Journal of King Saud University - Computer and Information Sciences 36 (2024) 101938

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University - Computer and Information Sciences

journal homepage: www.sciencedirect.com

Full length article

DeepDefend: A comprehensive framework for DDoS attack detection and prevention in cloud computing

Mohamed Ouhssini [a,*], Karim Afdel [a], Elhafed Agherrabi [b], Mohamed Akouhar [c], Abdallah Abarda [d]

[a] Lab SIV, Department of Computer Science, University IBN Zohr, Agadir, Morocco
[b] LabIRF-SIC, Department of Mathematics, Faculty of Science, Ibn Zohr University, Agadir, Morocco
[c] Lab Partial Differential Equations, Algebra and Spectral Geometry, University IBN Tofail, Kenitra, 14000, Morocco
[d] Laboratory of Mathematical Modeling and Economic Calculations, Hassan First University Settat, Settat, Morocco

ARTICLE INFO

Keywords:
DDoS attacks
Cloud computing systems
Entropy forecasting
Deep learning models
CNN
Genetic algorithm

ABSTRACT

DeepDefend is an advanced framework for real-time detection and prevention of DDoS attacks in cloud environments. It employs deep learning techniques, notably CNN-LSTM-Transformer networks, to predict network traffic entropy and detect potential attacks. The framework uses a genetic algorithm for optimal feature selection, enhancing the efficacy of the AutoCNN-DT model in distinguishing between normal and attack traffic. Tested on the CIDDS-001 traffic dataset, DeepDefend demonstrates high accuracy in entropy forecasting and rapid, precise detection of DDoS attacks. This integrated approach combines time series analysis, genetic algorithms, and deep learning, offering a robust solution to protect cloud computing infrastructure against DDoS threats.

1. Introduction

Cloud computing (CC) brings to the table a formidable set of tools and techniques that provide services via the internet, harnessing software and hardware systems housed in data centers (Armbrust et al., 2010; Knorr and Gruman, 2008). These systems enable the on-demand provision of computing power, storage, applications, and services, offering numerous benefits including scalability, accessibility, cost reductions, and improved business adaptability (Mell and Grance, 2011). The deployment of cloud services facilitates a rapid expansion of IT infrastructures, with capabilities that can be quickly provisioned and relinquished with minimal administrative overhead (Zhang et al., 2010). Fundamental delivery models for CC services include Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) (Zhang et al., 2010; Simmon, 2018), while access in a CC environment can be managed via four deployment options: private, public, community, and hybrid clouds (Mather et al., 2011).

Yet, despite these considerable benefits, the specter of security remains a significant hurdle to the broad adoption of cloud computing, with Distributed Denial of Service (DDoS) attacks posing serious threats to system availability (Ali et al., 2015; Yan et al., 2015). Such attacks aim to obstruct legitimate users from accessing computer or network services (Yan and Yu, 2015), potentially causing catastrophic consequences for businesses and organizations relying on cloud computing. DDoS attacks typically involve the use of large numbers of compromised computers or "bots" to launch coordinated onslaughts against a target (Osanaiye et al., 2016). These attacks start with the perpetrator infecting devices with malware to amass a botnet, which is then used to inundate a specific server with a deluge of traffic, thus overwhelming its capacity (Gupta and Badve, 2016).

The scale of such attacks is continually escalating. For instance, Amazon had to contend with a 2.3 Tbps DDoS attack in the first quarter of 2020, representing the most sizeable attack of its kind ever recorded (Cook, 2023). According to the data presented in Fig. 1 (Nicholson, 2022), this attack was approximately four times larger than the previous record. DDoS attacks can have a profound impact, potentially accounting for up to 25% of a country's total internet traffic

* Corresponding author.
E-mail address: mohamed.ouhssini@gmail.com (M. Ouhssini).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2024.101938
Received 9 August 2023; Received in revised form 13 January 2024; Accepted 19 January 2024
Available online 4 February 2024

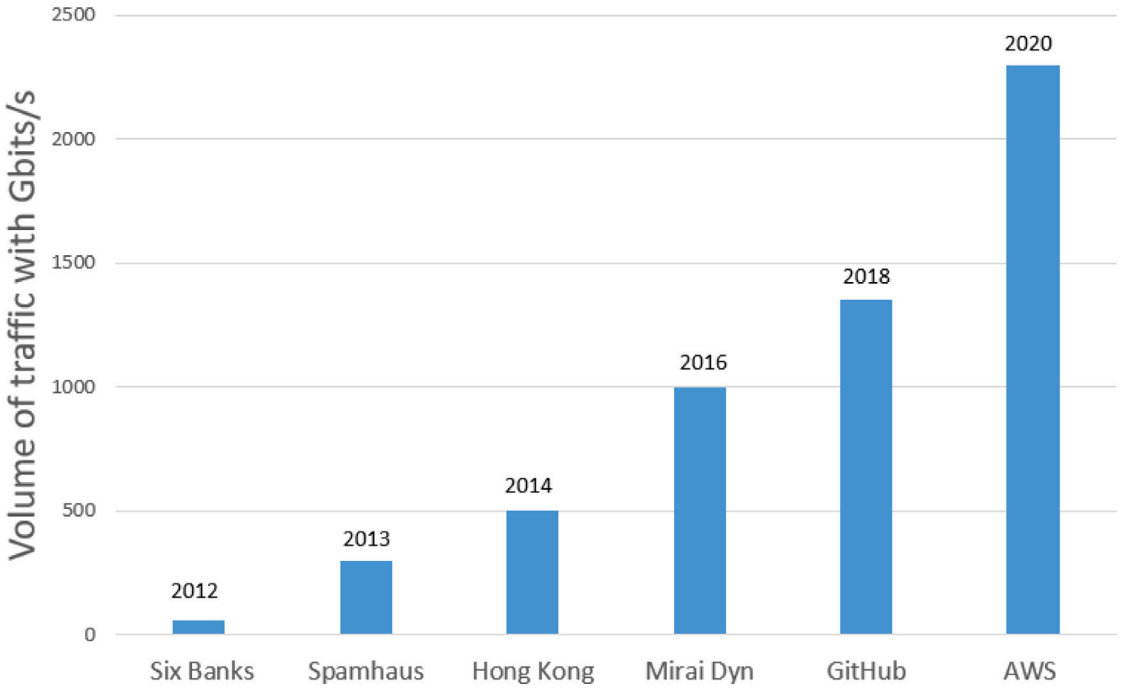

Fig. 1. Increasing volume of DDoS attacks.

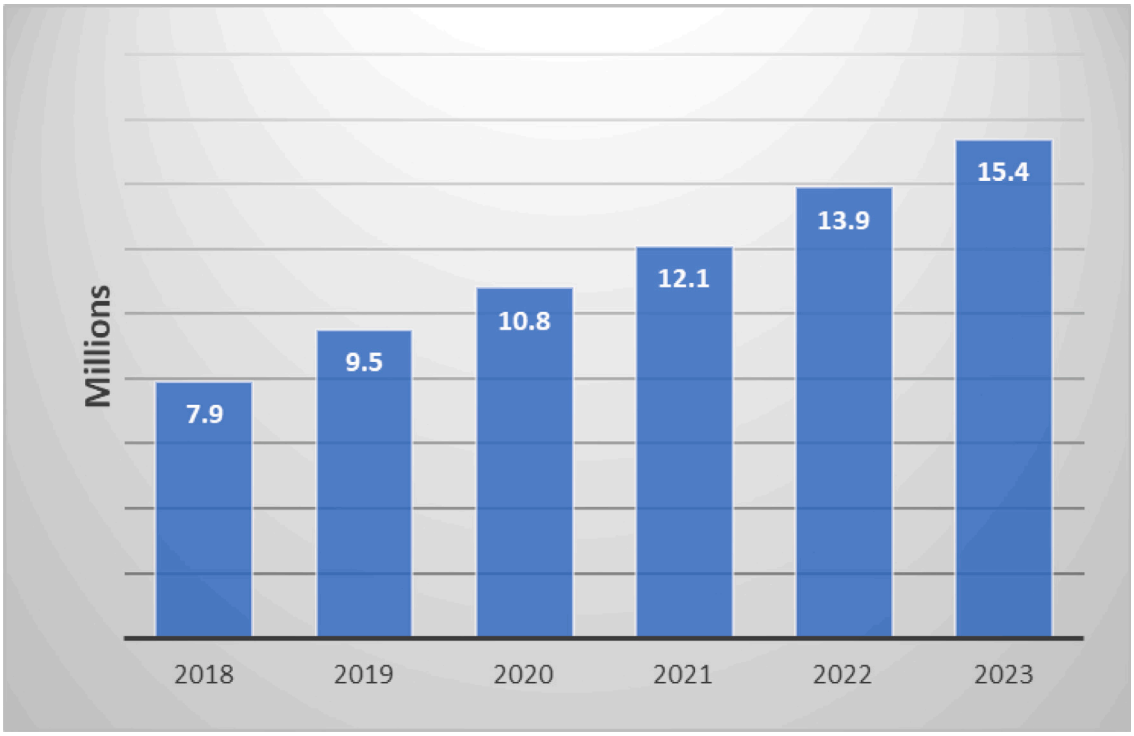

Fig. 2. Number of DDoS attacks over time.

during the attack (Cybersecurity Ventures, 2020). As per Fig. 2, the number of DDoS attacks is expected to more than double, reaching 15.4 million by 2023 (Cisco, 2020).

The significant risk posed by DDoS attacks to service providers makes it crucial to tackle this issue to assure the reliability of cloud computing. In response to the escalating threat of DDoS attacks in cloud settings, both researchers and practitioners have devised a multitude of strategies for their mitigation, detection, and prevention. The use of Anomaly-based Intrusion Detection Systems (IDS) has been prevalent, helping to discern unusual network activities that may signify a DDoS attack (Hezavehi and Rahmani, 2020). The detection and prevention DDoS attacks remain a paramount concern. The most effective systems in this field are those that analyze data traffic in real-time, allowing for the early detection and mitigation of such attacks. However, the effectiveness of these systems is often hampered by their resource-intensive nature. As data volumes increase, these real-time systems become burdened by the need for extensive computational resources, leading to diminished efficiency and potential vulnerabilities.

This situation underscores the urgent need for the development of a more resource-efficient real-time system for DDoS attack detection and prevention. The ideal solution would be a system that maintains the ability to analyze data traffic in real-time but does so using fewer resources. Such a system would not only be more sustainable in terms of operational costs but also more agile and effective in rapidly changing cyber environments. The development of this kind of system represents a significant challenge but is crucial for advancing cybersecurity measures in an era of ever-growing digital threats. It would constitute a pivotal advancement in maintaining robust digital defenses, ensuring better protection for vital online infrastructures against the increasing frequency and sophistication of DDoS attacks.

The DeepDefend framework is a groundbreaking solution in DDoS attack detection and prevention. Utilizing forecasting entropy, it responds effectively to attacks in their short duration while minimizing resource usage. Unlike other systems that require continuous computation, DeepDefend strategically uses its computational power for less than 1% of the time, focusing on periods when attacks occur. This efficient method, effective in just 0.31% of our monitoring duration, significantly enhances cybersecurity threat management by improving efficiency and reducing resource strain.

This study introduces DeepDefend, a comprehensive framework for real-time detection and prevention of DDoS attacks. The framework consists of seven key steps aimed at detecting and mitigating such attacks. These steps include:

1. **Traffic Collection**: Gathering relevant data pertaining to network activity and system performance.
2. **Data Aggregation**: The collected data is organized using a time window algorithm.
3. **Entropy Forecasting**: Historical data is used to predict the entropy of header features.
4. **Attack Predictor**: Time windows are classified as normal or anomalous based on forecasted entropy.
5. **Features Preprocessing**: Network traffic data is preprocessed to extract relevant features and identify attack patterns.
6. **Classification**: Detected DDoS attacks are categorized based on their characteristics.
7. **The Checker**: The system continuously measures entropy and responds to significant deviations.

In the forecasting stage, statistical methods are compared with deep learning networks (DNNs) that are optimized using Bayesian optimization. This approach helps in selecting the best hyperparameters for each DNN architecture. The experimental results demonstrate that DNNs outperform other methods in terms of performance.

To determine the entropy threshold, a decision tree model is utilized to assess the probability of an attack.

In the preprocessing stage, data cleaning, normalization, and feature selection techniques are applied. Subsequently, the classification stage employs DNNs to classify the data and detect DDoS attacks.

The DeepDefend framework makes substantial contributions to the field of DDoS attack detection and mitigation. It stands out by:

1. Leveraging Entropy Effectively:
    - Recognizing the pivotal role of entropy in DDoS prevention.
    - Utilizing entropy calculations to identify and distinguish anomalous traffic patterns from normal behavior.
    - Using entropy forecasting to predict windows that contain DDoS attacks and process them, reducing the heavy preprocessing and classification burden on the system compared to conventional systems.
2. Integrating Advanced Time Series Analysis Techniques:
    - Introducing a blend of statistical methods and sophisticated deep learning architectures.
    - Employing various techniques, including ARIMA, RNNs, LSTM, CNNs, autoencoders, and transformers, for accurate entropy forecasting based on historical data patterns.
3. Addressing Dynamic Threshold Challenges:
    - Tackling the challenge of dynamically determining entropy thresholds.
    - Proposing a strategy for continuous monitoring, analysis, and adaptation to changing network conditions.
4. Ensuring Efficiency in Resource Allocation:
    - Demonstrating efficiency gains through proactive entropy forecasting.
    - Optimizing resource allocation, concentrating efforts precisely when and where needed, and minimizing unnecessary resource expenditure.
5. Enabling Proactive Detection and Mitigation:
    - Highlighting the proactive nature of the system in identifying and mitigating suspicious traffic flows.
    - Anticipating anomalous behavior through entropy forecasting and initiating real-time security measures, thereby enhancing overall resilience against potential DDoS attacks.

Using entropy forecasting to predict windows that contain DDoS attacks and process them, DeepDefend significantly reduces the heavy preprocessing and classification burden on the system compared to conventional systems. This approach offers the following advantages:

- Efficient Resource Utilization:
    - By predicting windows that are likely to contain DDoS attacks, DeepDefend focuses its resources on analyzing and mitigating the most critical periods.
    - This targeted approach reduces the computational load associated with preprocessing and classifying all incoming data, optimizing resource allocation and improving overall system efficiency.
- Real-Time Response:
    - By accurately forecasting the occurrence of DDoS attacks, DeepDefend enables real-time response and mitigation.
    - It eliminates the need for post-attack analysis and minimizes the time required to detect and counteract attacks, enhancing the system's ability to protect against evolving threats.
- Scalability and Performance:
    - DeepDefend's entropy forecasting approach allows the system to scale more effectively as network traffic increases.
    - By reducing the computational overhead of preprocessing and classification, the system can handle larger volumes of data without sacrificing performance or responsiveness.
- Improved Accuracy:
    - Entropy forecasting enables DeepDefend to differentiate between normal traffic patterns and anomalous behavior associated with DDoS attacks more accurately.
    - By focusing on windows likely to contain attacks, DeepDefend can apply more sophisticated analysis techniques tailored to detecting and mitigating DDoS attacks, resulting in improved detection accuracy.

- Reduced False Positives:
  - DeepDefend's entropy forecasting approach helps minimize false positive detections by filtering out normal windows.
  - By identifying and processing windows with abnormal entropy levels, the system can reduce the number of false alarms, ensuring that security resources are allocated effectively.

Overall, our system marks a significant advancement in early DDoS attack detection, employing forecasting entropy to efficiently respond to attacks within their typically short duration. Unlike other online systems burdened with continuous heavy calculations, our system strategically allocates full computational power for less than 1% of the time, focusing on the brief periods when attacks occur. This method, observed to be effective in just 0.31% of our monitoring duration, not only improves efficiency but also minimizes resource strain, offering a more effective solution for DDoS attacks detection and prevention.

The remainder of the paper is structured as follows: Section 2 provides an overview of related works, while Section 3 focuses on the utilized dataset. In Section 4, the proposed approach is detailed, followed by experimentation and validation information in Section 5. Section 6 presents results and discussions. The paper concludes in Section 7, summarizing key findings and suggesting future research directions.

2. Related works

We have organized this state-of-the-art report into three parts. The first part focuses on recent advances in DDoS Attack Detection and Mitigation in Cloud Environments. The second part focuses on work based on the use of entropy as a powerful mechanism for detecting and mitigating attacks by analyzing the randomness and patterns within network traffic. The third part deals with the prediction of DDOS attacks using time series analysis.

2.1. Comparative analysis of modern techniques in detecting and mitigating DDoS attacks in cloud environments

In conclusion, the critical studies discussed in this context have highlighted a range of methods and principles employed for detecting DDoS attacks in cloud environments. Each approach brings its own set of advantages and limitations. The utilization of anomaly-based frameworks, deep learning techniques, optimized classifiers, and innovative strategies has demonstrated promising outcomes in the identification and mitigation of DDoS attacks. However, challenges such as the dependability of third-party auditors, security concerns, threshold determination, prevention and mitigation strategies, and validation in real-world environments still require attention (see Table 1).

2.2. Comparative analysis of related work using entropy to detect and mitigate DDoS attacks in cloud

In conclusion, we can observe that each study employs entropy-based techniques to detect and mitigate DDoS attacks in the cloud. The advantages of these approaches include promising results, effective differentiation of attacks, and outperforming previous methods. However, limitations include the need for real-world evaluation, challenges in threshold selection, reliance on synthetic or benchmark datasets, and limited discussion on practical deployment and scalability (see Table 2).

2.3. Comparative analysis of related work using time series analysis to detect and mitigate DDoS attacks

In this detailed analysis, we can observe that each study utilizes time series analysis as a core method for detecting and mitigating DDoS attacks. The advantages of these approaches include promising results, effective detection, and the ability to distinguish between normal and attack traffic. However, the limitations include the need for real-world evaluation, challenges in threshold definition, and the focus on detection rather than prevention or mitigation. Further research is required to validate the performance of these methods in practical environments, evaluate their effectiveness under different attack scenarios, and explore their integration with mitigation and prevention strategies. By focusing on novel approaches such as time series analysis and deep learning techniques, we hope our results can contribute to developing more effective measures against future cyber threats posed by DDoS attackers (see Table 3).

3. The CICIDS-001 dataset: Exploring features and utilization in the study

The dataset used in this study consists of various features that provide information about the network traffic, including source and destination IP addresses, port numbers, transport protocols, timestamps, duration, data volume, TCP flags, class labels, attack types, and unique identifiers for attacks. This comprehensive dataset allows analyzing and detecting different types of attacks in a detailed manner, providing valuable insights for developing effective defense mechanisms against cyber threats.

Table 4 presents the initial arrangement of a dataset designed for network traffic analysis. It encompasses various attributes such as source/destination IPs, ports, protocols, timing, data volume, TCP flags, security classifications, as well as attack types and descriptions. These attributes are provided by the creators without undergoing any preprocessing, which could potentially modify them for the purpose of analysis. Selecting an appropriate dataset for research purposes can prove to be a formidable task. In the existing body of literature, two datasets, namely CIDDS-001 (Ring et al., 2017) and CIC-IDS-2018 (Sharafaldin et al., 2018), which were generated in a cloud environment, have gained significant consideration. One notable advantage shared by both datasets is the inclusion of timestamps as a feature, rendering them conducive for time series analysis. However, it is imperative to acknowledge certain challenges associated with CIDDS-001. Firstly, its limited number of features poses constraints on conducting in-depth analyses, particularly when employing machine learning and deep learning techniques. Secondly, the dataset exhibits a notable class imbalance, particularly concerning attacks other than Distributed Denial of Service (DDoS), which may hinder the accuracy and robustness of predictive models. Lastly, CIDDS-001 contains a substantial amount of duplicated data, which could potentially introduce biases and impede the fidelity of the results. In contrast, CIC-IDS-2018 dataset, although also generated in a cloud environment, presents an aspect that warrants careful consideration. Specifically, the deliberate control of the date and time of launching DDoS attacks within this dataset diminishes its suitability for time series analysis, which relies heavily on the temporal nature of data. Despite the aforementioned limitations associated with CIDDS-001, the decision to utilize this particular dataset was made. Despite its imperfections, CIDDS-001 remains the superior choice due to its capacity to more accurately simulate real-world behaviors exhibited by attackers. The added fidelity to actual attack scenarios outweighs the dataset's shortcomings, and it is deemed to offer valuable insights for the intended research.

Table 1

Ref.	Method	Advantages	Limitations
Hezavehi and Rahmani (2020)	Anomaly-based DDoS detection framework using a third-party auditor (TPA)	High accuracy, low false positive/negative rates	Dependability of TPA, potential security issues, TPA itself can be a target of DDoS attacks
Virupakshar et al. (2020)	DDoS detection in private clouds using OpenStack, machine learning techniques, and a deep learning network	K-nearest neighbors and decision trees delivered the greatest F1-score scores	Focuses only on detection rather than prevention or mitigation
Bhardwaj et al. (2020)	DDoS detection using a Sparse Autoencoder (SAE) optimized by hyperparameter tuning	Decreased construction error, reduced gradient issues, avoidance of overfitting, superior performance on NSL-KDD dataset	Validating the approach in an actual cloud environment is still needed, only focuses on detection
Velliangiri et al. (2020)	DDoS detection using deep learning techniques and Elephant Herd Optimization	More effective in identifying DDoS attacks compared to other methods	Does not discuss how to mitigate attacks after detection
Kushwah and Ranga (2020)	DDoS detection using a voting-based ensemble extreme learning machine	Achieved the highest accuracy, sensitivity, and specificity	Did not compare their approach with other related works
Shidaganti et al. (2020)	DDoS detection and prevention in cloud environments using the Selective Cloud Egress Filter (SCEF)	Promising outcomes, can take specified corrective measures when an attack is discovered	Difficult to choose an appropriate threshold due to the statistical nature of the processes
Bhushan and Gupta (2018)	DDoS mitigation system using the capabilities of Software Defined Networking (SDN)	Successful strategy in reducing DDoS attacks and giving the system vital extra time to defend against them	Further research needed to integrate with other detection methods and assess its performance in real-world cloud environments
Cheng et al. (2018)	DDoS detection using an optimized random forest classifier based on genetic algorithm and flow correlation degree	Good accuracy with low false- and positive-rates	Should test the method on more recent datasets and compare the results to other similar research
Liu et al. (2023)	LDDoS attacks detection using asynchronous FL-bidirectional LSTM model	High accuracy (98.80%), high precision (99.34%), high recall (98.25%), and reduced time complexity	Further testing on recent datasets is recommended
Bamasag et al. (2022)	Real-time DDoS monitoring and detection using ML techniques (Naive Bayes, K-nearest neighbors, decision tree, random forest)	Achieved 99.38% accuracy for the random forest	Method should evaluate using most recent datasets, does not discuss required computational resources and potential for false positives
Sanjalawe and Althobaiti (2023)	DDoS detection using a hybrid CNN-LSTM model and ensemble feature selection	High accuracy, superior performance compared to existing systems	Tested only on a single dataset; future studies should test on more recent datasets and real-world data
Alduailij et al. (2022)	DDoS detection using MI and RFFI methods for feature selection and different classifiers	Reduced misclassification errors, outperforms existing methods in terms of accuracy	Does not discuss the required computational resources
Balasubramaniam et al. (2023)	DDoS detection using a Gradient Hybrid Leader Optimization (GHLBO) algorithm	High TPR, TNR values and testing accuracy	Method evaluated only on NSL-KDD dataset and for a specific set of parameters
Priyadarshini and Barik (2022)	DDoS mitigation in fog and cloud environments using a deep learning-based detection mechanism	Effective in detecting and mitigating DDoS attacks, outperforms existing methods in terms of accuracy and false positive rate	The proposed method may require significant computational resources to implement

4. The proposed approach

The methodology proposed in this paper, known as the DeepDefend framework, presents a detailed strategy for detecting and preventing DDoS attacks in cloud environments. The framework comprises seven key components: Traffic Collection, Aggregation, Entropy Forecasting, Attack Prediction, Checker, Feature Preprocessing, and Classification.

The process begins with the Traffic Collection component, which gathers data from the front-end devices of the network. This collected data is then subjected to Aggregation, utilizing a specific time window algorithm.

Next, the Entropy Forecasting component utilizes historical data to predict the entropy of header features. The Attack Predictor component categorizes these windows into two distinct categories: ‘‘normal’’ and ‘‘anomalous’’. When an anomalous window is detected, the Feature Preprocessing components are activated immediately to identify and block potential DDoS attacks using flow classification techniques.

To ensure the accuracy of the system, the Checker component validates operations when data windows are labeled as normal. It calculates entropy per minute and feeds this information to the Attack Predictor, which corrects any forecasting errors and improves the prediction accuracy for potential anomalies or attacks.

This cycle continues to operate, ensuring the effective detection and prevention of DDoS attacks. Figs. 3, 4 and 5 provide a comprehensive overview of the proposed framework.

4.1. The framework development

Step 1: Traffic Collection The initial phase involves the system initiating its data acquisition process by gathering relevant information from the network’s front-end devices. This collected data, encompassing various parameters related to network traffic, serves as the foundation for subsequent stages of analysis and detection.

Step 2: Data Aggregation In this stage, the system organizes the gathered data using a time window algorithm. This method allows for

Table 2

Ref.	Method	Advantages	Limitations
Ahalawat et al. (2019)	Entropy-based DDoS detection with rate limiting	Promising results in detecting DDoS attacks in simulations. Utilizes entropy calculations of key flow characteristics. Combines entropy-based approach with rate limiting.	Real-world evaluation necessary to assess performance in practical environments. Challenges in accurately selecting the appropriate threshold for entropy calculation.
Daneshgadeh et al. (2019)	Hybrid system with Kernel Online Anomaly Detection (KOAD), Shannon Entropy, and Mahalanobis Distance	Effective differentiation between DDoS attacks and Flash Events (FEs). Utilizes Mahalanobis Distance to distinguish various types of DDoS attacks. Low detection time and minimal resource requirements.	Evaluation using real-world data would be beneficial for validation. Reliance on synthetic DDoS attacks and limited real-world dataset.
Koay et al. (2018)	Multiple entropy-based features and ensemble classifier	Successful identification of various types of DDoS attacks. Generation of entropy-based features and entropy variation features. Outperforms several previous approaches in terms of detection.	Validation using recent real-world network traffic needed for comprehensive evaluation. Lack of comparison with other machine learning techniques beyond benchmark datasets.
Kesavamoorthy and Ruba Soundar (2018)	Multi-agent system with Particle Swarm Optimization (PSO) and entropy-based techniques	Accurate detection of DDoS attacks using multi-agent system. Utilizes PSO and entropy-covariance techniques for distinguishing network traffic. Promising results obtained from simulation experiments.	Performance evaluation under real-world cloud environments required for practical insights. Limited discussion on the robustness against various forms of DDoS attacks.
Idhammad et al. (2018)	Entropy estimation with information gain ratio and machine learning techniques	Enhanced accuracy with the merger of supervised and unsupervised learning. Effective clustering of anomalous clusters using Co-clustering algorithm. Outperforms many related works in terms of performance.	Evaluation limited to benchmark datasets, real-world evaluation would be valuable. Lack of discussion on the scalability and efficiency of the proposed method.
Aladaileh et al. (2023)	Evolutionary techniques with pre-processing, detection, and mitigation layers for DDoS attack detection	Improved accuracy with low false positive and false negative rates, exceptional response time, effectiveness over a wide array of attack types.	Simulation-based evaluation, scalability research needed.
Kareem and Jasim (2022)	Entropy-based technique for detecting low- and high-rate DDoS attacks in SDN systems	More accurate than other algorithms tested, potential option for detecting DDoS attacks.	Requires more validation against various forms of DDoS attacks.
Shetty (2022)	Entropy-based technique for identifying UDP flood attacks in SDN systems using OpenFlow protocol and POX controller	Outperforms other approaches in terms of detection rate and false positive rate for UDP flood attacks.	Requires validation against various forms of DDoS attacks, scalability research needed.

Table 3

Ref.	Method	Advantages	Limitations
Fadaei Fouladi et al. (2018)	Time series analysis and statistical measures	- Promising results in detecting DDoS attacks. - Utilizes statistical measures to uncover patterns in time series data. - Ability to distinguish between normal traffic and DDoS traffic.	- Real-world evaluation required to assess performance in practical environments. - Challenge in accurately defining the threshold for entropy calculation.
Ni et al. (2013)	Time series analysis with Adaptive Autoregressive (AAR) model and SVM	- Effective in detecting DDoS attacks using time series analysis. - Utilizes an AAR model to create time series data based on entropy calculations. - SVM classifier for DDoS attack classification.	- Real-world evaluation needed to validate the performance in live environments. - Limited to detecting DDoS attacks and not focused on mitigation or prevention.
Tabatabaie Nezhad et al. (2016)	Time series modeling with ARIMA and chaotic systems	- Accurate detection of DDoS attacks through time series modeling. - ARIMA model used for predicting packet count. - Chaotic behavior analysis using the maximum Lyapunov exponent.	- Further evaluation under different circumstances (e.g., low-rate DDoS attacks, flash crowds) required for a comprehensive assessment. - Relies on simulated traffic data, real-world evaluation needed.
Fouladi et al. (2020)	Time series analysis-based DDoS detection for SDN	- High accuracy in detecting DDoS attacks using time series analysis. - Utilizes ARIMA model and Lyapunov Exponent for anomaly detection. - Dynamic threshold employed for detecting unique source and destination IP addresses.	- Evaluation on more recent datasets and comparison with other related works is necessary. - Limited to detection and lacks focus on mitigation and prevention strategies.

Table 4
Features description of the dataset.

Feature	Description
Src IP	Source IP address of the traffic origin.
Src Port	Source port number indicating the specific application or service on the source device.
Dest IP	Destination IP address representing the destination of the network traffic.
Dest Port	Destination port number specifying the application or service on the destination device.
Proto	Transport protocol used for transmitting data packets (e.g., ICMP, TCP, UDP).
Date first seen	Timestamp indicating when the flow was first observed by the detection system.
Duration	Duration of a specific flow, representing how long it lasted.
Bytes	Numeric value indicating the number of bytes transmitted during each flow.
Packets	Numeric value indicating the number of packets transmitted during each flow.
Flags	Concatenation of all TCP flags associated with the flow.
Class	Class label assigned to each record, categorizing them as normal, attacker, victim, suspicious, or unknown.
AttackType	Type of attack that occurred within each record (e.g., portScan, DoS attacks).
AttackID	Unique identifier grouping flows belonging to individual attacks.
Description	Additional information about the attack, such as attempted password guesses for SSH brute force attacks.

a comprehensive examination of network traffic trends within distinct time frames. By fostering a complete understanding of traffic activity, the system becomes more adept at identifying anomalies and potential indicators of DDoS attacks.

Step 3: Entropy Forecasting Utilizing historical data, the system employs entropy forecasting techniques to predict the entropy of header features. Entropy forecasting provides valuable insights into expected network traffic patterns, enabling the system to identify deviations that may signify the presence of DDoS attacks.

Step 4: Attack Predictor Based on the forecasted entropy, the attack predictor component classifies time windows as either normal or anomalous. This critical step allows for the differentiation between legitimate network traffic and potentially malicious DDoS attack traffic, facilitating prompt detection and intervention.

Step 5: Features Preprocessing The collected network traffic data undergoes preprocessing to extract relevant features and identify potential DDoS attack patterns based on a genetic algorithm. This preprocessing stage enhances the system's ability to detect and differentiate DDoS attacks accurately, thereby improving overall security measures.

Step 6: Classification The system employs classification techniques to categorize the detected DDoS attacks based on their specific characteristics. This classification step enables a better understanding of attack patterns, facilitating the implementation of targeted prevention strategies and aiding in future analysis.

Step 7: The Checker To ensure maximum accuracy, the system consistently measures the entropy of the currently incoming data flows. In cases where the detected entropy significantly deviates from the predicted values, the system promptly shifts its focus to the attack predictor. This agile approach in the system facilitates swift responses to changing DDoS attack trends.

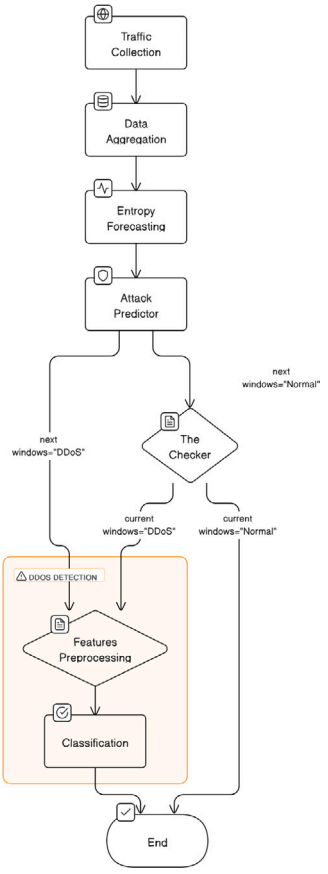

Fig. 3. The overview of the proposed framework.

4.1.1. Data collection and data aggregation

In this phase, the system extracts crucial data from network devices, including routers, switches, and firewalls. This data, referred to as network flow data, contains vital information such as source and destination IP addresses, port numbers, and packet and byte counts. To conduct thorough time series analysis, the system meticulously aggregates this network flow data into specific time windows, each lasting for 1 min. This aggregation process ensures the comprehensive analysis of traffic patterns over an extended period. By structuring the network flow data into these time windows, the system obtains a detailed perspective of network activity and behavior within each window. This approach allows for the prompt detection of anomalies or deviations within the defined time frame, potentially signaling the presence of DDoS attacks. The systematic analysis of traffic patterns, facilitated by the aggregation of network flow data into time windows, enhances the system's capability to identify subtle changes or abnormal traffic behavior associated with DDoS attacks.

Moreover, the utilization of specific time windows enables the accumulation of data over extended periods, providing a holistic view of network activity. This comprehensive perspective aids in the identification of patterns or trends that may indicate DDoS attacks. By monitoring traffic patterns within these aggregated time windows, the system effectively captures and analyzes overall network behavior, increasing its ability to detect deviations and abnormalities associated with DDoS attacks.

4.1.2. Entropy calculation and forecasting

Effective DDoS attack prevention methods rely on the utilization of entropy, which captures the randomness and unpredictability of network traffic patterns. Entropy serves as a measure of information content or disorder within network data, enabling various advantages in preventing DDoS attacks. One key advantage is anomaly detection, where entropy calculation enables the identification of abnormal or anomalous traffic patterns. By comparing the calculated entropy with

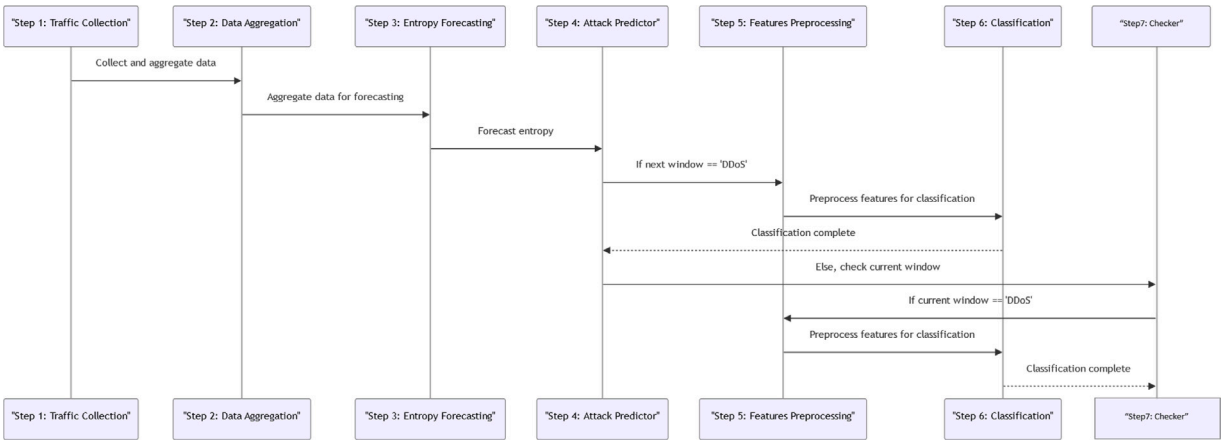

Fig. 4. The sequence diagram of the proposed framework.

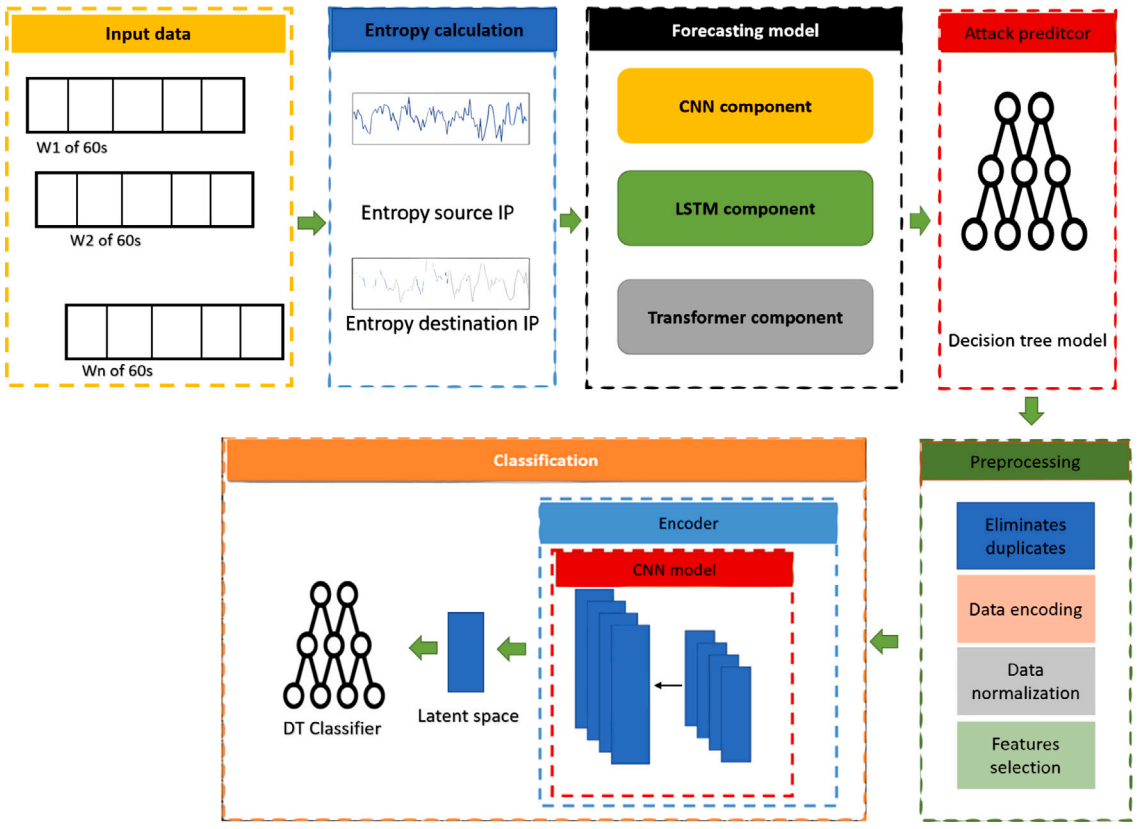

Fig. 5. DeepDefend framework - Real-Time DDoS detection and prevention using CNN-LSTM-Transformer and AutoCNN-DT: The CNN-LSTM-Transformer model forecast entropy over 60-s time intervals and subsequently sends these data to an attack predictor. The predictor evaluates whether the next time window exhibits indications of a potential DDoS attack. For this purpose, the AutoCNN-DT model intervenes to verify the actual presence of the attack. Prior to this, a preprocessing phase occurs, emphasizing the optimal selection of features through a genetic algorithm. These optimal features are then employed by AutoCNN-DT to determine whether the current time window corresponds to an attack or normal behavior.

expected or normal levels, deviations can be detected, indicating the presence of DDoS attacks or suspicious activities. Another advantage is pattern recognition. By analyzing the entropy of specific features such as IP addresses, patterns associated with DDoS attacks can be identified. This facilitates the development of accurate detection algorithms that differentiate between normal and malicious traffic.

Scalability is also crucial in DDoS prevention. Entropy-based methods effectively handle large volumes of network traffic, as DDoS attacks aim to overwhelm systems with excessive traffic. Entropy-based techniques provide a scalable approach for evaluating the information content and randomness of network flows, enabling efficient detection and mitigation of attacks.

Real-time analysis is another significant benefit. Entropy calculations can be performed in real-time, allowing immediate detection and response to DDoS attacks. Continuous monitoring and analysis of network traffic entropy enable timely identification of suspicious patterns and anomalies, enabling proactive measures to mitigate attacks and ensure network resource availability.

The proposed system incorporates entropy forecasting as a crucial step. The future entropy values are accurately predicted based on historical data patterns. Forecasting entropy for the upcoming minute

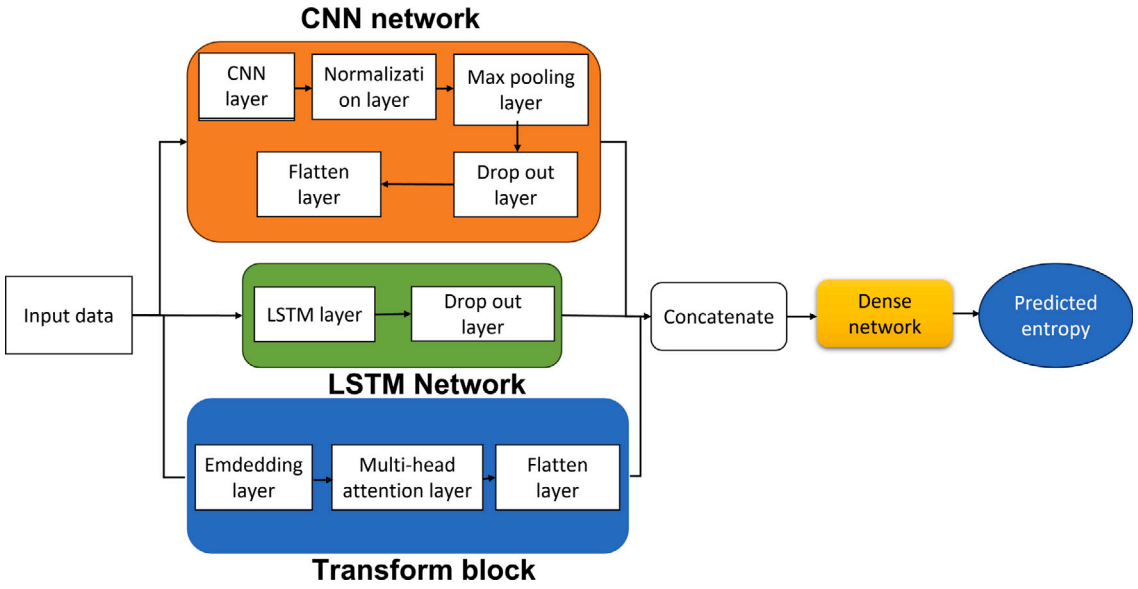

Fig. 6. Architecture adopted for predicting the entropy based on source and destination IP addresses over time within a 60-s timeframe.

helps anticipate suspicious characteristics that deviate from normal behavior. If the forecasted entropy exceeds the normal range, potential anomalous traffic flows are identified, triggering real-time security measures for analysis and mitigation.

To enable entropy forecasting, the system applies Algorithm 1, generating two lists: E_s (entropy values for source IP address windows) and E_d (entropy values for destination IP address windows). These lists serve as crucial inputs for constructing the entropy forecasting model, enhancing the system's ability to proactively identify suspicious traffic flows and effectively analyze and mitigate potential threats posed by anomalous network behavior.

Algorithm 1 Entropy calculation

Input: Network data
Output: lists of entropies: E_s, E_d
Begin
for each t **do**
    $E_s, E_d \leftarrow \text{entropy}(w_t)$
end for
End

Here, w_t represents the data of the window, which varies as 't' changes.

To develop the entropy forecaster, we first preprocess the data for time series analysis using a 60-s time window. Then, we compare different techniques of forecasting, including statistical models based on ARIMA, as well as deep learning techniques such as CNN, LSTM, CNN-LSTM, LSTM-Encoder and Transformer. Through this comparison, we aim to identify the most effective technique for accurately forecasting entropy in order to enhance the security of cloud computing systems. The transformer-based model produced the greatest results in terms of MSE, MAE and MAPE. These findings have been thoroughly examined and are reported in the research's results and discussion section.

We investigated the efficacy of different models in enhancing the security of cloud computing systems. Fig. 6 and Algorithm 2 illustrate the top-performing model, consisting of three key components: a Convolutional Neural Network (CNN), a Long Short-Term Memory (LSTM) network, and a Transformer block. The CNN is employed to extract distinctive features within a 60-s time window, the LSTM captures temporal dependencies within this timeframe, and the attention mechanism of the transformer is utilized to prioritize crucial features.

Algorithm 2 Convolutional-LSTM-Transformer Network

Input: Time series data
Define the CNN component of the model:
a. Add a 1D convolutional layer.
b. Add layer normalization.
c. Add max pooling.
d. Add dropout.
e. Add flattening.
Define the LSTM component of the model:
a. Add a single LSTM layer.
b. Add dropout.
Define the transformer component of the model:
a. Add an embedding layer.
b. Add a multi-head attention layer.
c. Add flattening.
Concatenate the outputs of the CNN, LSTM, and transformer block.
Pass the concatenated output through a series of dense layers:
a. Add ReLU activation.
b. Add dropout.
Pass the output of the dense layers through a single dense layer with linear activation.
Compile the model with the mean squared error (MSE) loss function and mean absolute error (MAE) metrics.

4.2. The attack predictor

The main challenge in this process is determining an appropriate entropy threshold for detecting DDoS attacks in cloud environments. This challenge stems from several factors that make threshold selection complex and dynamic:

1. **Threshold Variability:** Due to the diverse nature of network traffic patterns and unique characteristics of cloud environments, a universally applicable entropy threshold does not exist. The optimal threshold is variable and largely depends on various factors such as the types of applications hosted, network infrastructure, and user behavior. Therefore, finding a one-size-fits-all threshold that effectively works across all contexts presents a significant challenge.

2. **Dynamic Network Conditions:** Rapid changes in network conditions may lead to swings in entropy levels. These fluctuations do not always signify malicious activities but can trigger false alarms if not properly accounted for in the threshold. Thus, it is vital to consider the dynamic nature of network traffic and adjust the threshold accordingly to avoid false positives or negatives.
3. **Evolving Attack Techniques:** DDoS attack methods are constantly advancing, with attackers using intricate tactics to sidestep detection systems. They might use low-rate attacks or other strategies explicitly crafted to elude traditional statistical-based detection methods. As a result, a static entropy threshold might fall short in detecting these innovative attack techniques.
4. **Balancing False Positives and False Negatives:** The process of setting the entropy threshold requires delicately balancing the minimization of false positives (identifying non-existent attacks) and false negatives (overlooking actual attacks). A low threshold could increase the false positive rate, leading to unnecessary resource utilization and potential interruption of legitimate traffic. On the contrary, a high threshold might raise the likelihood of false negatives, thus enabling attacks to go unnoticed.

To address these challenges, continuous monitoring, analysis, and adaptation are necessary. It requires a combination of advanced anomaly detection techniques, machine learning algorithms, and real-time analytics to dynamically adjust the entropy threshold based on the evolving network conditions and attack landscape. Regular updates and improvements to the detection system are essential to stay ahead of emerging DDoS attack strategies. The adopted strategy for dynamically adjusting the entropy threshold relies on the use of a decision tree model as an attack predictor. To do this, we construct the training data for the model by taking two lists, E_s and E_d, and labeling them as anomalies or normal. These lists contain the entropies of the source and destination IP addresses calculated in the second step.

4.2.1. Features preprocessing

If the attack predictor identifies a window as anomalous, the system will undertake four tasks to process windows containing suspected flows.

1. Removes duplicate rows and rows with missing values to clean the data.
2. Converts categorical data into integer values utilizing the one-hot encoding technique.
3. Unifies the data type to numeric values, whereby all data is converted into numbers and scaled to a range between 0 and 1 using the Minmax method:

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

where X_{new} is the new value, and X is the array of values of the column.

4. Uses genetic algorithms to determine the subset of features that optimize the performance of the system in terms of accuracy and F1-score. Genetic algorithms are optimization techniques that involve evolving a population of candidate solutions via the genetic operators of crossover, mutation, and selection. In the current context, each candidate solution corresponds to a distinct subset of features. The fitness of the candidate solutions is evaluated using a decision tree classifier trained solely on the respective subset of features. The fitness function considers both the F1-score and accuracy of the decision tree model as criteria to assess candidate feature subsets. Specifically, feature subsets that achieve high values for both the harmonic mean of accuracy and F1-score are taken, serving as optimal feature combinations according to these performance metrics.

In genetic algorithms, it is essential to maintain the chromosome population as it contains candidate solutions. Each chromosome comprises a specific number of bits, and this length remains constant throughout the process. For our case, the number of features is 32. The value is 0 or 1; if 0, the feature is not selected, else if 1, then selected. We start with a random initializer to generate the first set of chromosomes. Then we apply crossover with a 0.8 chance and mutation with a 0.03 chance. The mutation flips a bit in the chromosome. We keep a population of 50 chromosomes and repeat this for 100 generations. The Fig. 7 and algorithm 3 detailed the used method for feature selection.

This algorithm 3 is a genetic algorithm for feature selection. The goal is to find the optimal subset of features. The input to the algorithm is a dataset, and the output is the optimal subset of features. The algorithm starts by initializing all parameters and generating an initial population of candidate solutions. Each candidate solution is represented by a chromosome, which is a binary string indicating which features are included (1) and which are excluded (0). The algorithm then enters a loop that continues until either a maximum number of generations has been reached or an optimal fitness level has been achieved. For each generation, the algorithm evaluates the fitness of each candidate solution by projecting the chromosome onto the data to obtain a subset of features, training a decision tree (DT) on the subset, and evaluating the performance of the DT on a test set using the harmonic mean of accuracy and f1-score. The fitness of each candidate solution is then compared, and the best fitness is selected. The algorithm uses selection, crossover, mutation, and reproduction operators to create a new population of candidate solutions based on the fitness of the previous generation. The cycle repeats until the termination condition is met. The selection operator chooses the fittest individuals to be parents and generates a new population by combining their chromosomes. The crossover operator randomly selects two parents and exchanges parts of their chromosomes to create two new offspring. The mutation operator randomly flips bits in the chromosomes to introduce new genetic material. Finally, the reproduction operator creates new individuals by copying the fittest individuals from the previous generation.

Algorithm 3 Optimal Subset Selection

Require: Dataset
Ensure: Optimal Subset
Initialize all parameters
Generate initial population
while $i < maxgeneration$ and not optimal fitness **do**
  for $j = 1$ to length(P_i) **do**
    Subset ← projection($chromosome_j$, $Data$)
    Train_subset, Test_subset ← Split($subset$)
    Train($DT, Train\_subset$)
    Fitness$_j$ ← Harmonic_mean($Evaluate(DT, Test\_subset)$)
  end for
  Fitness ← max($Fitness_j$)
  Selection
  Crossover
  Mutation
  Reproduction
end while

4.3. Flow classification

Our proposed approach addresses the critical need for real-time DDoS attack detection, emphasizing the importance of processing speed in detecting attacks before they can cause irreparable damage. By leveraging entropy forecasting and incorporating real-time data processing, our system can swiftly identify deviations from the normal range and trigger alerts, allowing for immediate response and mitigation.

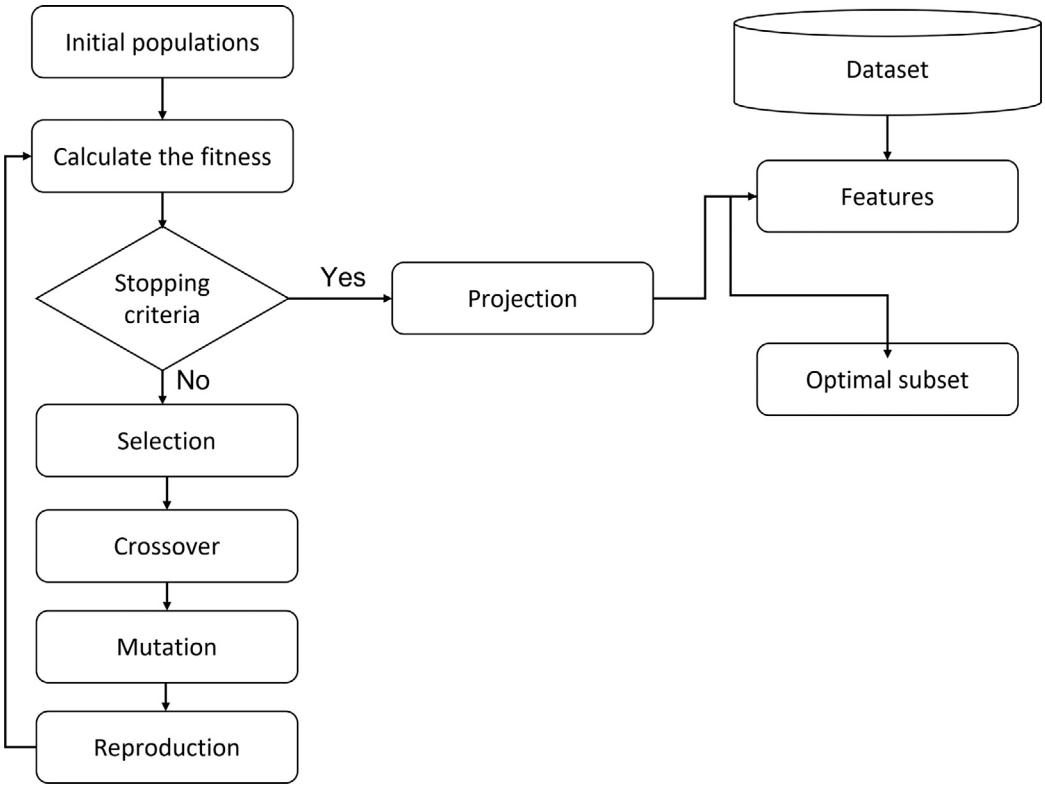

Fig. 7. Feature selection using GA.

The integration of machine learning and deep learning techniques in our classification component enables accurate identification of DDoS attacks in real-time. The use of genetic algorithms for feature selection optimizes the system's efficiency by identifying the most relevant features for detection. This feature selection process significantly reduces the dimensionality of the data, facilitating faster processing and decision-making. Furthermore, we enhance the speed and efficiency of our detection system by leveraging a CNN-based autoencoder. By compressing and extracting meaningful representations of the input data, the autoencoder reduces the computational complexity while maintaining the detection accuracy. The replacement of the fully connected layers with DT and RF classifiers within the autoencoder allows for faster classification and response to potential attacks. The real-time parameter in our approach is of paramount importance, as timely detection is crucial to preventing extensive damage caused by DDoS attacks. By detecting and alerting about attacks in real-time, our system provides an opportunity to implement countermeasures and mitigate the impact before it becomes irreparable. The combination of real-time processing, efficient feature selection, and integration of classification models within the autoencoder ensures the system's ability to quickly identify and respond to DDoS attacks, safeguarding the network infrastructure from potential harm.

The development process encompassed an extensive evaluation of different DNN (Deep Neural Network) architectures, including CNN (Convolutional Neural Network), LSTM (Long Short-Term Memory), and Autoencoder. Our customized model, AutoCNN-DT, was formulated by integrating an Autoencoder-based CNN with a Decision Tree (DT) as the output classifier. To enhance optimization, we conducted comprehensive training and testing across all models, utilizing both the full feature set and the optimal subset identified through the Genetic Algorithm for Decision Trees (GA-DT). This iterative approach enabled us to fine-tune our system for superior performance.

Fig. 8 offers an overview of the autoencoder employed to create the AE model, as depicted in Fig. 9. To construct the AutoCNN-DT model, we selected the best CNN model based on accuracy and detection time. The integration of the autoencoder was to leverage its primary advantage — the ability to compress data without sacrificing meaningful information. This advantageous feature significantly contributes to reducing detection time.

Fig. 10 and algorithm 4 illustrate the architecture of the CNN model, designed as a sequential model with multiple convolutional blocks. These blocks utilize 1D convolution to extract relevant features from the input data. To enhance the training process, we incorporate batch normalization, max pooling, and dropout layers in that order after each convolutional layer. The batch normalization layer normalizes the output of the convolutional layer, ensuring better stability and faster training. The max pooling layer reduces the output's dimensionality by applying a max filter with a pooling operation. In order to prevent overfitting, the dropout layer randomly drops out neurons during training. Following the convolutional blocks, the output is flattened and then passed to a dense layer with a SoftMax activation function. This activation function converts the dense layer's output into a probability distribution over the two classes, enabling the model to make predictions effectively.

Algorithm 4 CNN Architecture

```
Input data
for each convolutional block do
    Apply 1D convolution to extract features
    Apply batch normalization to normalize output
    Apply max pooling to reduce dimensionality
    Apply dropout to prevent overfitting
end for
Flatten output of convolutional blocks
Pass flattened output to dense layer with SoftMax activation
SoftMax activation converts output to a probability distribution over classes
Model makes predictions based on the probability distribution
```

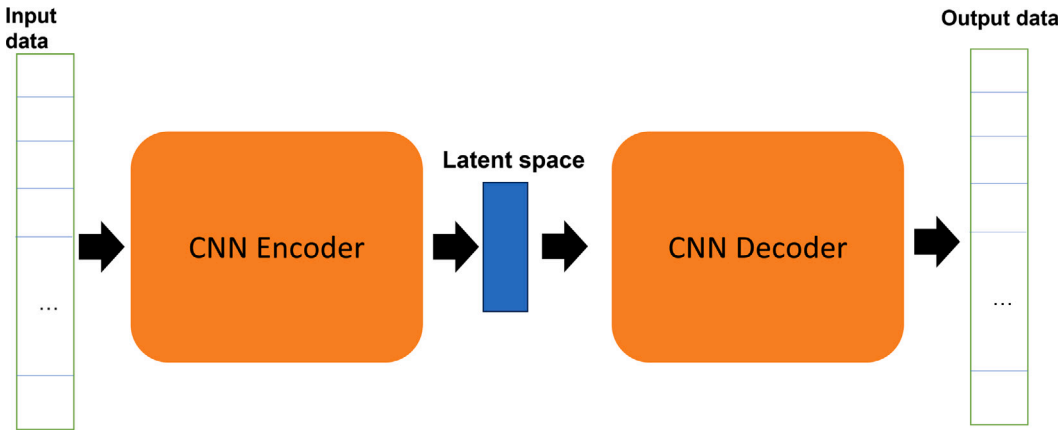

Fig. 8. AE model.

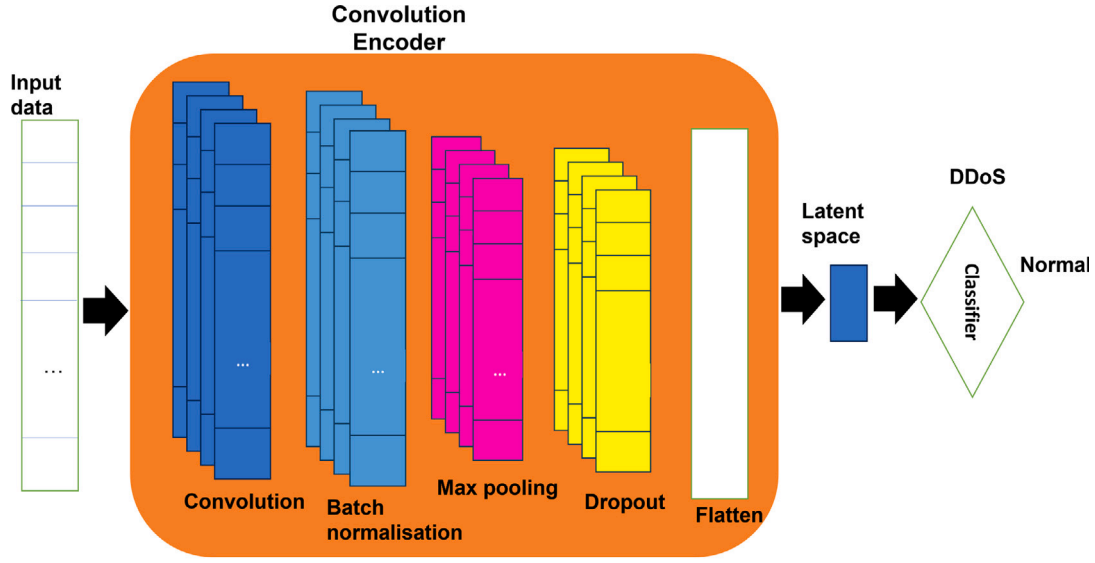

Fig. 9. The AutoCNN-DT model.

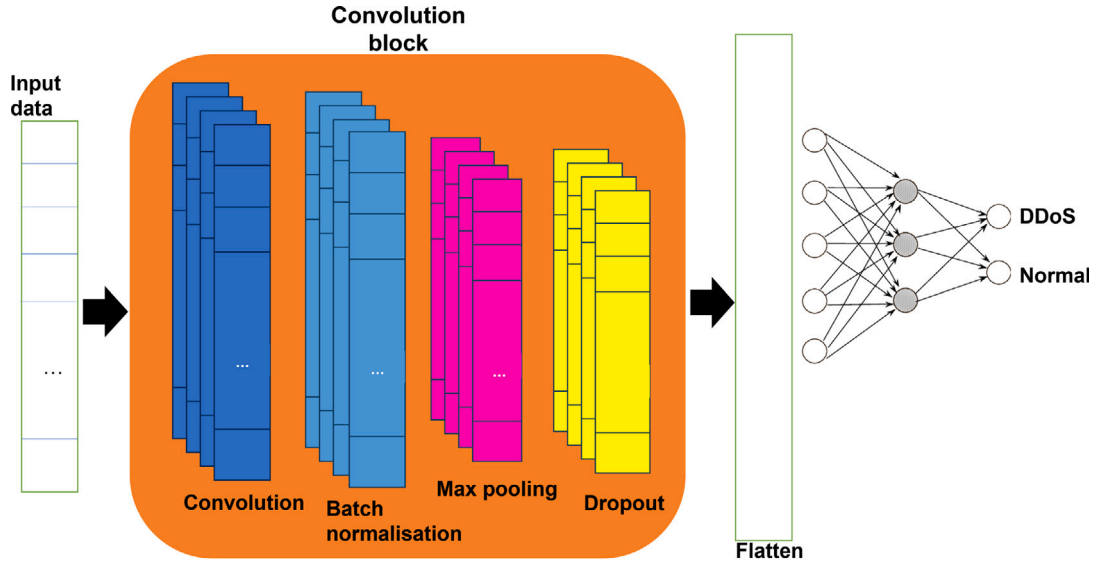

Fig. 10. The CNN model with FC.

In summary, our development process involved evaluating various DNN architectures, resulting in the creation of the specialized AutoCNN-DT model. We fine-tuned this model through extensive training and testing, utilizing both full and optimal feature sets. The integration of an Autoencoder in AutoCNN-DT reduced detection time significantly.

4.4. The functioning of the system

To illustrate the system's functioning, this section presents the steps executed by the system during its implementation in real-world situations, integrating with the network interface of the cloud. This follows the previous section, which detailed the creation and development of the framework, Figs. 11 and 12 outlines the principal components of our proposed approach. The initial stage involves entropy prediction where the system anticipates the entropies of source and destination IP addresses for the upcoming minute. These anticipated entropies are then converted into an attack predictor, which determines whether the next window is normal or anomalous. If it is detected as anomalous, the system initiates a classification process, starting with preprocessing and real-time flow classification.

Following this, the system stands by to gather the data for the current minute and compute the entropies, which are then used as inputs for the entropy prediction system to forecast the entropies for the succeeding minute.

In scenarios where the window is normal, the system uses the calculated entropies to feed the attack predictor in order to rectify any discrepancies made by the entropy prediction system. The algorithm 5, present the functioning of the system.

Algorithm 5 Entropy-based Attack Prediction System

```
while true do
    Predict the entropies of the source and destination IP addresses for the subsequent minute using the entropy forecasting system.
    Transform the predicted entropies into an attack predictor.
    Determine whether the next window is normal or anomalous using the attack predictor.
    if the window is identified as anomalous then
        Initiate the classification process.
        Preprocess the incoming data.
        Perform real-time flow classification.
    end if
    Pause momentarily to collect data for the current minute.
    Compute the entropies based on the collected data.
    Feed the calculated entropies into the attack predictor to rectify any inaccuracies made by the entropy forecasting system in the previous iteration.
    Update the entropy forecasting system with the calculated entropies for the next iteration.
end while
```

5. Experiments

This section details the experimental design and performance measures used to evaluate our proposed method. We implemented our system using Python and the TensorFlow 2.0 framework. The experiments used the CIDDS-001 dataset, which was pre-processed by removing duplicate data, normalizing it, and selecting features using genetic algorithms. The data was then divided into three groups: training data (the first 100 h of flow), validation data (the following 34 h of flow), and test data (the final 34 h of flow). Historical entropy of features was then calculated to prepare the data for time series analysis. The detection stage was based on a model that combined machine learning techniques with a deep learning architecture. Tests were performed on a Windows 10 PC with an Intel Core i7 2.8 GHz CPU and 8 GB of RAM, as well as Google's Colab (Google Colab, 2023) and Kaggle kernels (Kaggle, 2023) cloud services. Performance metrics of regression and classification were used to evaluate the system; regression was used in the forecasting stage and classification in the detection stage. Specifically, the metrics of forecasting included were:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4}$$

where y_i the real value, $\hat{y}_i$ the predicted value.

In the classification step, the evaluation criteria used to select the best model are accuracy, precision, recall, f1-score.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{8}$$

where TP: true positive; TN: true negative; FP: false positive; FN: false negative.

In Figs. 13 and 14 we depict the entropy associated with source and destination IP addresses. The blue markers represent the 60-s averages for the entropy of these IP addresses. The data suggests that for a network traffic window containing DDoS attacks, the entropy linked to the source features tends to be high, while that of the destination features remains low. This might suggest that DDoS attacks typically originate from a wide variety of sources, but they are often directed to a specific target.

During both the forecasting and classification stages, we utilized Deep Neural Network (DNN) architectures that were optimized through hyperparameter tuning. The following table showcases the various parameter values that were assessed throughout the training phase.

We carried out a range of experiments to assess the efficacy of various deep learning models. Utilizing the Tree-structured Parzen Estimator (TPE) method, we fine-tuned the hyperparameters of each model, as shown in Table 5. This table lists the specific hyperparameters and their associated interval values evaluated for every model. In the subsequent sections, we provide a detailed explanation of each hyperparameter and discuss their respective implications:

Table 5
Detailed overview of hyperparameter ranges for model tuning.

Hyper parameter	Interval values
Number of epochs	Classification: Min = 2, Max = 16 step = 1 Forecasting: Min = 10, Max = 300 step = 10
Batch size	[16,32,64,128,512,1024]
Convolution filter size	[1,3,5,7,9,11]
Number of filters	[32,64,128,256]
Strides	[1,2,3]
Pool size	[1,3,5,7]
Dropout rate	Between 25% and 75%
LSTM or GRU unit size	[16,32,64,128,512,1024]
Number of heads for transformer	[2,...,10]
Output unit size	[16,32,64,128,512,1024]
Number of layers	1,2,3
Optimizer	[Adam, Adelta, Rmsprop]

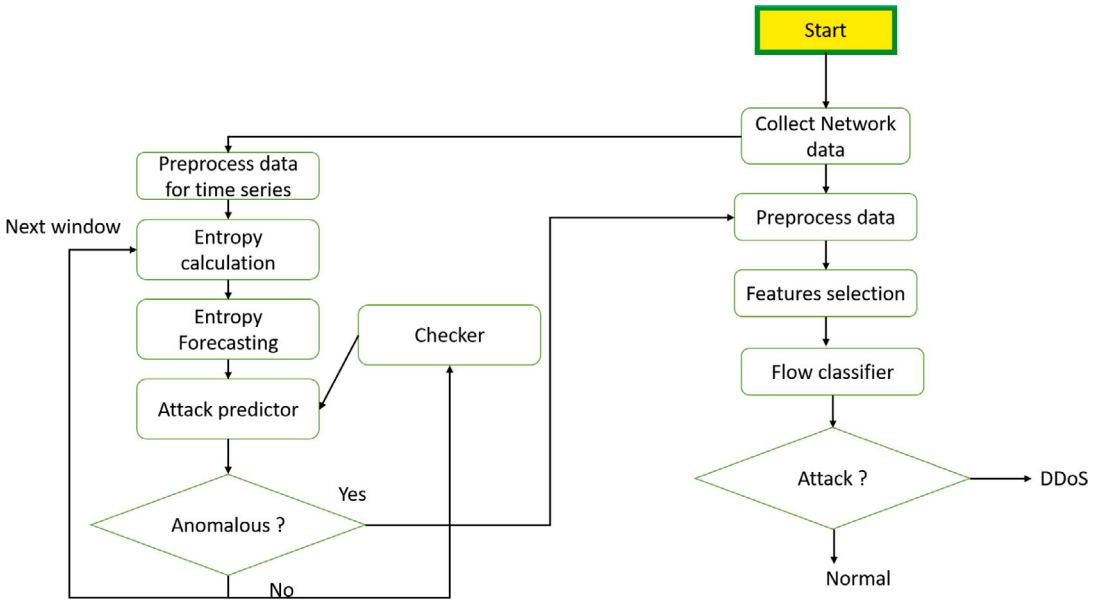

Fig. 11. The flowchart of construction and functioning of the proposed system.

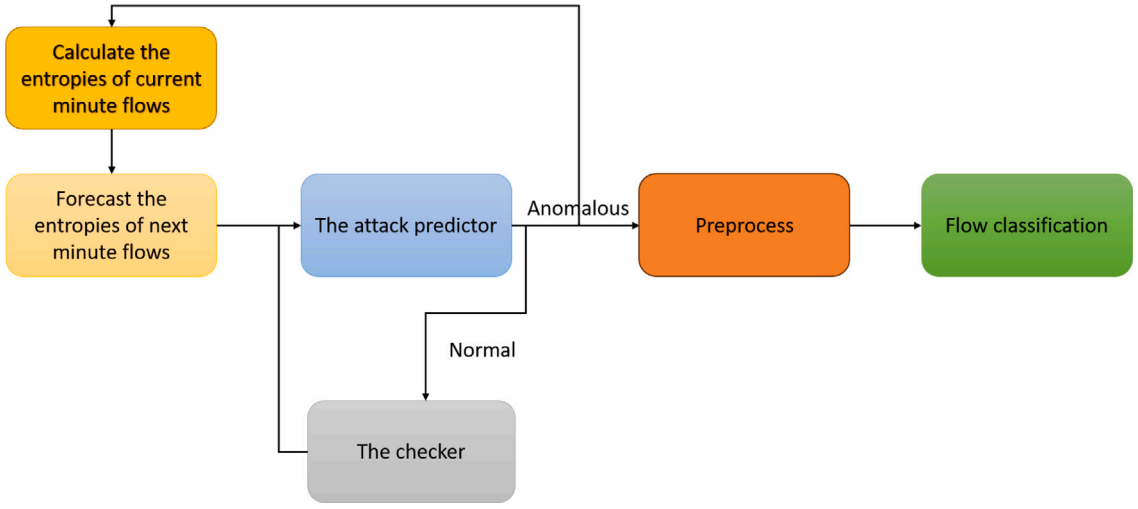

Fig. 12. An overview the functioning of the system.

- **Number of epochs:** This hyperparameter dictates the number of iterations the model will cycle through the entire training dataset. For classification, the range of values is from 2 to 16 (inclusive), with a step size of 1. For forecasting, the range of values is from 10 to 300 (inclusive), with a step size of 10.
- **Batch size:** This parameter determines the quantity of samples fed through the model simultaneously. We evaluated batch sizes of 16, 32, 64, 128, 512, and 1024.
- **Convolution filter size:** This hyperparameter sets the dimensions of the filter employed in the convolutional layer. The tested values were 1, 3, 5, 7, 9, and 11.
- **Number of filters:** This hyperparameter indicates the quantity of filters employed in the convolutional layer. We evaluated the values 32, 64, 128, and 256.
- **Strides:** This hyperparameter specifies the filter's step size as it traverses the input data. The tested values were 1, 2, and 3.
- **Pool size:** This hyperparameter defines the dimensions of the pooling window utilized in the pooling layer. We experimented with values of 1, 3, 5, and 7.
- **Dropout rate:** This is the proportion of randomly chosen neurons that will be disregarded during training. We varied this rate between 25
- **LSTM or GRU unit size:** This is the count of LSTM or GRU units in a recurrent neural network layer. We examined six different values: 16, 32, 64, 128, 512, and 1024.
- **Number of heads for transformer:** This hyperparameter determines the number of self-attention heads employed in a transformer layer. The considered values ranged from 2 to 10.
- **Output unit size:** This signifies the quantity of neurons in the output layer. We tried out six different values: 16, 32, 64, 128, 512, and 1024.
- **Number of layers:** This signifies the quantity of layers in the neural network. We tested three different values: 1, 2, and 3.
- **Optimizer:** This denotes the optimization algorithm employed during training. We tested three different values: Adam, Adelta, and Rmsprop.

We determined the ideal blend of hyperparameters for each model by targeting the maximum accuracy and F1-score, coupled with the minimum detection time for classification. Moreover, for forecasting,

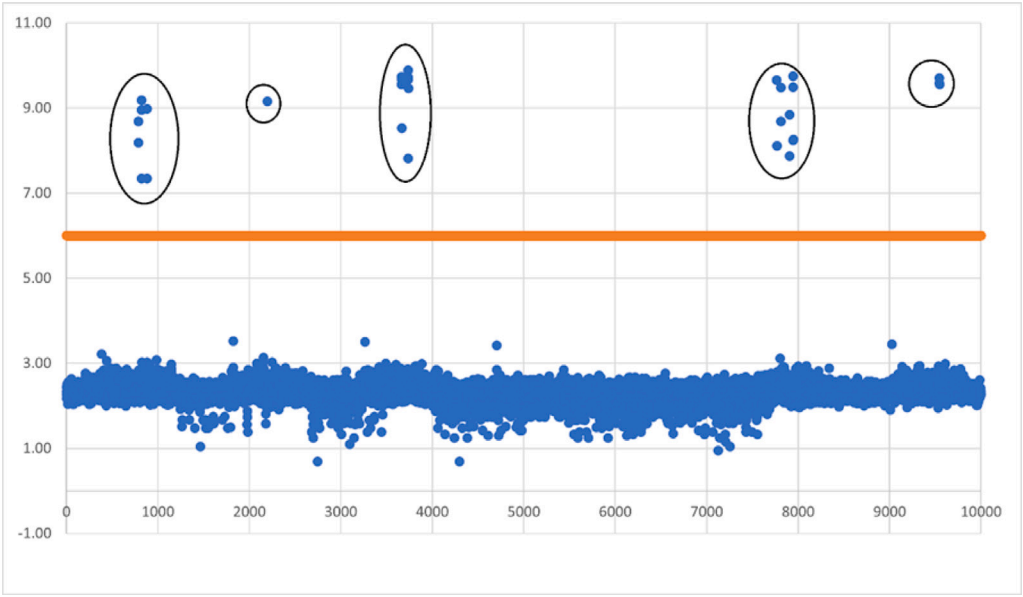

Fig. 13. Entropy of source IP over time with time frame 60 s.

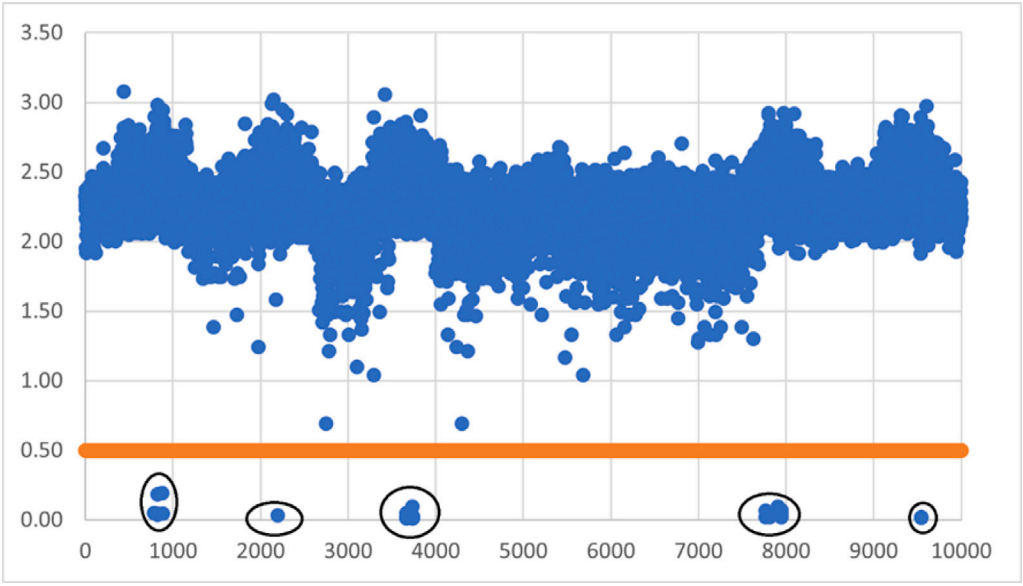

Fig. 14. Entropy of destination IP over time with time frame 60 s.

we aimed for reduced mean squared error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE).

6. Results and discussion

In this section, we present the findings from our comprehensive experiments aimed at framework construction. During the forecasting phase, we initially evaluated various ARIMA models by adjusting their parameters. Subsequently, the top-performing ARIMA model was benchmarked against several deep learning techniques, including CNN, LSTM, LSTM encoder, Transformer, and CNN-LSTM.

6.1. Evaluation and comparison of ARIMA forecasting models for IP entropy prediction

Table 6, Figs. 15–17 detail the performance of various ARIMA models in forecasting, using metrics such as MSE, MAE, and MAPE. A model's effectiveness is gauged by its ability to minimize these values. Our analysis revealed that the ARIMA (1,1,1) model outperformed others in forecasting the entropy of source IPs, as evidenced by its lowest MSE and MAE scores. This suggests superior accuracy over the other models. Additionally, its lowest MAPE score underscores its commendable relative accuracy. The ARIMA (1,0,0) and (1,1,2) models also showcased good performance, as indicated by their low MSE, MAE, and MAPE values. In contrast, the ARIMA (2,1,1) and (2,2,2) models registered higher values for these metrics, hinting at their

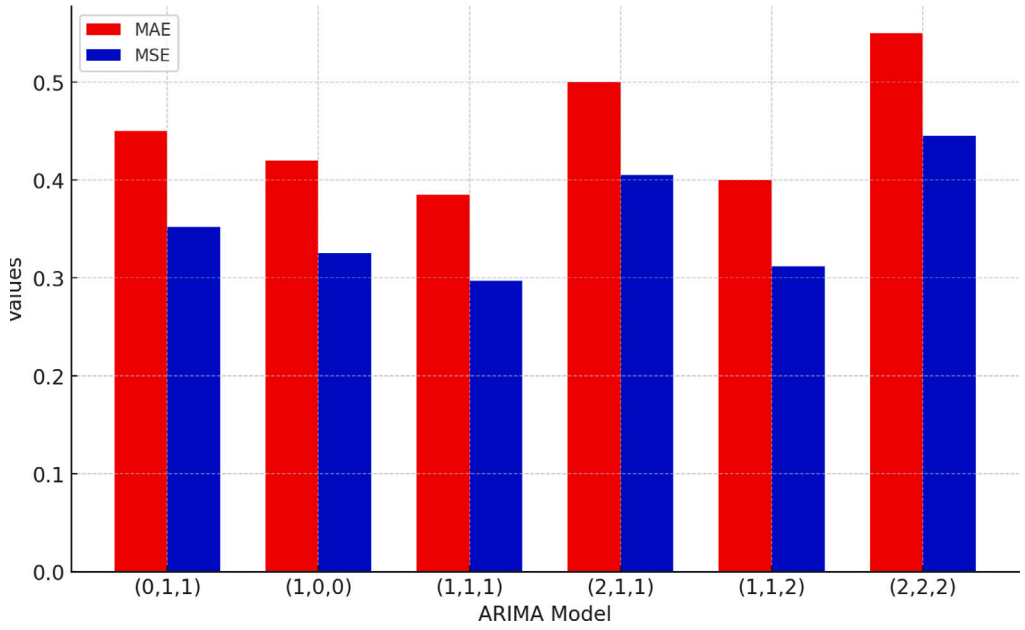

Fig. 15. MAE and MSE of entropy of source IP for different ARIMA models.

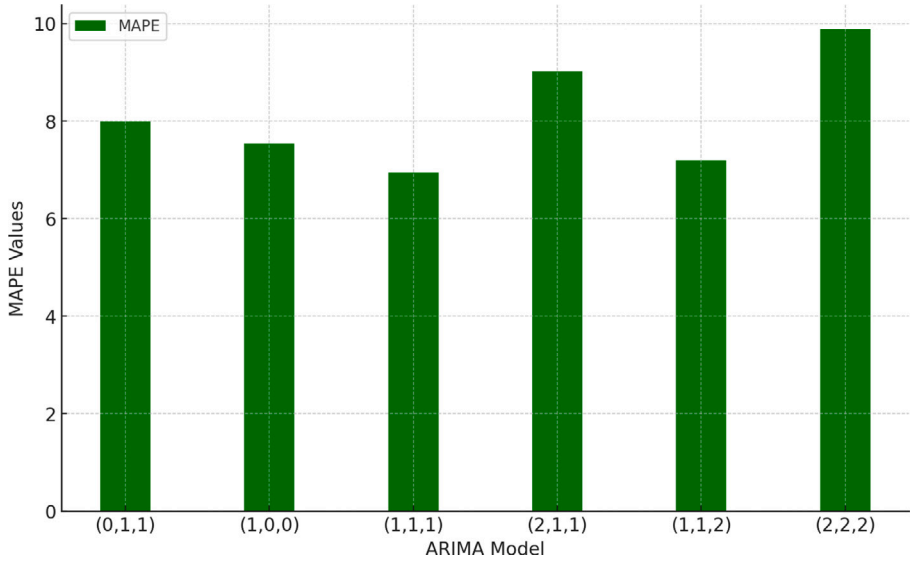

Fig. 16. Comparison of MAPE values for source IP across various ARIMA models.

Table 6
The results of ARIMA models in forecasting of source and destination IPs.

ARIMA model	MSE		MAE		MAPE	
	Source	Destination	Source	Destination	Source	Destination
(0,1,1)	0.352	0.188	0.450	**0.374**	7.986	29.081
(1,0,0)	0.325	0.195	0.420	0.389	7.542	30.113
(1,1,1)	**0.297**	0.214	**0.385**	0.411	**6.941**	31.598
(2,1,1)	0.405	**0.177**	0.500	0.395	9.014	**27.571**
(1,1,2)	0.312	0.201	0.400	0.395	7.185	30.633
(2,2,2)	0.445	0.225	0.550	0.432	9.882	33.267

reduced accuracy. When forecasting the entropy of destination IPs, the ARIMA (2,1,1) model stood out with the lowest MSE and MAPE scores, marking it as the most accurate among the models tested. Conversely, the ARIMA (2,2,2) model recorded the highest MSE and MAE values, positioning it as the least accurate model in our study.

6.2. Comparative analysis of forecasting models for IP entropy prediction: Efficacy of deep learning architectures over ARIMA

The Table 7, Figs. 18 and 19, present the results of comparing different models of forecasting in prediction the entropy of next minute of source IPs. The evaluation metrics used are MSE, MAE and MAPE the lowest indicate better performance. Based on the results, we can observe that for all three assessment measures, the LSTM-encoder, Transformer and CNN-LSTM-Transformer models perform better than all other examined models. In particular, the model transformer has achieved an MSE value of only 0.240 which represents a significant improvement over the previously best performing ARIMA (1,1,1) model with its MSE value of 0.297. Likewise, both the transformer and CNN-LSTM-Transformer models were able to reach error rates under 5% when using the MAPE measurement, indicating that they are reliably identifying patterns or trends in our data and improving forecasting. In other hand, the ARIMA model has the highest MSE and MAE values, indicating that ARIMA model not suitable for this task, as we know, it is

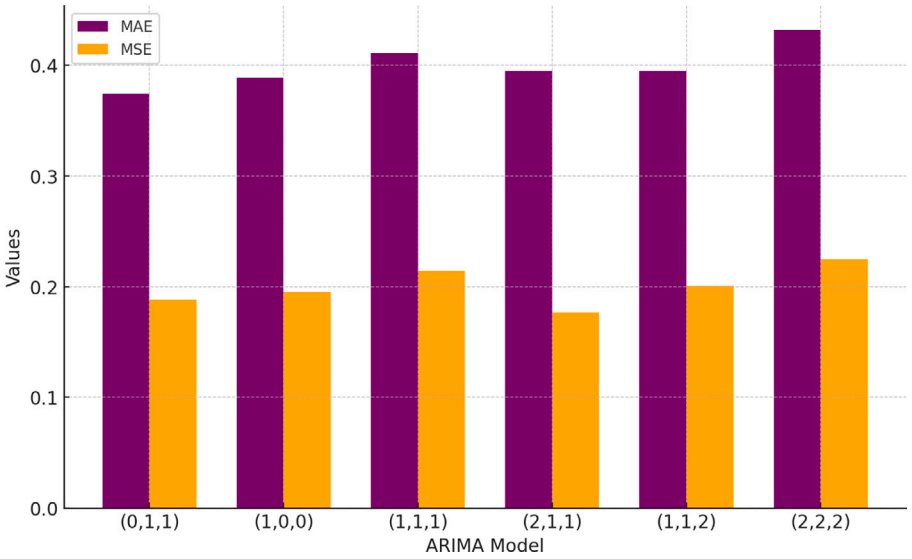

Fig. 17. Evaluation of MAE and MSE for destination IP entropy across ARIMA models.

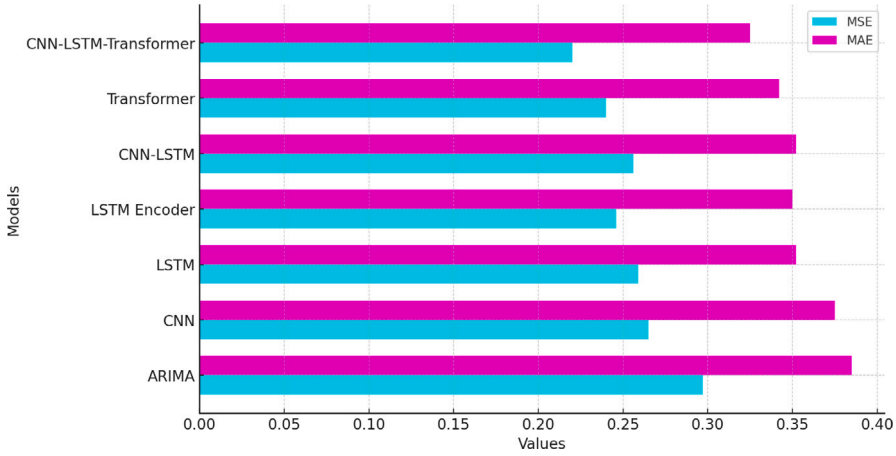

Fig. 18. MSE and MAE of different models on forecasting entropy source IPs.

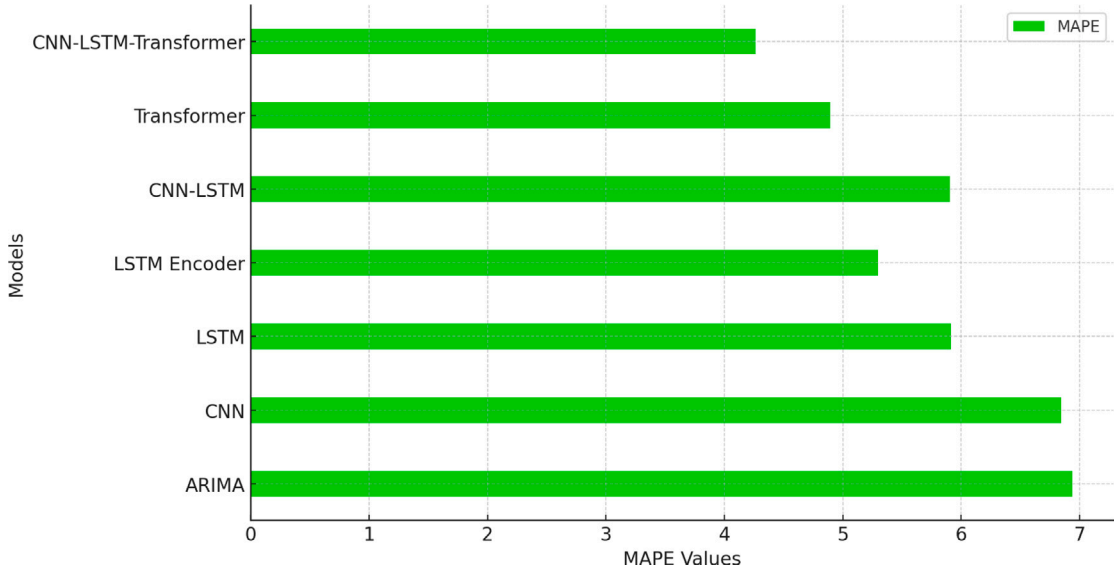

Fig. 19. MAPE of different models on forecasting entropy source IPs.

Table 7
The results of different models on forecasting entropy of source IPs.

	ARIMA	CNN	LSTM	LSTM Encoder	CNN-LSTM	Transformer	CNN-LSTM-Transformer
MSE	0.297	0.265	0.259	0.246	0.256	0.240	**0.220**
MAE	0.385	0.375	0.352	0.350	0.352	0.342	**0.325**
MAPE	6.941	6.844	5.916	5.297	5.905	4.895	**4.265**

Table 8
The results of different models on forecasting entropy of destination IPs.

	ARIMA	CNN	LSTM	LSTM Encoder	CNN-LSTM	Transformer	CNN-LSTM-Transformer
MSE	0.177	0.153	0.190	0.156	0.174	0.142	**0.135**
MAE	0.359	0.331	0.378	0.331	0.357	0.325	**0.320**
MAPE	27.571	27.746	26.695	26.170	26.377	24.231	**23.785**

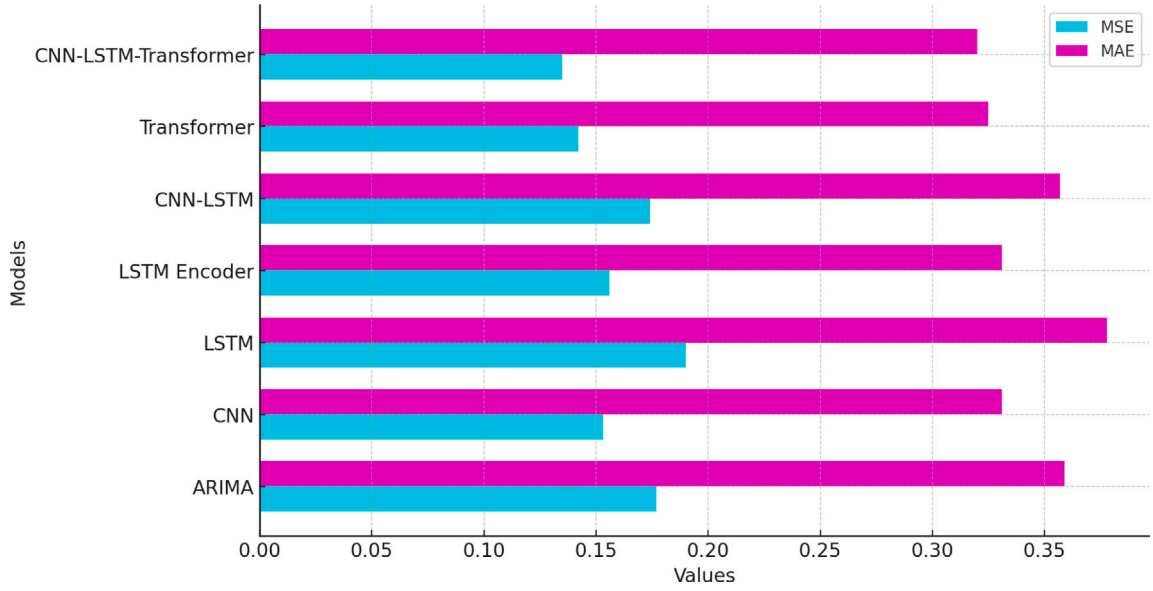

Fig. 20. MSE and MAE of different models on forecasting entropy destination IPs.

difficult to ARIMA models to handling the non-linear trends. The CNN, LSTM, and CNN-LSTM models also perform well, but not as good as the LSTM-Encoder, CNN-LSTM-Transformer, and Transformer models. Overall, the results suggest that advanced deep learning architectures including, LSTM-Encoder, CNN-LSTM-Transformer, and Transformer can be effective in detecting potential security threats in cloud computing systems, outperforming traditional statistical models such as ARIMA and other less complex deep learning models such as CNN and LSTM.

The Table 8, Figs. 20 and 21 compares the performance of seven different forecasting models—ARIMA, CNN, LSTM, LSTM Encoder, CNN-LSTM, Transformer, and CNN-LSTM-Transformer on the entropy of destination IPs. Each model's performance is evaluated on three metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). Of all the models, the CNN-LSTM-Transformer model performs best, achieving the lowest error rates across all three metrics (MSE of 0.135, MAE of 0.320, and MAPE of 23.785). Following this, the Transformer model and the CNN model offer the next best performances. The results suggest that more advanced, combination models like the CNN-LSTM-Transformer tend to perform better on this task, possibly due to their ability to better handle the complexity and temporal dependencies in the data. The integration of CNN, LSTM, and Transformer architecture in the CNN-LSTM-Transformer model yielded superior results across all three metrics, namely lowest MSE, lowest MAE, and lowest MAPE. This indicates that combining these models can lead to more accurate predictions compared to using them separately. Similarly, when forecasting entropy source IPs, the ARIMA model was outperformed by deep learning models. These findings suggest that deep learning models, particularly those incorporating CNN, LSTM, and Transformers, can deliver a robust Cloud Security system. Overall, the CNN-LSTM-Transformer model appears to be the highest performing model across all three criteria, followed closely by the Transformer model.

The Fig. 22, show the architecture of the best model in term of MSE, MAE and MAPE in forecasting of entropy source IPs. Similarly, for forecasting entropy of destination IPs.

The Fig. 23, present the training and validation loss of the best model on forecasting of source IPs. The difference in the parameters with the same architecture. As we mentioned in proposed approach section, the checker is a decision tree model with the input the forecasted entropy of source and destination IPs, the output anomalous or normal window. The model is trained based on a dataset that contains the calculated entropies and evaluated with the same data. The test results show that decision tree model was able to accurately classify the windows with 100% accuracy.

As previously discussed, once the attack predictor notify that a windows is anomalous, a classification process is initiated.

6.3. Evaluating threshold techniques in DDoS detection

In Table 9 various methods for setting thresholds in DDoS attack detection, as outlined in Refs. Daneshgadeh et al. (2019), Koay et al. (2018), Kesavamoorthy and Ruba Soundar (2018), Idhammad et al. (2018), Aladaileh et al. (2023), Kareem and Jasim (2022) and Shetty (2022), present distinct challenges and considerations. The Mahalanobis distance method (Daneshgadeh et al., 2019) is prone to false

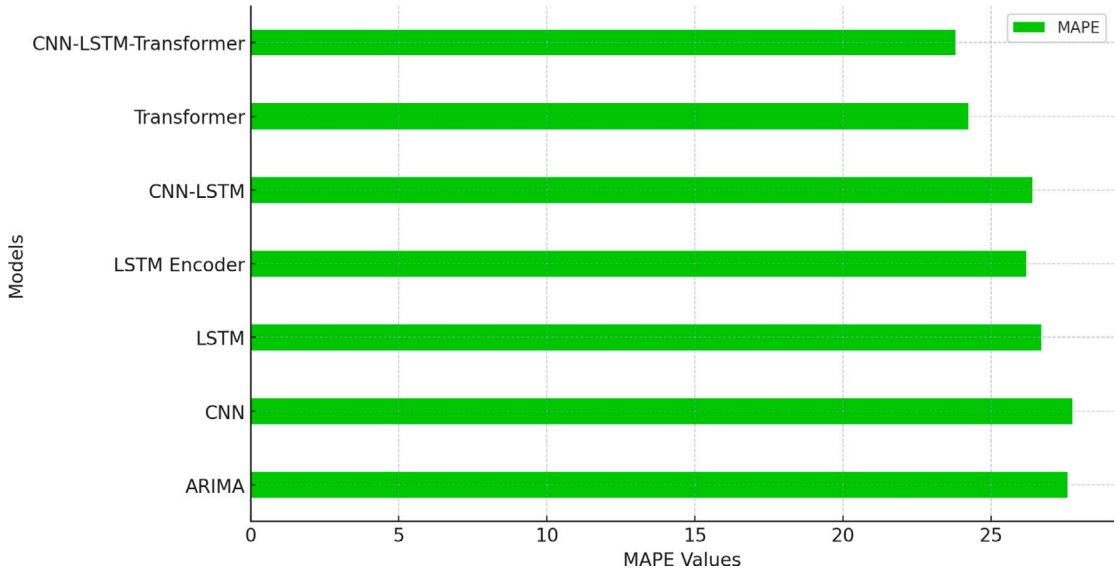

Fig. 21. MAPE of different models on forecasting entropy destination IPs.

Table 9
Comparative analysis of threshold setting methods in DDoS attack detection.

Ref.	Method of threshold	Summarized comments
Daneshgadeh et al. (2019)	Mahalanobis distance	Prone to errors, especially with DDoS attacks similar to Flash Events.
Koay et al. (2018)	Machine learning techniques	Challenges in managing classifier complexity.
Kesavamoorthy and Ruba Soundar (2018)	Specified experimentally	Risk of misidentification if network behavior deviates from expected patterns.
Idhammad et al. (2018)	Specified experimentally	Effectiveness depends on the network environment and attack types.
Aladaileh et al. (2023)	Specified experimentally	Less effective against low-rate attacks; issues with static threshold values.
Kareem and Jasim (2022)	Specified experimentally	Needs real-world adjustments; threshold calibration is crucial.
Shetty (2022)	Specified experimentally	Effectiveness varies with network configurations; careful threshold setting needed.

alarms or missed detections, particularly with DDoS attacks resembling Flash Events. Machine learning techniques (Daneshgadeh et al., 2019) face difficulties in optimal threshold selection and classifier complexity. Several experimentally specified methods (Kesavamoorthy and Ruba Soundar, 2018; Idhammad et al., 2018; Aladaileh et al., 2023; Kareem and Jasim, 2022; Shetty, 2022) highlight the risk of misidentifying attack instances due to deviation from anticipated network patterns, the dependence of effectiveness on specific network environments and attack types, challenges in detecting low-rate DDoS attacks due to their similarity to normal traffic, and the need for careful calibration and adjustment of thresholds to match network traffic patterns and avoid false positives or negatives. These observations underscore the complexity and dynamic nature of effectively detecting DDoS attacks.

Our method of dynamic thresholding, when integrated with a decision tree model, achieves near-perfect accuracy in the classification of network traffic windows as normal or anomalous for DDoS attack detection. This system's use of dynamic thresholds allows it to adeptly adjust to fluctuating network conditions, significantly reducing false positives and negatives. The decision tree model is particularly adept at interpreting complex data structures, making it highly effective in identifying the subtle distinctions between normal traffic and sophisticated low-rate DDoS attacks. This approach is exquisitely tailored to the dynamic and intricate nature of network traffic, offering a highly accurate and adaptive solution for distinguishing between normal and anomalous behavior within traffic windows, with an accuracy score approaching the ideal 1.0000.

6.4. Efficiency of AutoCNN-DT model in DDoS attack detection

In Table 10, we provide the outcomes from all the classification models, which include CNN, LSTM, CNN+DT, CNN+RF, LSTM+DT, and LSTM+RF. Table 10 and Fig. 24 demonstrate high accuracy across all models, with the highest F1 score signaling strong predictive capabilities as well as substantial precision and recall. When considering detection time performance, however, there are noticeable differences. The quickest models are CNN+FC and CNN+DT, clocking in at 5.98 s and 5.32 s respectively, while the slowest are LSTM+DT and LSTM+RF, at 20.52 s and 21.6 s respectively. The LSTM and CNN models coupled with Decision Trees (DT) and Random Forests (RF) outshine the CNN+FC and LSTM+FC models.

Table 10
The results of the classification using all features.

Model	Accuracy	Precision	Recall	F1-score	Time detection
CNN+FC	0.9926	0.9933	0.9919	0.9925	5.98
CNN+DT	0.9996	0.9996	0.9996	0.9996	5.32
CNN+RF	0.9995	0.9997	0.9995	0.9996	11.72
LSTM+FC	0.9925	0.9932	0.9917	0.9924	14.75
LSTM+DT	0.9997	0.9997	0.9997	0.9997	20.52
LSTM+RF	0.9997	0.9997	0.9997	0.9997	21.6

Where:

- **CNN+FC** and **LSTM+FC**: These terms refer to networks combined with a fully connected network and logistic function.
- **CNN+DT** and **LSTM+DT**: These terms represent networks that integrate a decision tree in the output, instead of traditional neural networks.
- **CNN+RF** and **LSTM+RF**: These terms signify networks that incorporate a random forest in the output, rather than traditional neural networks.

To expedite the detection process, we harnessed the power of genetic algorithms and decision trees to isolate a subset of features that

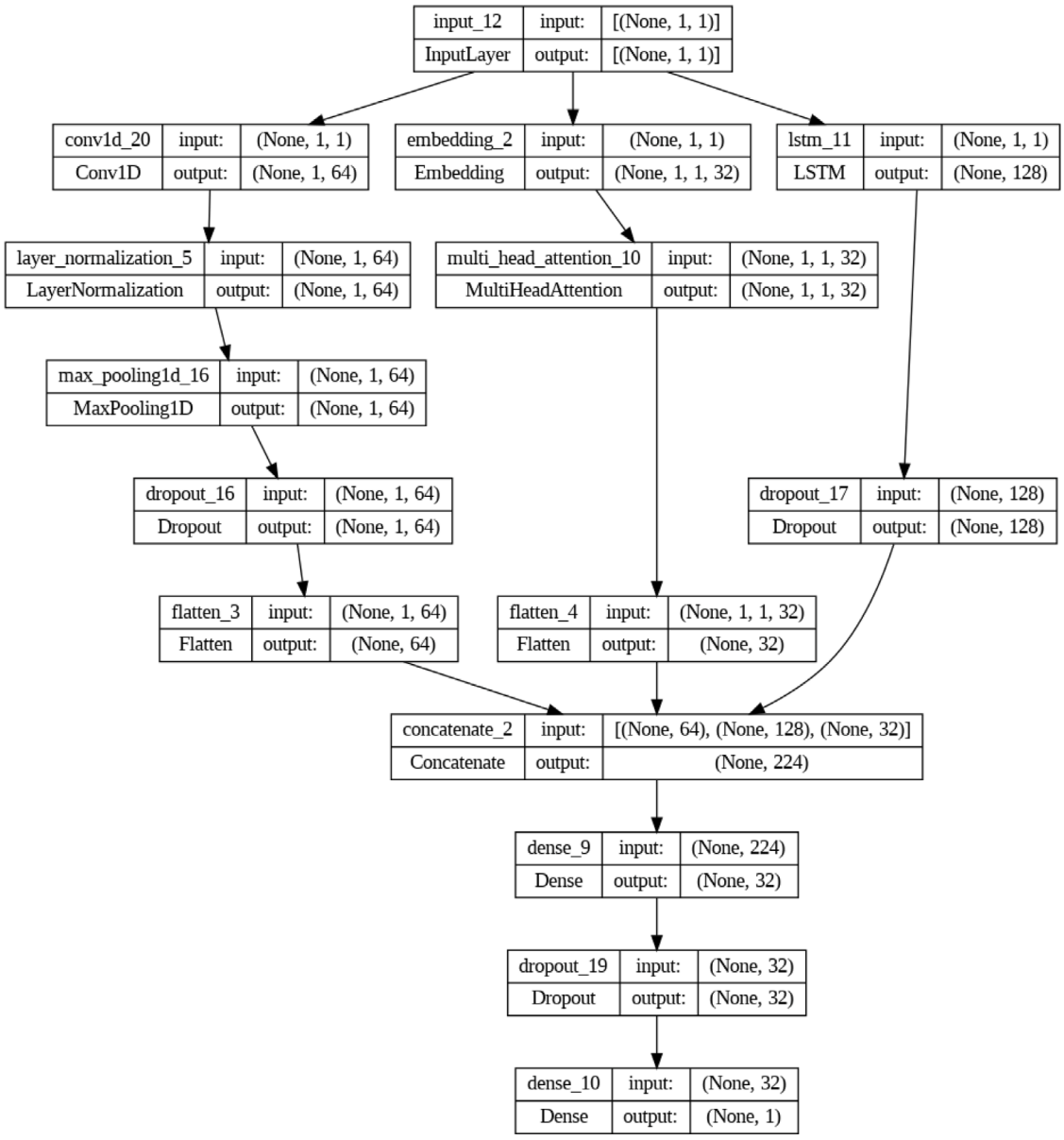

Fig. 22. The architecture of the best model.

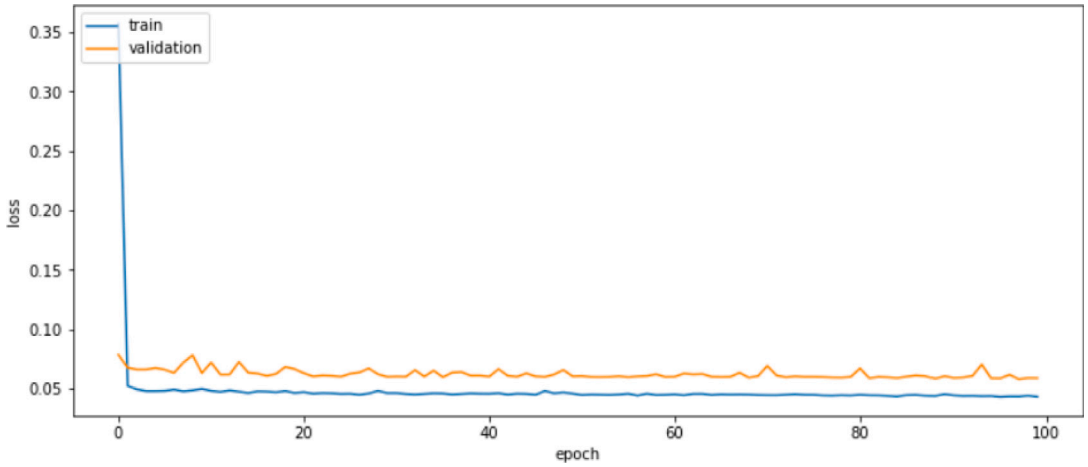

Fig. 23. Training and validation loss of the CNN-LSTM-Transformer model (Entropy source IP).

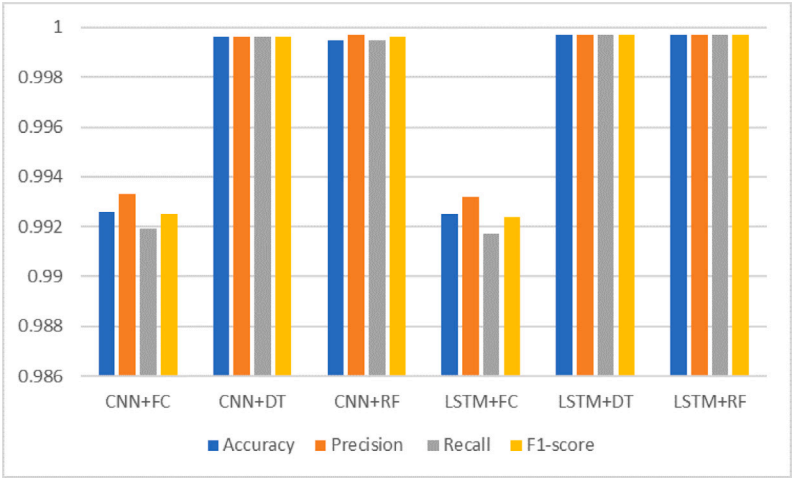

Fig. 24. The results of classification with all features.

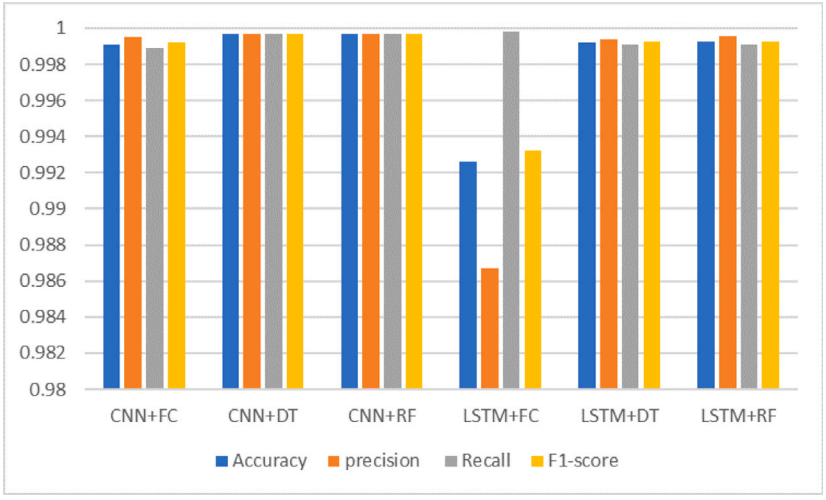

Fig. 25. The results of classification with optimal subset.

Table 11
The results of classification with the best subset.

Model	Accuracy	Precision	Recall	F1-score	Time detection
CNN+FC	0.9991	0.9995	0.9989	0.9992	5.89
CNN+DT	0.9997	0.9997	0.9997	0.9997	5.18
CNN+RF	0.9997	0.9997	0.9997	0.9997	7.01
LSTM+FC	0.9926	0.9867	0.9998	0.9932	11.9
LSTM+DT	0.9992	0.9994	0.9991	0.9993	11.47
LSTM+RF	0.9993	0.9996	0.9991	0.9993	12.37

Table 12
The best parameters of CNN model for classification.

Model parameter	Value		
Batch size		32	
Number of layers		3	
Drop rate	0.23	0.16	0.11
Number of filters	32	64	256
Kernel	3	3	1
Pool size	2	2	1
Optimizer		Adam	
Number of epochs		10	

yielded the topmost F1-score. This method of feature selection downsized the feature set from 32 to 11. Table 11 and Fig. 25 displays the outcomes from all models employing this optimally identified subset of features, derived via genetic algorithms.

According to Table 11 and Fig. 25, All models had very good accuracy, precision, recall, and F1-score values ranging from 0.9926 to 0.9997. This shows that the models are extremely effective in detecting DDoS attacks in cloud computing systems, with very low false positives rate and false negatives rate. In terms of time detection, the CNN+FC model was the fastest, detecting attacks in 5.89 s, while the LSTM+RF model was the slowest, requiring 12.37 s. It can be seen that the accuracy and F1 score for many models is improved when using optimum subsets in a model, while also reducing detection time across all models.

In Figs. 27 and 26, we show the architecture of CNN model that reach the best results in terms of accuracy, f1-score and detection time. In the Table 12, we present the best parameters of CNN model obtained by Bayesian hyper-parameters tuning.

The CNN autoencoder (AE) is being used to extract and compress data, which is then fed into the decision tree for classification. This approach is intended to improve the efficiency of the detection process by reducing the amount of data that needs to be processed. the combination of a convolutional neural network (CNN) autoencoder and a decision tree (DT) has achieved high accuracy and a f1-score of 0.9997 in detecting DDoS attacks in network traffic. In terms of detection time, this model appears to be the most efficient, with a detection time of 3.88 s. The combination of an autoencoder with a convolutional neural

Layer (type)	Output Shape	Param #
conv1d_89 (Conv1D)	(None, 9, 32)	128
batch_normalization_65 (BatchNormalization)	(None, 9, 32)	128
max_pooling1d_65 (MaxPooling1D)	(None, 4, 32)	0
dropout_58 (Dropout)	(None, 4, 32)	0
conv1d_90 (Conv1D)	(None, 2, 64)	6208
batch_normalization_66 (BatchNormalization)	(None, 2, 64)	256
max_pooling1d_66 (MaxPooling1D)	(None, 1, 64)	0
dropout_59 (Dropout)	(None, 1, 64)	0
conv1d_91 (Conv1D)	(None, 1, 256)	16640
batch_normalization_67 (BatchNormalization)	(None, 1, 256)	1024
max_pooling1d_67 (MaxPooling1D)	(None, 1, 256)	0
dropout_60 (Dropout)	(None, 1, 256)	0
flatten_1 (Flatten)	(None, 256)	0

Total params: 24,384
Trainable params: 23,680
Non-trainable params: 704

Fig. 26. The architecture of the CNN model with the details of each layer.

network (AE-CNN) and a decision tree (DT) has significantly reduced the time needed for detection compared to using a CNN and DT with all features. Specifically, the AutoCNN-DT model has reduced detection time by 72% compared to the CNN-DT with all features and by 69% compared to the CNN-DT with an optimal subset of features obtained with genetic algorithms.

6.5. Comparative evaluation of DDoS detection and prevention techniques

When comparing the effectiveness of different techniques for the detection and prevention of DDoS attacks in Table 13, several key themes emerge:

1. **Models:** Deep learning techniques like LSTM and CNN and ensemble methods like Random Forest tend to yield high detection accuracy. It is notable that the attention mechanism with Bi-LSTM also showed promising results.
2. **Datasets:** Both synthetic and real-world datasets are used. Synthetic datasets allow for control over specific attack scenarios, but public datasets provide a more realistic picture of the network environment.
3. **Evaluation Metrics:** Accuracy, precision, recall, and F1-score are commonly used to evaluate the effectiveness of the models, with many achieving high scores. Other valuable metrics are detection time, detection rate, false positive rate, and in some cases, True Positive Rate (TPR) and True Negative Rate (TNR).
4. **Prevention Mechanisms:** Only a few techniques incorporate prevention mechanisms, which are crucial for stopping an attack in its tracks. In these cases, the results show a trend towards better overall performance, especially when paired with machine learning models like Random Forest.

Our technique with a CNN-DT model and includes a prevention mechanism, stands out due to its extremely high accuracy and F1-score

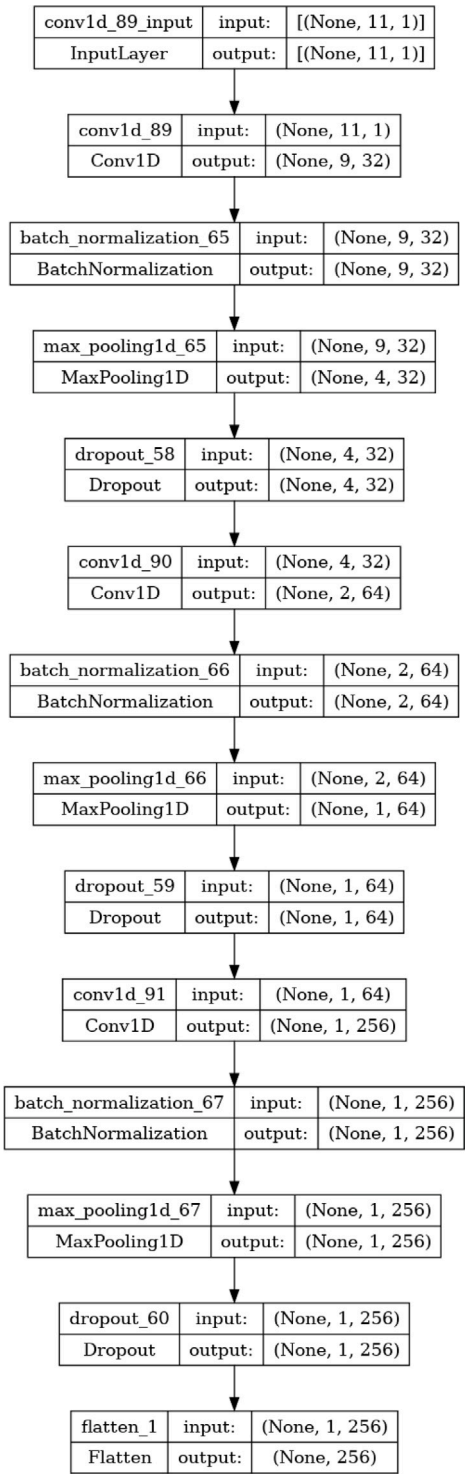

Fig. 27. The architecture of the CNN model.

(0.9997), and reasonable detection time (3.88 s), indicating a highly effective approach for both detecting and preventing DDoS attacks.

In conclusion, while many techniques show promise in detecting DDoS attacks, the inclusion of prevention mechanisms in the system design significantly enhances their real-world utility. Future research should concentrate on enhancing DDoS threat detection speed and incorporating stronger prevention mechanisms, with a particular focus on creating suitable datasets in the cloud environment for time series analysis, to maximize the efficiency of combating such threats.

Table 13
Comparison of machine learning techniques used for DDoS attacks detection.

Ref.	Dataset	Technique	Prev	Evaluation metrics	Results
Liu et al. (2023)	ISCX-2016 and 1999 DARPA	Bi-LSTM and attention mechanism	No	Accuracy, precision, and recall	Accuracy: 0.9868
Bamasag et al. (2022)	CAIDA DDoS Attack 2007, Synthetic dataset and NSL-KDD	Random forest	Yes	Accuracy	Accuracy: 0.9938
Sanjalawe and Althobaiti (2023)	CICIDS 2017	CNN and LSTM	No	Accuracy, precision, and recall	Accuracy: 0.9790, precision: 0.9830, recall: 0.9790, F1-score: 0.9810
Alduailij et al. (2022)	CICIDS 2017 and CICDDoS 2019	Random forest	Yes	Accuracy	Accuracy: 0.9999
Balasubramaniam et al. (2023)	Synthetic dataset	Deep stacked autoencoder	Yes	Accuracy, TPR, and TNR	Accuracy: 0.9017, TPR: 0.9090, TNR: 0.9090
Priyadarshini and Barik (2022)	ISCX 2012 and Synthetic dataset	LSTM	Yes	Accuracy	Accuracy: 0.9948
Aladaileh et al. (2023)	Synthetic dataset	Entropy	No	Detection rate and False positive rate	DR: 0.9700, FPR: 0.0300
Kareem and Jasim (2022)	Synthetic dataset	Entropy	No	Detection time	DT: 0.445 s
Shetty (2022)	NSL-KDD2009 and CIC-IDS2017	LSTM	No	Accuracy, precision, recall, and F1-score	Accuracy: 0.9910, precision: 0.9880, recall: 0.9940, F1-score: 0.9910
Our	CIDDS-001	CNN-DT	Yes	Accuracy, precision, recall, F1-score, and detection time	Accuracy: 0.9997, F1-score: 0.9997, DT: 3.88 s

6.6. Proactive detection of DDoS attacks using forecasting entropy in online environments

Our research demonstrates a significant improvement in early DDoS attack detection through the use of forecasting entropy in online systems. This approach addresses the common challenges of complexity and scalability in online environments. Notably, DDoS attacks usually occur within a brief period, often in seconds or minutes, and rarely hours. In our study, these attacks represented only 0.31% of the total monitoring time. Traditional detection methods in online contexts are often resource-intensive. However, our system distinguishes itself by deploying its maximum computational capabilities for less than 1% of the time, targeting the specific, short durations of DDoS attacks. This strategy significantly boosts efficiency and reduces the system's resource demands, offering a more effective solution for managing cybersecurity threats compared to continuous, high-load operations in other systems.

6.7. Insights from SHAP value analysis

In order to render our system interpretable and to identify features that greatly influence classification, we implement the SHAP method. SHAP, or SHapley Additive exPlanations, is a technique employed to elucidate the predictions of a machine learning model and ascertain the impact of each feature on the model's prediction. By calculating the SHAP values for every feature and graphically representing them, we can more effectively comprehend which features predominantly influence the model's predictions. This method also aids in uncovering any potential biases or anomalies in the model's conduct.

Table 14 displays the 11 selected features following the preprocessing stage. In contrast, Table 4 represents the original features, whereas Table 14 showcases the features after undergoing preprocessing.

The Figs. 28, 29 illustrate that TCP flags have the largest Shap value, indicating that they have the greatest impact on the DDoS detection. This suggests that the TCP flags are more significant in determining whether an attack is present than the other features being analyzed.

TCP Flags are used in TCP (Transmission Control Protocol) to control and manage the state of a connection. Each flag corresponds to a bit in the TCP header. There are six TCP flags:

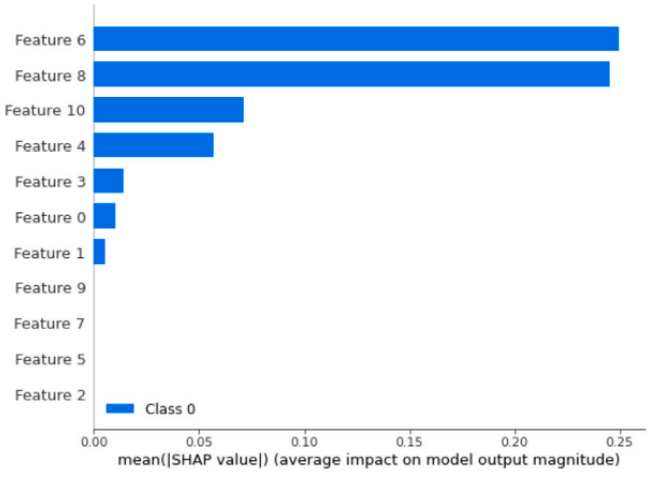

Fig. 28. The average Shap value of features.

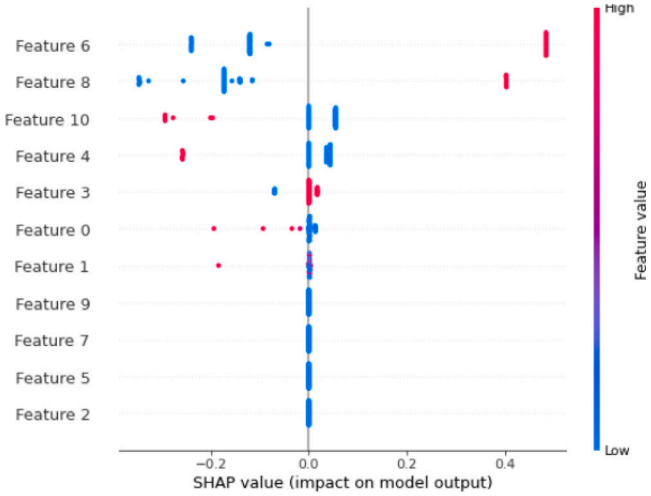

Fig. 29. Shap value of features.

Table 14
The 11 selected features.

Feature 1	Duration
Feature 2	Bytes
Feature 3	Proto_IGMP
Feature 4	Proto_TCP
Feature 5	Flags_.....
Feature 6	Flags_....S.
Feature 7	Flags_.A..SF
Feature 8	Flags_.A.R.F
Feature 9	Flags_.AP.SF
Feature 10	Flags_.APRSF
Feature 11	Tos_32

- URG (Urgent): Indicates that the data should be processed urgently.
- ACK (Acknowledgment): Acknowledges receipt of a packet.
- PSH (Push): Asks to push the buffered data to the receiving application.
- RST (Reset): Resets the current connection.
- SYN (Synchronize): Initiates a connection.
- FIN (Finish): Closes a connection.

The columns in the dataset represent different combinations of these flags.

- **Flags_.....S.** - Indicates a SYN flag, used to establish TCP connections
- **Flags_...R..** - Indicates a RST flag, used to reset TCP connections
- **Flags_...RS.** - Combination of SYN and RST flags
- **Flags_.A....** - ACK flag, indicates acknowledgment of received data
- **Flags_.A...F** - ACK and FIN flags set, indicates end of data
- **Flags_.A..S.** - ACK and SYN flags
- **Flags_.A..SF** - ACK, SYN, and FIN
- **Flags_.A.R..** - ACK and RST
- **Flags_.A.R.F** - ACK, RST, and FIN
- **Flags_.A.RS.** - ACK, RST, and SYN
- **Flags_.A.RSF** - ACK, RST, SYN, and FIN

Type of Service (ToS)

The Type of Service (ToS) field in the IP header is used to instruct routers and switches how to handle the packet priority and the route the packet should take. While this field has been largely deprecated and replaced by the Differentiated Services (DS) field, it is still used in some cases.

The ToS field is divided into five sub-fields:

- Precedence: This 3-bit field defines the priority of the packet.
- Delay: If this bit is set, it indicates that the packet should be sent with low delay.
- Throughput: If this bit is set, it indicates that high throughput is important.
- Reliability: If this bit is set, it indicates that high reliability is important.
- Cost: If this bit is set, it indicates that the route with the lowest cost should be used.

7. Conclusion

Our research introduces the DeepDefend framework as an innovative solution for detecting and mitigating DDoS attacks in cloud environments. This framework stands out due to its integration of advanced machine learning and deep learning techniques, which enable accurate and prompt detection of potential attacks. Key components such as the CNN-LSTM-Transformer for entropy forecasting, genetic algorithms for data pre-processing optimization, and the AutoCNN-DT for real-time attack prediction significantly enhance the efficiency of the DDoS detection process. Moreover, the incorporation of an autoencoder (AE) in our CNN+DT model has been instrumental in reducing detection times.

One of the most significant findings of our research is the substantial improvement in early detection of DDoS attacks through entropy forecasting in online systems. This approach effectively addresses the complexities and scalability challenges in cloud environments. Given that DDoS attacks typically occur within very short time frames, often in seconds or minutes, our system's ability to focus its computational power during these brief periods—representing only 0.31% of our total monitoring time—marks a major advancement. Unlike traditional detection methods that are resource-intensive, our framework optimizes resource usage by concentrating on these critical moments, thereby enhancing overall efficiency and reducing the burden on system resources.

Looking ahead, we plan to implement the DeepDefend framework in real-world cloud settings to test its adaptability and resilience under varied and unpredictable conditions. We also aim to create a dedicated dataset for time series analysis in cloud environments, which will be invaluable for refining our approach and contributing to broader research in the field.

Moreover, we intend to delve deeper into the features identified by the Shap method to gain a fuller understanding of their influence on the detection process. This exploration will enable more precise feature engineering and optimization strategies, further improving the accuracy and speed of our detection methods.

Overall, our research lays a robust foundation for enhancing DDoS detection and prevention strategies in cloud environments. The continued development and application of the DeepDefend framework hold great promise for strengthening cloud security, an essential aspect of our increasingly interconnected digital world.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work we used Chatgpt-4 in order to improve the linguistic and writing capabilities when writing this paper. After using this service, we reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

Ahalawat, Anchal, Dash, Shashank Sekhar, Panda, Abinas, Babu, Korra Sathya, 2019. Entropy based ddos detection and mitigation in OpenFlow enabled SDN. In: 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking. ViTECoN, IEEE, pp. 1–6. http://dx.doi.org/10.1109/ViTECoN.2019.8711169.

Aladaileh, Mohammad Adnan, Bahashwan, Abdullah Ahmed, Anbar, Mohammed, Al-Amiedy, Taief Alaa, Hintaw, Ahmed J, Hasbullah, Iznan H, Ibrahim, Dyala R, 2023. Effectiveness of an entropy-based approach for detecting low- and high-rate ddos attacks against the SDN controller: Experimental analysis. Appl. Sci. (ISSN: 2076-3417) 13 (2), 775. http://dx.doi.org/10.3390/app13020775, URL https://www.mdpi.com/2076-3417/13/2/775.

Alduailij, Mona, Khan, Qazi Waqas, Tahir, Muhammad, Sardaraz, Muhammad, Alduailij, Mai, Malik, Fazila, 2022. Machine-learning-based ddos attack detection using mutual information and random forest feature importance method. Symmetry 14 (6), 1095. http://dx.doi.org/10.3390/sym14061095.

Ali, Mazhar, Khan, Samee U., Vasilakos, Athanasios V., 2015. Security in cloud computing: Opportunities and challenges. Inform. Sci. 305, 357–383. http://dx.doi.org/10.1016/j.ins.2015.01.025.

Armbrust, Michael, Fox, Armando, Griffith, Rean, Joseph, Anthony D., Katz, Randy, Konwinski, Andy, Lee, Gunho, Patterson, David, Rabkin, Ariel, Stoica, Ion, Zaharia, Matei, 2010. A view of cloud computing. Commun. ACM 53 (4), 50–58. http://dx.doi.org/10.1145/1721654.1721672.

Balasubramaniam, S, Joe, C Vijesh, Sivakumar, T A, Kumar, K Satheesh, Kavitha, V, Dhanaraj, Rajesh Kumar, Prasanth, A, 2023. Optimization enabled deep learning-based ddos attack detection in cloud computing. Int. J. Intell. Syst. 2023, 2039217. http://dx.doi.org/10.1155/2023/2039217.

Bamasag, Omaimah, Alsaeedi, Alaa, Munshi, Asmaa, Alghazzawi, Daniyal, Alshehri, Suhair, Jamjoom, Arwa, 2022. Real-time ddos flood attack monitoring and detection (RT-AMD) model for cloud computing. PeerJ Comput. Sci. 7, e814. http://dx.doi.org/10.7717/peerj-cs.814.

Bhardwaj, Aanshi, Mangat, Veenu, Vig, Renu, 2020. Hyperband tuned deep neural network with well posed stacked sparse AutoEncoder for detection of ddos attacks in cloud. IEEE Access 8, 181916–181929. http://dx.doi.org/10.1109/ACCESS.2020.3028690.

Bhushan, Kriti, Gupta, B.B., 2018. Distributed denial of service (ddos) attack mitigation in software defined network (SDN)-based cloud computing environment. J. Ambient Intell. Humaniz. Comput. 9 (6), 1753–1766. http://dx.doi.org/10.1007/s12652-018-0800-9.

Cheng, Jieren, Sheng, Victor S, Li, Mengyang, Tang, Xiangyan, Liu, Yifu, Guo, Wei, 2018. Flow correlation degree optimization driven random forest for detecting ddos attacks in cloud computing. Secur. Commun. Netw. 2018, 6459326. http://dx.doi.org/10.1155/2018/6459326.

Cisco, 2020. Cisco annual internet report (2018–2023) white paper. URL https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Accessed: July 29, 2023.

Cook, Sam, 2023. 20+ Ddos attack statistics and facts for 2018–2023. URL https://www.comparitech.com/blog/information-security/ddos-statistics-facts/. Retrieved July 29, 2023.

Cybersecurity Ventures, 2020. The 15 top ddos statistics you should know in 2020. URL https://cybersecurityventures.com/the-15-top-ddos-statistics-you-should-know-in-2020/. Accessed on: July 29, 2023.

Daneshgadeh, Salva, Ahmed, Tarem, Kemmerich, Thomas, Baykal, Nazife, 2019. Detection of ddos attacks and flash events using Shannon entropy, koad and mahalanobis distance. In: 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops. ICIN, IEEE, pp. 222–229. http://dx.doi.org/10.1109/ICIN.2019.8685891.

Fadaei Fouladi, Ramin, Kayatas, Cemil Eren, Anarim, Emin, 2018. Statistical measures: Promising features for time series based ddos attack detection. In: Proceedings of the International Workshop on Computational Intelligence for Multimedia Understanding. IWCIM, MDPI, p. 96. http://dx.doi.org/10.3390/proceedings2020096.

Fouladi, Ramin Fadaei, Ermiş, Orhan, Anarim, Emin, 2020. A ddos attack detection and defense scheme using time-series analysis for SDN. J. Inf. Secur. Appl. 54, 102587. http://dx.doi.org/10.1016/j.jisa.2020.102587.

2023. Google colab. Accessed: 28-Jul-2023. https://colab.research.google.com/.

Gupta, B.B., Badve, Omkar P., 2016. Taxonomy of DoS and ddos attacks and desirable defense mechanism in a cloud computing environment. Neural Comput. Appl. 27 (8), 2179–2195. http://dx.doi.org/10.1007/s00521-016-2317-5.

Hezavehi, Sasha Mahdavi, Rahmani, Rouhollah, 2020. An anomaly-based framework for mitigating effects of ddos attacks using a third party auditor in cloud computing environments. Cluster Comput. 23 (6), 4723–4735. http://dx.doi.org/10.1007/s10586-019-03031-y.

Idhammad, Mohamed, Afdel, Karim, Belouch, Mustapha, 2018. Detection system of HTTP ddos attacks in a cloud environment based on information theoretic entropy and random forest. Secur. Commun. Netw. 2018, 1263123. http://dx.doi.org/10.1155/2018/1263123.

Kaggle, 2023. Kaggle: Your machine learning and data science community. Accessed: 28-Jul-2023. https://www.kaggle.com/.

Kareem, Mohammed Ibrahim, Jasim, Mahdi Nsaif, 2022. Entropy-based distributed denial of service attack detection in software-defined networking. Indones. J. Electr. Eng. Comput. Sci. (ISSN: 2502-4752) 27 (3), 1542–1549. http://dx.doi.org/10.11591/ijeecs.v27.i3.pp1542-1549.

Kesavamoorthy, R., Ruba Soundar, K., 2018. Swarm intelligence based autonomous ddos attack detection and defense using multi agent system. Cluster Comput. 21 (3), 1477–1486. http://dx.doi.org/10.1007/s10586-018-2365-y.

Knorr, Eric, Gruman, Galen, 2008. What Cloud Computing Really Means. InfoWorld.

Koay, Abigail, Chen, Aaron, Welch, Ian, Seah, Winston K.G., 2018. A new multi classifier system using entropy-based features in ddos attack detection. In: 2018 International Conference on Information Networking. ICOIN, IEEE, pp. 162–167. http://dx.doi.org/10.1109/ICOIN.2018.8343214.

Kushwah, Gopal Singh, Ranga, Virender, 2020. Voting extreme learning machine based distributed denial of service attack detection in cloud computing. J. Inf. Secur. Appl. 53, 102532. http://dx.doi.org/10.1016/j.jisa.2020.102532.

Liu, Zengguang, Guo, Cuiyun, Liu, Deyong, Yin, Xiaochun, 2023. An asynchronous federated learning arbitration model for low-rate ddos attack detection. IEEE Access 11, 18448–18458. http://dx.doi.org/10.1109/ACCESS.2023.3247512.

Mather, Timothy, Kumaraswamy, Subra, Latif, Shahed, 2011. Securing the cloud. In: Cloud Security: A Comprehensive Guide To Secure Cloud Computing. Elsevier, pp. 29–52. http://dx.doi.org/10.1016/B978-1-59749-592-9.00002-6.

Mell, Peter, Grance, Timothy, 2011. The NIST Definition of Cloud Computing. Technical Report 800–145, National Institute of Standards and Technology, http://dx.doi.org/10.6028/NIST.SP.800-145.

Ni, Tongguang, Gu, Xiaoqing, Wang, Hongyuan, Li, Yu, 2013. Real-time detection of application-layer ddos attack using time series analysis. J. Control Sci. Eng. 2013, 821315. http://dx.doi.org/10.1155/2013/821315.

Nicholson, Paul, 2022. Five most famous ddos attacks and then some. URL https://www.a10networks.com/blog/5-most-famous-ddos-attacks/. Accessed: July 29, 2023.

Osanaiye, Opeyemi, Cai, Haibin, Choo, Kim-Kwang Raymond, Dehghantanha, Ali, Xu, Zheng, Dlodlo, Mqhele, 2016. Ensemble-based multi-filter feature selection method for ddos detection in cloud computing. EURASIP J. Wireless Commun. Networking 2016 (1), 130. http://dx.doi.org/10.1186/s13638-016-0623-3.

Priyadarshini, Rojalina, Barik, Rabindra Kumar, 2022. A deep learning based intelligent framework to mitigate ddos attack in fog environment. J. King Saud Univ.-Comput. Inf. Sci. 34 (2), 825–831. http://dx.doi.org/10.1016/j.jksuci.2019.04.010.

Ring, Markus, Wunderlich, Sarah, Grüdl, Dominik, Landes, Dieter, Hotho, Andreas, 2017. Flow-based benchmark data sets for intrusion detection. In: Proceedings of the 16th European Conference on Cyber Warfare and Security. ACPI. pp. 361–369.

Sanjalawe, Yousef, Althobaiti, Turke, 2023. Ddos attack detection in cloud computing based on ensemble feature selection and deep learning. CMC: Comput. Mater. Contin. 75 (2), 3572–3589. http://dx.doi.org/10.32604/cmc.2023.037386.

Sharafaldin, Iman, Lashkari, Arash Habibi, Ghorbani, Ali A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization.. ICISSp 1, 108–116.

Shetty, Anuj, 2022. Detection of ddos attack in SDN network using entropy in pox controller.

Shidaganti, Ganeshayya Ishwarayya, Inamdar, Amogh Shreedhar, Rai, Sindhuja V., Rajeev, Anagha M., 2020. SCEF: A model for prevention of ddos attacks from the cloud. Int. J. Cloud Appl. Comput. 10 (3), 67–81. http://dx.doi.org/10.4018/IJCAC.2020070104.

Simmon, Eric, 2018. Evaluation of Cloud Computing Services Based on NIST SP 800-145. Special Publication 500–322, National Institute of Standards and Technology, http://dx.doi.org/10.6028/NIST.SP.500-322.

Tabatabaie Nezhad, Seyyed Meysam, Nazari, Mahboubeh, Gharavol, Ebrahim A, 2016. A novel DoS and ddos attacks detection algorithm using ARIMA time series model and chaotic system in computer networks. IEEE Commun. Lett. http://dx.doi.org/10.1109/LCOMM.2016.2517622.

Velliangiri, S., Karthikeyan, P., Kumar, V. Vinoth, 2020. Detection of distributed denial of service attack in cloud computing using the optimization-based deep networks. J. Exp. Theor. Artif. Intell. 32 (5), 781–797. http://dx.doi.org/10.1080/0952813X.2020.1744196.

Virupakshar, Karan B, Asundi, Manjunath, Channal, Kishor, Shettar, Pooja, Patil, Somashekar, Narayan, DG, 2020. Distributed denial of service (ddos) attacks detection system for OpenStack-based private cloud. Procedia Comput. Sci. 167, 2297–2307. http://dx.doi.org/10.1016/j.procs.2020.03.282.

Yan, Qiao, Yu, F. Richard, 2015. Security and privacy in emerging networks: Distributed denial of service attacks in software-defined networking with cloud computing. IEEE Commun. Mag. 53 (4), 52–57. http://dx.doi.org/10.1109/MCOM.2015.7081084.

Yan, Qiao, Yu, F. Richard, Gong, Qingxiang, Li, Jianqiang, 2015. Software-defined networking (SDN) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges. IEEE Commun. Surv. Tutor. http://dx.doi.org/10.1109/COMST.2015.2487361.

Zhang, Qi, Cheng, Lu, Boutaba, Raouf, 2010. Cloud computing: state-of-the-art and research challenges. J. Internet Serv. Appl. 1, 7–18. http://dx.doi.org/10.1007/s13174-010-0007-6.