34
3	Naive Bayes	75.58
4	SVM	76.91
5	Logistic Regression	68.81
6	KNN	39.77
7	ANN	28.17
8	LSTM	92.89
Average		54.00
Max		92.89
Min		19.54

6. Conclusions

This paper suggests using the webserver access log as a dataset to train supervised ML classification models to classify websites' new visits as benign or DDoS traffic, and later the trained models tested in classifying the websites' visits registered in the webserver access log as benign or malicious traffic generated by other types of attack. As a result, we find that the recommended approaches effectively detect DDoS attack traffic with average detection accuracy, average precision, average recall, and average F1-score of 95.59%, 95.02%, 98.19%, and 96.55%, respectively, for ML algorithms and with detection accuracy, precision, recall, and F1-score of 98% for the ANN algorithm, and 98.53%, 99%, 99%, 99%, respectively, for the LSTM model. We find among the ML algorithms that the RF and KNN algorithm registered the best accuracy score of 98.4%, which is somewhat identical to the score we got from the ANN algorithm and the LSTM model. Furthermore, at the end, we tested the models to detect other types of attack, like (1) SQL injection, (2) sensitive data exposure, (3) security misconfiguration, and (4) cross-site scripting (XSS). Then we validated the pre-trained model using these attack records and we got the following accuracy scores among different algorithms: LSTM (92.89%) and it is the best accuracy score registered, SVM (76.91%), NB (75.58%), LR (68.81%), KNN (39.77%), DT (30.34%), ANN (28.17%), and RF (19.54%) and it is the worst accuracy score registered. So, to conclude, we found that both the ML and DL models registered an excellent percentage in detecting DDoS attacks (98%), but ML models have poorer performance in detecting other types of attack, while LSTM was good in detecting other types of attack (92.89%). This issue was expected because the models and the dataset used in training these models were designed to just detect DDoS attacks and no other types of attack. But using a complex model such as LSTM has proven to be a powerful tool for classification, especially since some of our dataset fields are text (referrer, request and user agent), with the ability to handle sequential data and capture long-term dependencies. However, at this point, the validation of our detection models is currently based on theoretical and modeling approaches. While these have provided promising results, there is a critical need for real-world testing to substantiate these findings. Future research should focus on practical experiments, possibly in collaboration with industry partners, to test these models against actual DDoS attacks. This will not only enhance the credibility of the research but also ensure that the models can effectively operate in diverse and dynamic real-world environments.

https://doi.org/10.3390/engproc2026150101

Author Contributions: Conceptualization, I.K. and G.M.; methodology, I.K.; software, I.K.; validation, I.K. and G.M.; formal analysis, I.K.; investigation, I.K.; resources, I.K.; data curation, I.K.; writing—original draft preparation, I.K.; writing—review and editing, I.K. and G.M.; visualization, I.K.; supervision, G.M.; project administration, G.M.; funding acquisition, none. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the findings of this study are not publicly available due to privacy and confidentiality restrictions imposed by the telecommunications company from which the data were obtained. Data may be available from the corresponding author upon reasonable request and with permission from the data owner.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Thomos, N.; Maugey, T.; Toni, L. Machine Learning for Multimedia Communications. *Sensors* **2022**, *22*, 819. [CrossRef] [PubMed]
2. Ali, O.; Ishak, M.K.; Bhatti, M.K.L.; Khan, I.; Kim, K. Il A Comprehensive Review of Internet of Things: Technology Stack, Middlewares, and Fog/Edge Computing Interface. *Sensors* **2022**, *22*, 995. [CrossRef] [PubMed]
3. Verma, D.; Srivastava, J.; Singh, R.; Tiwari, R. A Machine Learning Inspired Attendance Management System through Face Recognition. *IJARCCE* **2022**, *11*, 14–18.
4. Mokhtari, S.; Nikzad, S.; Mokhtari, S.; Sabour, S.; Hosseini, S. Investigating the reasons for students' attendance in and absenteeism from lecture classes and educational planning to improve the situation. *J. Educ. Health Promot.* **2021**, *10*, 221. [CrossRef] [PubMed]
5. Biju, J.; Sairam, S.; Kumar, K. Enhancing Attendance Management Systems Using Facial Recognition. *Comput. Sci. Eng.* **2024**, *13*. [CrossRef]
6. Raj, N.; Vadivel, R. Face recognition based attendance system using machine learning with location identification. *J. Adv. Res. Rev.* **2023**, *18*, 1029–1035. [CrossRef]
7. Jha, P.B.; Basnet, A.; Pokhrel, B.; Pokhrel, B.; Thakur, G.K.; Chhetri, S. An Automated Attendance System Using Facial Detection and Recognition Technology. *Apex J. Bus. Manag.* **2023**, *1*, 103–120. [CrossRef]
8. Kortli, Y.; Jridi, M.; Al Falou, A.; Atri, M. Face Recognition Systems: A Survey. *Sensors* **2020**, *20*, 342. [CrossRef] [PubMed]
9. Deepan, K.; Manoj Kumar, S.; Maheswari, M.; Balaji, A.S. Face Recognition System Using in Attendance for Educational Institutions. *IJARCCE* **2023**, *12*. [CrossRef]
10. Bhise, A.; Khichi, R.; Korde, A.; Lokare, P.D. Attendance System Using NFC Technology with Embedded Camera on Mobile Device. *IJARCCE* **2015**, *4*, 350–353. [CrossRef]
11. Sinha, P.; Harsh; Nithiya, S. Multitype student attendance management system. *AIP Conf. Proc.* **2024**, *3075*, 020048. [CrossRef]
12. Al-Muhaidhri, G.; Hussain, J. Smart Attendance System Using Face Recognition. *Int. J. Eng. Res.* **2019**, *8*, 10–17577. [CrossRef]
13. Patil, V.; Narayan, A.; Ausekar, V.; Dinesh, A. Automatic Students Attendance Marking System Using Image Processing and Machine Learning. In Proceedings of the 2020 International Conference on Smart Electronics and Communication (ICOSEC), New York, NY, USA, 10–12 September 2020.
14. Gunawan, T.S.; Shah, S.A.A.; Ishak, M.K.; Kartiwi, M.A. Deep learning approach to DDoS attack with imbalanced data at the application layer. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 785–793.
15. Lee, S.; Kim, K. Detection of application-layer DDoS attacks with PCA technique in web server environment. *J. Inf. Process. Syst.* **2012**, *8*, 605–616.
16. Xie, B.; Liu, J.; Zhang, Y. Application-layer DDoS attack detection using explicit duration recurrent network-based application-layer protocol communication models. *Int. J. Intell. Syst.* **2023**, *1*, 1–14. [CrossRef]
17. Bhayo, J.; Shah, S.A.; Hameed, S.; Ahmed, A.; Nasir, J.; Draheim, D. Towards a machine learning-based framework for DDoS attack detection in software-defined IoT (SD-IoT) networks. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106432. [CrossRef]
18. Salama, A.M.; Mohamed, M.A.; AbdElhalim, E. Enhancing network security in IoT applications through DDoS attack detection using ML. *Mansoura Eng. J.* **2024**, *49*, 10. [CrossRef]
19. Estupiñán Cuesta, E.P.; Martínez Quintero, J.C.; Avilés Palma, J.D. DDoS attacks detection in SDN through network traffic feature selection and machine learning models. *Telecom* **2025**, *6*, 69. [CrossRef]

20. Chen, L.; Wang, Z.; Huo, R.; Huang, T. An adversarial DBN-LSTM method for detecting and defending against DDoS attacks in SDN environments. *Algorithms* **2023**, *16*, 197. [CrossRef]
21. Osa, E.; Alhassan, A. Performance analysis of shallow and deep learning classifiers leveraging the CICIDS 2017 dataset. *Int. J. Inf. Secur. Appl. (IJISA)* **2025**, *17*, 40–53. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.