INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGIES
BULGARIAN ACADEMY OF SCIENCES
CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **25**, No 4
Sofia • 2025 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2025-0034

Edge-Fog-Cloud Distributed Architecture for Intelligent DDoS Detection and Mitigation

Hedjaz Sabrine[1], *Baadache Abderrahmane*[2], *Semchedine Fouzi*[3]

[1]*Research Unit LaMOS, Faculty of Exact Sciences, Department of Computer Science, University of Bejaia, 06000, Bejaia, Algeria*
[2]*Research Unit LaMOS, University of Bejaia, 06000, Algeria, Department of Computer Science, University of Algiers, 16000, Algiers, Algeria*
[3]*Department of Computer Science, University of Setif, 19000, Algeria*
E-mails: sabrine.hedjaz@univ-bejaia.dz (corresponding author) a.baadache@univ-alger.dz fouzi.semcheddine@univ-setif.dz

***Abstract*:** *Cloud and distributed infrastructures face significant challenges from increasingly sophisticated Distributed Denial-of-Service* (*DDoS*) *attacks. Real-time efficiency is limited by the latency and scalability issues that affect traditional centralized detection systems. This paper presents a multi-layered DDoS detection and mitigation framework built on the Edge-Fog-Cloud paradigm. Hierarchical intelligence is integrated into the architecture to strike a balance between adaptive defense, resource efficiency, and responsiveness. A threshold-guided lightweight classifier quickly distinguishes malicious, suspicious, and benign traffic at the edge. A compact Deep Neural Network* (*DNN*) *verifies anomalies in suspicious flows that are escalated to the fog. For context-aware mitigation, a deep classifier at the cloud layer categorizes confirmed attacks into two main families: reflection/amplification and exploitation. Evaluation on the CICDDoS2019 dataset demonstrates high accuracy, a low false-positive rate, and efficient traffic handling. The modular design ensures scalability and adaptability for modern distributed computing infrastructures.*

***Keywords*:** *Edge-Fog-Cloud architecture, DDoS detection, Lightweight threshold-based filtering, DNN, Adaptive mitigation.*

1. Introduction

One of the most widely used paradigms today is cloud computing, which enables organizations to access scalable and cost-effective services for multiple users without heavy investments in hardware or software. However, it is this benefit that makes cloud infrastructure vulnerable to DDoS attacks. The elasticity and abundant resources of the cloud provide opportunities for attackers to launch large-scale attacks. The problem continues to worsen with the growing adoption of edge and fog computing, as these decentralized environments introduce resource constraints, heterogeneous traffic patterns, and the need for rapid decision-making to meet real-time requirements. Traditional centralized or purely deep learning-based detection systems can achieve high accuracy; however, they incur high latency and computational costs, making them unsuitable for deployment at the network edge. Moreover, most existing solutions do not distinguish between suspicious and confirmed malicious traffic, and they lack adaptive multi-layer mitigation strategies.

To address these gaps, this paper proposes a multi-layer architecture for DDoS detection and mitigation that integrates the edge, fog, and cloud layers. It enables lightweight, threshold-guided classification at the edge for fast filtering and early detection. At the fog layer, a DNN model is employed to validate suspicious traffic with high precision without overloading the edge. The cloud layer performs attack-type classification to differentiate between exploitation-based and reflection-based attacks and deploys tailored mitigation strategies, such as virtual machine migration for exploitation attacks, or upstream filtering and Border Gateway Protocol (BGP) rerouting for reflection-based threats.

Our architecture is evaluated using the CICDDoS2019 dataset [1], which includes twelve DDoS attack types categorized into two main classes: exploitation and reflection. The results show that the proposed approach achieves high detection accuracy with minimal latency, reduces false positives by delegating uncertain cases across layers, and enables scalable, adaptive, and context-aware protection for modern distributed infrastructures. The main contributions of this paper are:

- A scalable, three-layer DDoS detection architecture suitable for Edge-Fog-Cloud environments.
- A threshold-guided detection mechanism at the edge for fast, lightweight classification.
- A deep neural network at the fog layer for validating suspicious flows.
- A deep neural network classifier at the cloud layer to distinguish between exploitation and reflection attacks, enabling context-aware mitigation.

The rest of the paper is structured as follows: Section 2 reviews related work. Section 3 describes the proposed approach and deployment architecture. Section 4 presents the design and evaluation of each layer. Finally, Section 5 concludes the paper.

2. Related works

Recent advancements in DDoS detection for Edge-Fog-Cloud environments have shown exceptional accuracy; however, limitations in scalability, deployment efficiency, and the absence of multi-layer defense mechanisms continue to affect most of the methods described below. For instance, AlSaleh, Al-Samawi and Nissirat [2] achieved multi-class detection in a cloud environment with reasonably high accuracy using two Bayesian Convolutional Neural Network (CNN) models: BaysCNN and BaysFusCNN. These models, however, suffer from deep, resource-intensive architectures and address only cloud-level detection, without supporting lightweight edge detection, fog refinement, or multi-layer mitigation. Ec-Sabery et al. [3] designed the GA-OELM model, a Genetic Algorithm-Optimized Extreme Learning Machine, which enables efficient real-time inference, cloud-level processing, and partial analysis of suspicious traffic through Shapley-based interpretation. However, their approach does not incorporate early mitigation mechanisms, fog-centric learning, or a multi-layered architectural design.

Some studies have explored early-stage or distributed detection, as illustrated by the framework proposed by Songa and Karri [4], Recursive Detection and Anomaly Evaluation in Routing (RDAER). The authors embedded clustering and feature elimination in Software-Defined Networking (SDN) switches; their approach enables prompt anomaly detection at the edge, marking one of the few with support for early mitigation at edge nodes. Nonetheless, RDAER lacks layered ML refinement, graduated mitigation strategies, and its suspicious traffic handling remains partially rule-based. Amitha and Srivenkatesh [5] propose a cloud-centric hybrid Radial Basis Function-Long Short Term Memory (RBFLSTM) architecture, which offers high accuracy and robust detection but acts only in a passive role; it includes neither fog level refinement nor any mitigation capabilities.

The evolution of fog-based solutions is notable. Gudla et al. [6] embedded Deep Normalized Multi-Layer Perceptron (DNMLP) classifiers at fog nodes within a hierarchical Edge-Fog-Cloud architecture. The proposed system demonstrates multi-layer scalability and a fog-level ML approach, but it only supports binary classification and does not implement adaptive mitigation or proactive filtering at the edge. An Adaptive Neuro-Fuzzy Inference System (ANFIS) was integrated with SDN for fog-level classification and immediate SDN packet blocking by Bensaid et al. [7]. This outcome demonstrates their adept management of suspicious traffic and use of fog-based decision-making. However, this method is exclusively designed to address SYN Flood attacks and, as a result, cannot be generalized to encompass other threats or layers. Pokale, Sharma and Mane [8] proposed a hybrid model with Deep Maxout and Deep Belief Networks (DBN) with regard to enhanced preprocessing and feature optimization techniques. Their system supports refinement at the fog layer and multi-layer scalability. However, it remains limited to offline hybrid optimization with no means for dynamic edge mitigation or graduated responses.

More recently, Selim and Sadek [9] presented a fog layer intrusion detection framework based on Bidirectional LSTM (BiLSTM), which was extensively preprocessed to improve accuracy as well as latency. Although their model targets fog-level detection and machine learning based refinement, it lacks edge-level deployment and adaptive mitigation strategies. Furthermore, it imposes a significant computational burden on fog resources.

Overall, the comparison in Table 1 reveals clear trends and persistent gaps in current DDoS detection research. Most recent works focus on high-accuracy deep learning models at the cloud or fog levels to achieve strong classification performance, but they involve heavy computation workloads and low scalability. Although a few studies explore early detection at the edge, suspicious traffic handling remains either rule-based or partial, and most existing solutions lack a coordinated multi-layer defense strategy or graduated mitigation process that adapts to attack severity and network context. Addressing these gaps, our proposed architecture combines lightweight DDoS detection at the edge, deep learning refinement at the fog, and attack classification at the cloud. The proposed framework addresses these identified gaps by reducing false positives, enhancing real-time responsiveness, minimizing upstream traffic, and enabling a scalable, collaborative mitigation strategy across all layers.

Table 1. Comparative analysis of related DDoS detection and mitigation studies

Feature/Work	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	Our work
Edge-level lightweight detection	×	×	×	×	×	×	×	×	✓
Fog layer refinement (ML/DL-based)	×	×	×	×	✓	✓	✓	✓	✓
Cloud-level final classification	✓	✓	✓	✓	✓	×	✓	×	✓
Suspicious traffic handling	×	Partial	Partial	×	×	✓	×	×	✓
Early mitigation at the edge layer	×	×	✓	×	×	×	×	×	✓
Graduated mitigation Strategy	×	×	×	×	×	×	×	×	✓
Real-time performance focus	Partial	✓	✓	✓	Partial	✓	✓	✓	✓
Multi-layer architecture scalability	×	×	Partial (Fog, Cloud)	×	✓	Partial (Fog, SDN)	✓	×	✓

3. Proposed architecture

In this section, we present the proposed three-layer DDoS detection and mitigation architecture, organized into edge, fog, and cloud layers that cooperate to deliver intelligent, scalable, and context-aware responses to distributed attacks. Our architecture supports early filtering, local analysis, and hierarchical mitigation, while tailored mitigation strategies are applied at each layer according to its capabilities and the corresponding detection confidence level.

For reliable coordination among layers, this work adopts a hierarchical Message Queuing Telemetry Transport (MQTT) communication architecture, inspired by the approach of Kurdi and Thayananthan [10]. To enhance the security of industrial IoT deployments, Kurdi and Thayananthan [10] deployed a multi-tier MQTT system with multiple brokers across fog and cloud layers. MQTT enables low-overhead and asynchronous message exchange between layers, which is a lightweight publish/subscribe protocol offering quality of service options and support for Transport Layer Security (TLS) [11]. In that work [10], the authors demonstrated that distributed brokers can reduce latency, network congestion, and overload on the central broker by bringing communication closer to end devices and reducing upstream traffic. Each broker tier contributes to maintaining scalability and resilience within the distributed system.

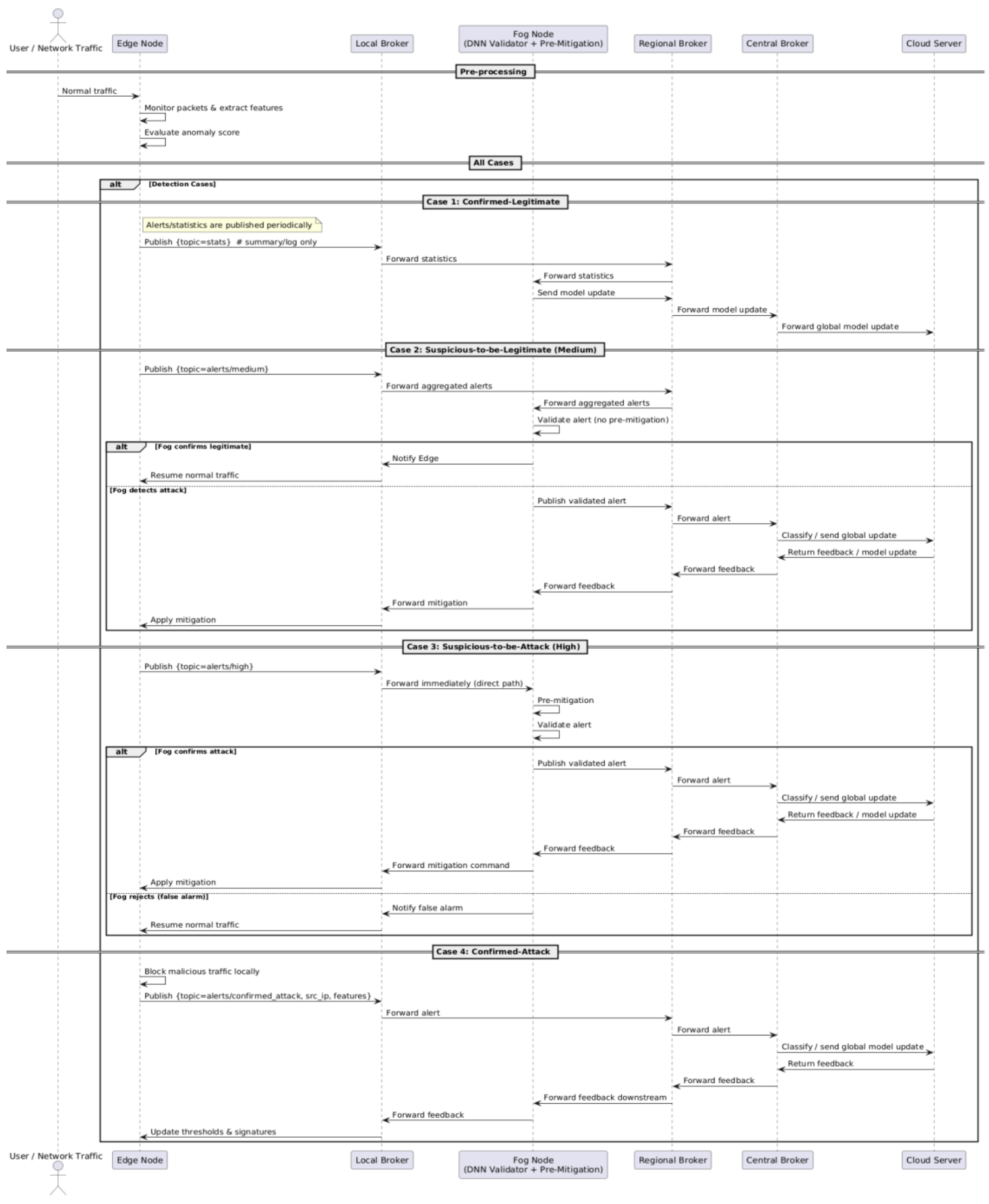
User / Network Traffic
Edge Node
Local Broker
Fog Node (DNN Validator + Pre-Mitigation)
Regional Broker
Central Broker
Cloud Server
Pre-processing
Normal traffic
Monitor packets & extract features
Evaluate anomaly score
All Cases
alt [Detection Cases]
Case 1: Confirmed-Legitimate
Alerts/statistics are published periodically
Publish {topic=stats} # summary/log only
Forward statistics
Forward statistics
Send model update
Forward model update
Forward global model update
Case 2: Suspicious-to-be-Legitimate (Medium)
Publish {topic=alerts/medium}
Forward aggregated alerts
Forward aggregated alerts
Validate alert (no pre-mitigation)
alt [Fog confirms legitimate]
Notify Edge
Resume normal traffic
[Fog detects attack]
Publish validated alert
Forward alert
Classify / send global update
Return feedback / model update
Forward feedback
Forward feedback
Forward mitigation
Apply mitigation
Case 3: Suspicious-to-be-Attack (High)
Publish {topic=alerts/high}
Forward immediately (direct path)
Pre-mitigation
Validate alert
alt [Fog confirms attack]
Publish validated alert
Forward alert
Classify / send global update
Return feedback / model update
Forward feedback
Forward feedback
Forward mitigation command
Apply mitigation
[Fog rejects (false alarm)]
Notify false alarm
Resume normal traffic
Case 4: Confirmed-Attack
Block malicious traffic locally
Publish {topic=alerts/confirmed_attack, src_ip, features}
Forward alert
Forward alert
Classify / send global model update
Return feedback
Forward feedback
Forward feedback downstream
Forward feedback
Update thresholds & signatures

Fig. 1. Sequence diagram illustrating inter-layer communication and dynamic behavior of the proposed system

To guarantee reliable communication between our proposed three-layer architecture, we define three different kinds of brokers:

- Local brokers: They are deployed within the fog domain but positioned close to edge nodes. Each local broker manages a small group of edge nodes and subscribes to their alert topics (edge/alert, edge/status). These brokers aggregate incoming messages and forward them selectively to the regional broker for higher-level coordination, or to the fog nodes for deep validation.

- Regional brokers: They are also located within the fog layer and subscribe to multiple local brokers to coordinate intra-fog operations. There are several types of messages exchanged, such as validated alert confirmations, local mitigation actions, and model update notifications (fog/validation, fog/mitigation, and fog/model-update). This hierarchical structure reduces unnecessary upstream traffic by processing most communications within the regional layer.
- The central cloud broker: This is the main coordinator that subscribes to aggregated topics from regional brokers. It receives validated attack detections, global alerts, and mitigation feedback (cloud/alert, cloud/detection, and cloud/feedback). The central broker can also publish global policies or model updates to regional brokers, which are then propagated downward.

This hierarchical communication design enables rapid alert propagation, coordinated mitigation across layers, and sustained operation even under attack conditions. Each MQTT broker acts as a middleware interface that connects the detection models across layers. Fig.1. illustrates the sequence of interactions, demonstrating how the system dynamically manages different traffic states.

3.1. Edge layer: threshold-guided filtering and early mitigation

The lightweight threshold-guided machine learning model is deployed at the edge layer, typically on network devices such as routers or gateways, where it can distinguish between suspicious, malicious, and benign traffic. The suspicious category is further divided into “suspicious-to-be-legitimate” and “suspicious-to-be-an-attack”. If the traffic is confirmed benign, it is forwarded normally. In this case, the edge does not alert the fog to avoid unnecessary communication; however, it can periodically send summary messages to report general traffic statistics. If the traffic is confirmed malicious, the edge immediately blocks or drops the packets and alerts the cloud layer via an MQTT message that includes relevant information (e.g., the attack signature, source IP address, and detection confidence level). For traffic identified as suspicious-to-be-legitimate or suspicious-to-be-an-attack, the edge nodes alert the fog layer through an MQTT message containing some relevant metadata. The traffic itself is then forwarded to the fog for deeper validation. The routing of these alerts depends on the detection confidence level: alerts labeled as confirmed benign, confirmed malicious, or suspicious-to-be-legitimate are sent to the local broker, which forwards them to the regional broker for further coordination and monitoring, while alerts classified as suspicious-to-be-attack are sent directly from the local broker to the fog nodes for rapid validation and pre-mitigation. This selective mechanism can prevent overloading the fog with minor alerts, limit unnecessary upstream communication, and ensure that critical traffic is analyzed promptly. As a result, our system maintains low latency in attack validation and mitigation, high responsiveness at the edge, and effective resource utilization across layers.

3.2. Fog layer: neural validation and regional mitigation

The fog layer validates traffic alerts received from the edge using a DNN model deployed on fog nodes such as proxy servers or local data centers. Local brokers handle communication with nearby edge nodes, while inter-fog coordination occurs through regional brokers. The fog layer performs a detailed validation when it receives an alert message labeled as suspicious-to-be-legitimate. If the traffic is confirmed legitimate, the fog notifies the edge nodes through MQTT feedback. Conversely, if the validation confirms that the traffic is malicious, the fog immediately triggers local mitigation measures such as rate limiting or traffic isolation and publishes the result to the cloud for global awareness and coordinated response. For traffic marked as suspicious-to-be-attack, the fog immediately applies temporary pre-mitigation strategies such as traffic shaping or rerouting while performing deeper validation. If the validation confirms that the traffic is benign, the fog promptly cancels the pre-mitigation measures and sends a feedback message to the edge via MQTT to update its local classification model. Conversely, if the validation confirms an attack, the fog reinforces local mitigation mechanisms and publishes the result to the cloud for coordinated mitigation.

3.3. Cloud layer: attack type classification and strategic mitigation

The third model operates at the cloud layer, where it is deployed on cloud proxy servers to perform attack-type classification and coordinate global mitigation strategies. At the same time, a central MQTT broker coordinates message flows across regional brokers. Once the cloud layer receives an alert message indicating that an attack has been validated, it classifies it into exploitation-based or reflection-based categories. Exploitation attacks trigger actions such as dynamic virtual machine migration, service isolation, or the activation of application-level security policies. In contrast, reflection-based attacks may trigger upstream filtering, BGP rerouting, or collaboration with Intrusion Prevention Systems (IPS) to block traffic closer to the source.

3.4. Proposed hierarchical architecture over centralized solutions

Compared to a centralized detection solution, the proposed hierarchical MQTT-based architecture offers significant advantages. In a centralized solution, all traffic and alerts must reach the cloud for analysis and mitigation, which increases latency, overloads communication channels, and exposes the system to a single point of failure. By contrast, our design performs early detection and filtering directly at the edge. Confirmed benign and confirmed malicious cases are handled locally without involving the fog or cloud layers, which reduces end-to-end latency and upstream traffic. Only suspicious cases are forwarded to the fog layer for validation and mitigation. This coordination ensures scalability through distributed processing, minimizes cloud load, and strengthens security by detecting and mitigating attacks before they reach critical cloud resources. Although our proposed architecture may generate additional inter-layer messaging, previous studies such as that of Kurdi and Thayananthan [10] have demonstrated that multi-tier MQTT networks provide lightweight, asynchronous, and reliable communication with quality of service guarantees.

4. Layer-wise detection evaluation

In this section, we present the detection techniques and evaluation results for each layer of the proposed architecture. Each layer integrates a model tailored to its specific role and constrained by its available computational resources. The CICDDoS2019 dataset was used to train and validate all proposed models. The objective is to demonstrate how the layered approach enhances detection accuracy, reduces false positives, and enables context-aware mitigation while maintaining computational efficiency.

4.1. Dataset

There are many DDoS attack datasets in the literature; the most popular are NSL-KDD [12] and KDD Cup 1999 [13]. Although these datasets are still widely used, they are considered outdated. In addition to these datasets, more recent DDoS datasets, such as CICIDS2017 [14] and CICDDoS2019 [1], have been made available. We selected the CICDDoS2019 for this study because it contains recent and diverse DDoS attack scenarios, focuses exclusively on DDoS attacks, and provides distinct training and testing captures collected on different days, enabling a more reliable evaluation of the model's generalization. The training dataset includes UDP, UDP-lag, SYN, and WebDDoS attacks as exploitation types, and DNS, LDAP, NetBIOS, SNMP, TFTP, SSDP, MSSQL, and NTP attacks as reflection types. The testing dataset contains UDP, UDP-lag, and SYN flood as exploitation attacks, and LDAP, NetBIOS, MSSQL, and Portmap as reflection attacks.

4.2. Metrics

Performance measures play a crucial role in evaluating the quality of learning methods and trained models [15]. Accuracy, Precision, Recall, and F1-score are the most commonly used measures. The Confusion Matrix presents several metrics, including True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) [16]. The associated metrics are delineated as follows [16]:

Accuracy. The proportion of instances in a given data set that are correctly classified. This metric is only valid when the classes are highly balanced,

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}. \tag{1}$$

Precision. The proportion of correct classifications from a set of predictions classified as attacks,

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}. \tag{2}$$

Recall. The ratio of accurately classified attack instances within a specified set of attacks. This measurement signifies the model's capability to detect attacks,

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}. \tag{3}$$

F1-score. The harmonic means of precision and recall. For class-imbalanced problems, this is a valuable metric for model evaluation,

$$\text{F1score} = 2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision+Recall}}. \tag{4}$$

4.3. Edge layer: Lightweight detection and early mitigation

In this section, we present our threshold-guided detection method, where thresholds are derived from a machine learning model rather than being statically defined. Traditional threshold-based DDoS detection methods often suffer from high false positive or false negative rates when thresholds are poorly calibrated [17, 18]. By leveraging machine learning, adaptive thresholds can be learned from traffic patterns to balance the trade-off between normal and malicious behaviors. However, such thresholds may not generalize across all attack types and must be dynamically updated to reflect evolving traffic conditions. In the proposed approach, thresholding is applied at the initial stage to identify confirmed cases only, while suspicious cases are escalated to the fog layer for deeper analysis using a DNN model. This design improves the efficiency and accuracy of the overall system. To accomplish this purpose, we need to extract three thresholds: a minimum ThresHold (THmin), a maximum ThresHold (THmax), and a median ThresHold (THmed). Based on these thresholds, it is possible to classify incoming traffic into four categories: confirmed legitimate traffic, confirmed attack, traffic suspected to be an attack, and traffic suspected to be legitimate. The proposed approach is evaluated using the CICDDoS2019 dataset.

4.3.1. Preprocessing

Due to the large size of the CICDDoS2019 dataset, several modifications were applied to the original dataset. CICDDoS2019 is subdivided into two folders: training and testing. We randomly extracted data from each folder (benign and attack) and concatenated all CSV files to construct a training and testing set. The CICDDoS2019 contains more than eighty features, but not all of these features are used for our evaluation. According to [1], 24 features are identified as the most relevant. Through the experiment, we observed that these features were sufficient to achieve optimal performance in terms of validation accuracy and F1-score. To prepare the data, the following preprocessing steps were performed:

- The original dataset contains many missing values (infinity and NaN), which were removed. Redundant samples were also eliminated.
- In this approach, we adopt a binary classification, where we need to classify incoming traffic as either benign or malicious. The string values "benign" and "attack type" were encoded as 0 and 1, respectively.
- The data was normalized using a standard scaler.

Finally, the data was partitioned into three distinct subsets: training, validation, and testing. The validation set was used to evaluate the model's performance and accuracy. The testing set was extracted from the original testing folder and was not involved in any stage of training or hyperparameter tuning, ensuring an unbiased and realistic evaluation.

4.3.2. Training model

Our experiments showed that several traditional machine learning algorithms, such as Support Vector Machines (SVM), Random Forests (RF), Naive Bayes, Logistic Regression (LR), and Decision Trees (DT), produced results comparable to those reported in previous studies using the CICDDoS2019 dataset, such as [19] and [20]. To enhance accuracy and performance, we employed the Stochastic Gradient Descent (SGD) classifier, which combines the advantages of stochastic updates with the classification strength of an SVM [21]. This algorithm is fast, scalable, and memory-efficient, and it performed better than other traditional machine learning algorithms in this study. To achieve optimal performance, the model's hyperparameters were tuned using the RandomizedSearchCV method. This cross-validation-based hyperparameter optimization method randomly samples parameter combinations from predefined ranges, as described by the authors in [22]. The parameters considered included the loss, penalty, alpha, learning rate, class weight, and eta_0, which were evaluated using stratified 5-fold cross-validation. The F1-score was used as the optimization metric. The best-performing configuration was loss = "hinge", penalty = "elasticnet", alpha = 0.0001, learning_rate = "invscaling", class_weight = {1:0.4, 0:0.6}, and $eta_0 = 1$. This configuration achieved higher validation accuracy and F1-score compared to the initial baseline model and outperformed the results reported in previous studies. Therefore, these parameters were selected as the final configuration for our model. Following this optimization, threshold values were extracted for detection.

4.3.3. Thresholds extraction

The threshold extraction method is based on interpreting the precision–recall trade-off of the trained model. The precision-recall curve was generated using the validation set to assess the classifier's performance at various decision thresholds, as shown in Fig. 2. This was achieved using the precision_recall_curve function, which expects as input the continuous confidence scores produced by the model's decision_function. These scores represent the distance of each sample from the decision boundary, which allows flexible threshold tuning. The function outputs three vectors (recall, precision, and thresholds) that describe how the classifier performs at a given threshold.

The precision vector represents the ratio of detected attacks to all predicted attacks; high precision corresponds to a low false-positive rate. Recall is the rate of actual attacks detected correctly; a high recall means there are relatively few false negatives. Thresholds are the cut-off values of decision scores that distinguish the benign class from the attack class by balancing precision and recall according to the objective of the system. Using these three vectors, we plotted the precision-recall curve and computed the F1-score at each threshold. Since the F1-score is the harmonic mean of precision and recall, it provides a more informative measure than accuracy in the presence of imbalanced datasets [23], as in our case. Therefore, the F1-score is particularly useful for evaluating the proposed model. Based on this analysis, three representative thresholds were extracted. These thresholds were obtained by applying the next three equations, where the operator argmax denotes the index of the maximum value in each vector:

(5) $\mathrm{TH}_{\min} = \mathrm{threshold}[\arg\max(\mathrm{recall})]$ ($\mathrm{TH}_{\min}$, which maximizes recall),

(6) $\mathrm{TH}_{\mathrm{med}} = \mathrm{threshold}[\arg\max(f1\ \mathrm{score})]$ ($\mathrm{TH}_{\mathrm{med}}$, which maximizes f1 score),

(7) $\mathrm{TH}_{\max} = \mathrm{threshold}[\arg\max(\mathrm{precision})]$ ($\mathrm{TH}_{\max}$, which maximizes precision).

4.3.4. Threshold-guided classification mechanism

Once the binary detection model had been trained and validated, its output was tested on an independent test dataset that had not been used during training or validation. First, the model was evaluated using the default decision threshold (0.5) via the predict method, which returns discrete class labels (0 or 1). Then the proposed threshold-guided algorithm employed the decision_function method, which returns continuous decision scores rather than binary labels. The extracted thresholds were applied within this process to classify traffic into four confidence-based classes, as shown in Algorithm 1. The standard evaluation metrics (recall, precision, F1-score, and accuracy) were then computed to verify that the proposed mechanism maintains comparable overall performance while achieving a better balance between recall and precision. To further validate the correctness of the boundary thresholds, samples strictly below the minimum and above the maximum thresholds were separately analyzed. The detailed results are presented in Subsection 4.3.5.

```
Algorithm 1: Threshold-Based Algorithm
Inputs:
   - model: pre-trained model
   - $TH_{min}$, $TH_{med}$, $TH_{max}$: predefined threshold values
Outputs:
   - Yres: binary classification vector
Data:
   - Xtest: test instances
   - Ytest: ground-truth labels
Procedure:
Step 1. Compute the decision scores:
       1.1 Prediction ← model.DecisionFunction(Xtest)
Step 2. For each instance i in Prediction do
     Step 2.1. If Prediction[i] < $TH_{min}$ then
           Yres[i] ← 0        // Traffic confidently classified as legitimate
     Step 2.2. Else if Prediction[i] ≥ $TH_{max}$ then
           Yres[i] ← 1        // Traffic confidently classified as malicious
     Step 2.3. Else if Prediction[i] < $TH_{med}$ then
           Yres[i] ← 0        // Traffic likely to be legitimate
     Step 2.4. Else
           Yres[i] ← 1        // Traffic likely to be malicious
Step 3. End For
Step 4. Return Yres
```

4.3.5. Results and discussion

We evaluated our model on the independent test set and generated the Receiver Operating Characteristic (ROC) curve to evaluate the model's performance. Fig. 3 illustrates that our model yields an AUC of 99.9%, indicating that it accurately distinguishes between positive and negative classes with a 99.9% separation capability. For a more realistic evaluation, we first evaluate our model using the default threshold and then our proposed threshold-based algorithm on the test dataset. The default threshold for normalized predicted probabilities (ranging from 0 to 1) was set to 0.5. However, as noted in [23], this default value may not always provide the optimal interpretation of predicted probabilities. As shown in Table 2, performance remains generally stable, but the threshold-guided strategy achieved a more balanced trade-off between precision and recall.

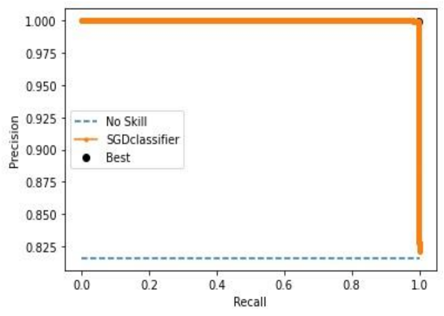

Fig. 2. Recall precision curve for the proposed model

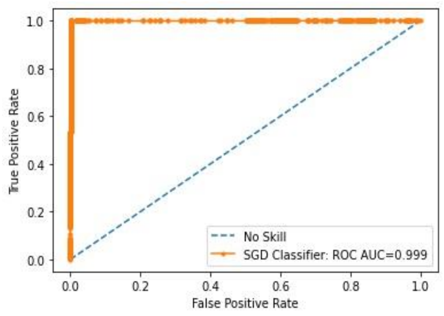

Fig. 3. Receiver Operating Curve (ROC) for the proposed model

The results were compared with the following studies [19, 24, 20, 25]. In [19], DT and RF achieved an accuracy of 99% among six evaluated algorithms. In [24], RF was enhanced using the Arithmetic Optimization Algorithm (AOA) and achieved 99.11% accuracy. In [20], an ANOVA-based feature selection combined with XGBoost achieved 98.34% accuracy. In [25], the proposed approach achieved 99.74% accuracy. Compared with these studies, our proposed approach achieved the highest precision, recall, F1-score, and overall accuracy. Although the method in [25] achieved slightly higher accuracy, its precision, recall, and F1-score were lower than those achieved by our proposed model. This demonstrates that our approach delivers a more balanced and consistent performance across all evaluation metrics. Table 2 presents the detailed comparison.

Table 2. Comparative performance evaluation of the proposed model against other studies

Model	Accuracy	Recall	Precision	F1-score
[19]	0.9900	0.9900	0.9900	0.9900
[24]	0.9911	0.9815	0.9877	0.9846
[20]	0.9835	0.9800	0.9800	0.9800
[25]	0.9974	0.9907	0.9912	0.9911
Our model (Default threshold)	0.9960	0.9987	0.9960	0.9973
Our model (Extracted threshold)	0.9960	0.9986	0.9996	0.9990

Based on our evaluation, we separately tested the minimum threshold. This threshold was applied to classify any instance with a predicted probability below it as benign. Approximately 4 % of the benign samples in the test set fell into this category and were correctly identified as benign, with no false negatives. Similarly, we tested the maximum threshold, which was applied to classify any instance with a predicted probability above it as an attack. This rule also achieved 99.99 % accuracy, and about 9 % of the attack samples in the test set were classified in this manner, with no false positives.

4.4. Fog layer: Neural validation and regional defense

This section details the architecture of our proposed model based on DNNs, which is a class of Artificial Neural Networks (ANNs) distinguished from standard neural networks by the use of multiple hidden layers. The term deep specifically refers to the presence of multiple hidden layers in the architecture of the network. DNNs have gained popularity in both supervised and unsupervised learning tasks [26]. The proposed architecture consists of an input layer whose number of units corresponds to the feature dimension, followed by three hidden layers, each comprising 64 units. From our experimental analysis, we observed that these three hidden layers were sufficient to achieve optimal performance. The Rectified Linear Unit (ReLU) activation function is applied to the input and hidden layers, while the output layer uses a sigmoid activation. The architecture of the proposed model is illustrated in Fig. 4. We evaluate our model using the CICDDoS2019 dataset described in the previous section.

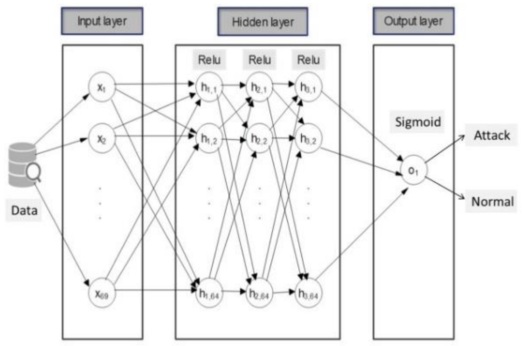

Fig. 4. Architecture of the proposed DDN model

4.4.1. Preprocessing

We prepared the dataset to facilitate the model training process. As the dataset contains a significant amount of data divided into two separate folders (training and testing), we randomly extracted samples from each folder (benign and attack) and merged all CSV files to construct the training and test datasets. The following steps summarize the data preparation process:

- We excluded certain features, such as Source IP, Destination IP, Timestamp, and Flow ID, to enhance the dataset. These features do not contribute to the classification process and may interfere with the model's learning process. We removed records containing infinite or NaN values.

- We removed twelve features that showed zero variance with respect to the target variable. Finally, the model was trained with 69 features.
- Since we aim to adapt our model for binary classification, which classifies incoming traffic as either normal or attack, we encoded the string value "benign" as "0" and "type attack" as "1".
- Classification errors may occur if the model is trained directly on the original data; therefore, we normalized all features using a standard scaler.

4.4.2. Training model

Initially, we divided the dataset (training file) into three subsets: training, validation, and test. The validation set was used for hyperparameter tuning and, in this sense, formed part of the model training process. The test split was employed to evaluate and estimate the model's performance and accuracy. To obtain more realistic results, we further evaluated the model using a test set derived from the separate testing file. This file contained data collected on a different day and was not used in any stage of training. More details about the dataset structure are provided in Subsection 4.1. The ReLU activation function was applied to the input and hidden layers, while the sigmoid activation function was used for the output layer. According to [27], the principal key behind the ReLU activation function is that it yields a value of 0 for an input of 0 or any negative value. In contrast, for positive inputs, it returns the input value itself. Equation (8) in [27] provides the mathematical formulation for the ReLU activation function, where the input vector is denoted by x,

(8) $$\mathrm{relu}(x) = \max(0, x).$$

The sigmoid function can provide probabilistic outputs that seem to work well for inputs ranging from 0 up to 1 [28]. The mathematical specification of the sigmoid activation function, applied to the input vector denoted by x, is in Equation (9) of [28],

(9) $$\mathrm{sigmoid}\ (x) = \frac{1}{1 + e^{-x}}.$$

We use the binary cross-entropy function as a loss function. According to [29], cross-entropy denotes a measure of the difference between two distributions of probability associated with a given random variable or set of events. It optimizes the models to make appropriate decisions in binary predictions. The mathematical formula of the binary cross-entropy function is given by (10) in [29], where y is the true value, and $\hat{y}$ is the predicted outcome,

(10) $$\mathrm{Loss}(y, \hat{y}) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y})).$$

In our approach, we employ the SGD optimizer with Nesterov momentum. In contrast to the normal gradient descent, which only considers current parameters, it looks forward to where the parameters might be after an update. It converges on the given optima much faster [30]. The momentum term improves stability by accounting for prior gradient directions [31] while the Nesterov modification makes refinement by aiding intermediate displacements as defined by (11) and (12) in [32]:

(11) $$v_{t+1} = \mu . v_t - \alpha \nabla f(\theta_t + \mu . v_t),$$

(12) $$\theta_{t+1} = \theta_t + v_{t+1},$$

where t is the timestep, θ_t is the parameter vector at timestep t, v_t is the velocity at timestep t, α is the learning rate, μ is the momentum term, and $\nabla f(\theta_t)$ is the gradient of the objective function with respect to the parameter vector at the current position. We trained the model for 50 epochs with a batch size of 64.

4.4.3. Evaluation and results

The performance of our proposed DNN model is illustrated in Figs 5 and 6. To further evaluate its classification capability, we generated the ROC curve using the independent test set, as shown in Fig. 7. The resulting AUC of 99.9% confirms that the model can distinguish between positive and negative classes with very high reliability. The model was first evaluated using the test split, achieving 99.97% accuracy, about 100% precision, 99.97% recall, and an F1-score of 99.99%. To obtain a more realistic evaluation, an additional test was conducted on an independent dataset collected on a different day, with no overlap with the training or validation sets. In this evaluation, which reflects real-world conditions, the model achieved an accuracy of 99.95%, precision of 99.97%, recall of 99.97%, and an F1-score of 99.97%, demonstrating a fine balance between accuracy and robustness.

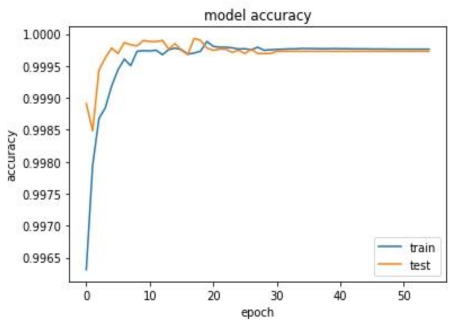

Fig. 5. Accuracy of our proposed model (Test split)

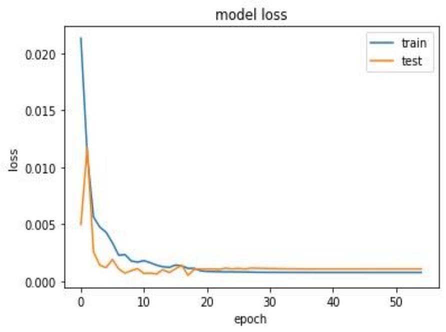

Fig. 6. Loss of our proposed model (Test split)

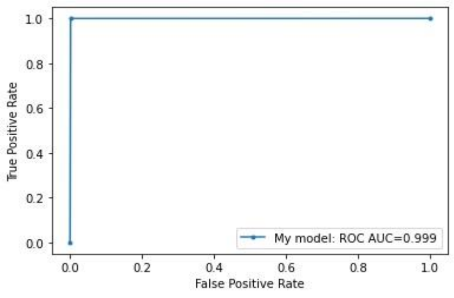

Fig. 7. Receiver Operating Curve (ROC) for the proposed model (Test set)

To demonstrate the effectiveness of our approach regarding performance and computational complexity, we compared it with several recent studies. Notably, the hybrid architecture proposed in [33] achieved a recall of 99.95% and an accuracy of 99.99%. This result was obtained using a hybrid architecture based on CNNs, PCA, and Vision Transformers. However, while producing remarkable results, it tends to have increased complexity due to multimodal integration and feature transformation, making real-time deployment significantly more challenging. Similarly, the study in [34] proposed a hybrid MLP-CNN model for DDoS detection in SDN, enhanced by a Bayesian optimizer and Shapley Additive feature selection, achieving an accuracy of 99.95%. However, the integration of CNN and MLP layers together with Bayesian optimization makes the model computationally more complex compared to our simpler design. The authors in [35] used a Bayesian-based CNN with data fusion (BaysFusCNN) to achieve an accuracy of 99.79%. However, the Bayesian fusion process increases both implementation and training costs. In contrast, our proposed DNN achieved comparable or superior performance with lower computational complexity, offering a better balance between computational cost and detection performance. The comparative results are summarized in Table 3.

Table 3. Comparative performance evaluation of the proposed model against other studies

Model	Accuracy	Recall	Precision	F1-score
[33] CNN+PCA+ViT	99.99	99.95	99.8	99.85
[34] CNN/MLP+BaysOpt	99.95	99.97	99.90	99.93
[35] BaysFusCNN	99.79	98.75	98.57	98.56
Our DNN(Test split)	99.97	99.97	≈ 100	≈ 99.99
Our DNN(Test set)	99.95	99.97	99.97	99.97

4.5. Cloud layer: attack-type classification and strategic mitigation

In the cloud layer, we adopted the same DNN architecture as in the fog layer to ensure architectural consistency and reduce design complexity. At this level, the only modification concerns the training data, which was adapted for a binary classification task distinguishing between two attack types: exploitation and reflection. This specialization allows the cloud layer to perform fine-grained validation and categorization of traffic previously flagged as malicious by the upstream layers. This classification also enables context-aware mitigation.

4.5.1. Preprocessing

At the cloud level, we use only attack samples, excluding benign traffic, ensuring a balanced distribution between reflection and exploitation attacks. We present the different steps followed to prepare the dataset for model training:

- Following the same preprocessing approach applied to the fog-level dataset, we exclude irrelevant features that do not contribute to the classification process and may hinder model learning. We also remove any records containing infinite or NaN values.
- We remove 12 features that exhibit zero variance with respect to the target variable. After this step, the model was trained using 69 features.
- Since our goal is to adapt the model for binary classification, where incoming traffic is categorized as either exploitation-based or reflection-based, we encode the exploitation-based attacks as “0” and the reflection-based attacks as “1”.
- To reduce potential classification errors caused by differences in feature scales, we normalize the data using a standard scaler**.**

4.5.2. Training model

Initially, the dataset was partitioned into three subsets: training, validation, and test. For a more robust and realistic evaluation of algorithms, an additional evaluation was performed using an independent test set. The same training parameters were applied to this model as those used for the DNN fog model.

4.5.3. Evaluation and results

The evaluation results indicate that our model can classify DDoS attacks into exploitation-based and reflection-based types with an accuracy of 96%, as illustrated in Figs. 8 and 9. Further evaluation on an independent test set showed a detection accuracy of 99%, as illustrated by the ROC curve in Fig. 10.

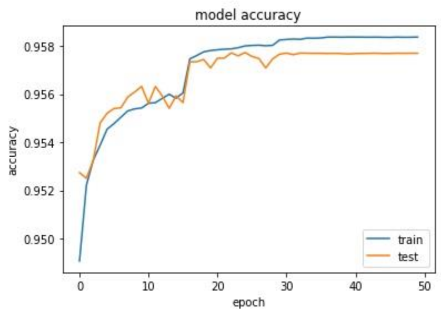

Fig. 8. Accuracy of our proposed model (Test split)

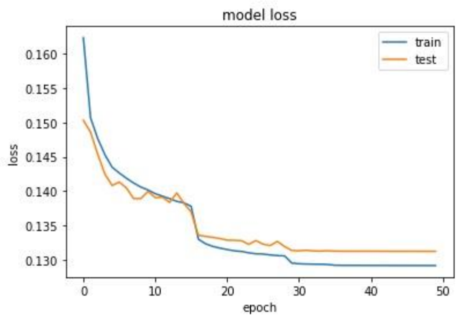

Fig. 9. Loss of our proposed model (Test split)

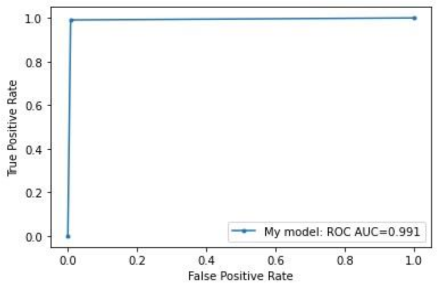

Fig. 10. Receiver Operating Curve (ROC) for the proposed model (Test set)

Notably, the accuracy achieved on the independent test set extracted from the dataset file was higher than that on the test split derived from the training dataset file. This difference is likely due to the limited presence or absence of certain attack types in the independent test set compared to the test split. More details about the dataset are provided in Subsection 4.1. A comparative analysis with the results reported in [36], summarized in Table 4, confirms that our proposed model outperforms previous work in this domain.

Table 4. Comparative performance evaluation of the proposed model against other studies

Model	Accuracy	Recall	Precision	F1-score
[36] DNN	0.9457	0.9515	0.8049	0.9604
Our DNN(Test set)	0.9913	0.9906	0.9923	0.9915
Our DNN(Test split)	0.9580	0.9367	0.9853	0.9604

5. Conclusion and future works

A hierarchical Edge-Fog-Cloud DDoS detection and mitigation architecture is proposed in this paper. The framework combines lightweight threshold-guided filtering at the edge, deep neural validation at the fog, and attack-type classification at the cloud. The layers cooperate to ensure reliable, real-time detection and adaptive mitigation. That is achieved through an MQTT-based communication workflow. Experimental results on the CICDDoS2019 dataset confirm the effectiveness of our approach. It achieves nearly 100% accuracy in distinguishing confirmed benign and attack traffic. It identifies suspected attacks and suspected benign cases with 99.6% accuracy at the edge. The fog layer employs a DNN to validate suspicious traffic, reaching 99.97% accuracy and nearly 100% precision, while the cloud layer classifies DDoS attacks with about 96% accuracy. The architecture achieves an effective balance between accuracy, responsiveness, and computational efficiency while reducing false positives and upstream load. Future work will focus on fine-grained attack classification and the simulation of large-scale deployments using containerized microservices to evaluate the scalability, responsiveness, and fault tolerance of the proposed architecture under real-world conditions.

References

1. Sharafaldin, I., A. H. Lashkari, S. Hakak, A. A. Ghorbani. Developing a Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. – In: Proc. of 53rd International Carnahan Conference on Security Technology, Chennai, India, 2019, IEEE, 2019.
2. AlSaleh, I., A. Al-Samawi, L. Nissirat. Novel Machine Learning Approach for DDoS Cloud Detection: Bayesian-Based CNN and Data Fusion Enhancements. – Sensors, Vol. **24**, 2024, No 5, 1418.
3. Ec-Sabery, M., A. B. Abbou, A. Boushaba, F. Mrabti, R. B. Abbou. The Optimized Extreme Learning Machine (GA-OELM) for DDoS Attack Detection in a Cloud Environment. – Journal of Computer Science, Vol. **21**, 2025, No 1, pp. 146-157.
4. Songa, A. V., G. R. Karri. An Integrated SDN Framework for Early Detection of DDoS Attacks in Cloud Computing. – Journal of Cloud Computing, Vol. **13**, 2024, 64.
5. Amitha, M., M. Srivenkatesh. DDoS Attack Detection in Cloud Computing Using Deep Learning Algorithms. – International Journal of Intelligent Systems and Applications in Engineering, Vol. **11**, 2023, No 4, pp. 82-90.
6. Gudla, S. P. K., S. K. Bhoi, S. R. Nayak, A. Verma. DI-ADS: A Deep Intelligent Distributed Denial of Service Attack Detection Scheme for Fog-Based IoT Applications. – Computational Intelligence and Neuroscience, Vol. **2022**, 2022, pp. 1-17.
7. Bensaid, R., N. Labraoui, A. A. A. Ari, L. Maglaras, H. Saidi, A. M. A. Lwahhab, S. Benfriha. Toward a Real-Time TCP SYN Flood DDoS Mitigation Using an Adaptive Neuro-Fuzzy Classifier and SDN Assistance in Fog Computing. – Security and Communication Networks, Vol. **2023**, 2023, 6651584.

8. P o k a l e, N. B., P. S h a r m a, D. T. M a n e. Deep Hybrid Model for Attack Detection in IoT-Fog Architecture with Improved Feature Set and Optimal Training. – In: Web Intelligence, 2024.
9. S e l i m, I. M., R. A. S a d e k. An Effective Fog-Aware NIDS for DDoS Attack Detection. – IAENG International Journal of Computer Science, Vol. **52**, 2025, No 6, pp. 1806-1814.
10. K u r d i, H., V. T h a y a n a n t h a n. A Multi-Tier MQTT Architecture with Multiple Brokers Based on Fog Computing for Securing Industrial IoT. – Applied Sciences, Vol. **12**, 2022, No 7173, pp. 1-15.
11. N a i k, N. Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP, and HTTP. – In: IEEE International Systems Engineering Symposium (ISSE'17), Vol. **1**, 2017, No 7, pp. 1-7.
12. T a v a l l a e e, M., E. B a g h e r i, W. L u, A. G h o r b a n i. A Detailed Analysis of the KDD Cup 99 Data Set. – In: Proc. of 2nd IEEE Symposium on Computational Intelligence for Security and Defense Applications. IEEE, 2009.
13. D i v e k a r, A., M. P a r e k h, V. S a v l a, M. S h i r o l e. Benchmarking Datasets for Anomaly-Based Network Intrusion Detection: KDD Cup 99 Alternatives. – In: Proc. of IEEE International Conference on Computing, Communication and Security, Kathmandu, Nepal, 2018, IEEE, 2018.
14. S h a r a f a l d i n, I., A. H. L a s h k a r i, S. H a k a k, A. A. G h o r b a n i. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. – In: Proc. of 4th International Conference on Information Systems Security and Privacy (ICISSP'18), Portugal, 2018, Scitepress, 2018.
15. F e r r i, C., J. H e r n a n d e z-O r a l l o, R. M o d r o i u. An Experimental Comparison of Performance Measures for Classification. – Pattern Recognition Letters, Vol. **30**, 2009, No 1, pp. 27-38.
16. G a m a g e, S., J. S a m a r a b a n d u. Deep Learning Methods in Network Intrusion Detection: A Survey and an Objective Comparison. – Pattern Recognition Letters, Vol. **169**, 2020, No 2.
17. G u b t a, B. B., M. M i s r a, R. C. J o s h i. An ISP-Level Solution to Combat DDoS Attacks Using a Combined Statistical-Based Approach. – International Journal of Information Assurance and Security, Vol. **3**, 2008, No 2, pp. 102-110.
18. J i s a, D., T. S i z a. Efficient DDoS Flood Attack Detection Using Dynamic Thresholding on Flow-Based Network Traffic. – Computers and Security, Vol. **82**, 2019, pp. 284-295.
19. A z a h r a n i, R. J., A. A z a h r a n i. Security Analysis of DDoS Attacks Using Machine Learning Algorithms in Network Traffic. – Electronics, Vol. **10**, 2021, No 23.
20. G a u r, V., R. K u m a r. Analysis of Machine Learning Classifiers for Early Detection of DDoS Attacks on IoT Devices. – Arabian Journal for Science and Engineering, Vol. **47**, 2022, pp. 1353-1374.
21. W i j n h o v e n, R. G. J., P. H. N. de W i t h. Fast Training of Object Detection Using Stochastic Gradient Descent. – In: Proc. of 20th International Conference on Pattern Recognition, Istanbul, Turkey, 2010, IEEE, 2010.
22. P r o b s t, P., M. N. W r i g h t, A.-L. B o u l e s t e i x. Hyperparameters and Tuning Strategies for Random Forest. – WIREs Data Mining and Knowledge Discovery, Vol. **9**, 2019, No 3.
23. Z o u, Q., S. X i e, Z. L i n, M. W u, Y. J u. Finding the Best Classification Threshold in an Imbalanced Classification. – Big Data Research, Vol. **5**, 2016, pp. 2-8.
24. J i m i n, L., Z. J i a n y e, Z. H u i q i, D. X u e y u, G. X i n. An Improved Random Forest Intrusion Detection Model Based on Tent Mapping. – In: Proc. of 4th World Symposium on Artificial Intelligence, Jilin, China, 2022. IEEE, 2022.
25. B a r o n a, R., E. B a b u r a j. An Efficient DDoS Attack Detection and Categorization Using an Adolescent Identity Search-Based Weighted SVM Model. – Peer-to-Peer Networking and Applications, Vol. **16**, 2023, No 2, pp. 1227-1241.
26. F a k e r, O., E. D o g d u. Intrusion Detection Using Big Data and Deep Learning Techniques. – In: Proc. of 2019 ACM SouthEast Conference (ACM SE'19), ACM, April 2019.
27. B a n e r j e e, C., T. M u k h e r j e e, E. P a s i l i a o. An Empirical Study on Generalizations of the ReLU Activation Function. – In: Proc. of 2019 ACM SouthEast Conference (ACM SE'19), ACM, April 2019.

28. A k h i l e s h, A. W., K. S. B r i j e s h. Performance Analysis of Sigmoid and ReLU Activation Functions in a Deep Neural Network. – In: A. Sheth, A. Sinhal, A. Shrivastava, A. K. Pandey, Eds. Intelligent Systems, Algorithms for Intelligent Systems, Springer, 2021, pp. 39-52.
29. J a d o n, S. A Survey of Loss Functions for Semantic Segmentation. – In: Proc. of IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'20), Via del Mar, Chile, 2020, IEEE, 2020.
30. Z a h e e r, R., S. H u m e r a. A Study of the Optimization Algorithms in Deep Learning. – In: Proc. of 3rd International Conference on Inventive Systems and Control (ICISC'19), Coimbatore, India, 2019, IEEE, 2019.
31. Q i a n, N. On the Momentum Term in Gradient Descent Learning Algorithms. – Neural Networks, Vol. **12**, 1999, No 1, pp. 145-151.
32. S u t s k e v e r, I., J. M a r t e n s, G. D a h l, G. H i n t o n. On the Importance of Initialization and Momentum in Deep Learning. – In: Proc. of 30th International Conference on International Conference on Machine Learning (ICML'13), ACM, 2013.
33. S h a i k h, J., T. A. S y e d, S. A. S h a h, S. J a n, Q. U l A i n, P. K. S i n g h. Advancing DDoS Attack Detection with Hybrid Deep Learning: Integrating Convolutional Neural Networks, PCA, and Vision Transformers. – International Journal on Smart Sensing and Intelligent Systems, Vol. **17**, 2024, No 1, pp. 1-16.
34. M e h m o o d, S., R. A m i n, J. M u s t a f a, M. H u s s a i n, F. S. A l s u b a e i, M. D. Z a k a r i a. Distributed Denial of Services (DDoS) Attack Detection in SDN Using Optimizer-Equipped CNN-MLP. – PLOS ONE, Vol. **20**, 2025, No 1.
35. A l S a l e h, I., A. A l-S a m a w i, L. N i s s i r a t. Novel Machine-Learning Approach for DDoS Cloud Detection: Bayesian-Based CNN and Data Fusion Enhancements. – Sensors, Vol. **24**, 2024, No 5, 1418.
36. A b d u l l a h, E., K. Y i l d i z. Detection of DDoS Attacks with Feed-Forward Based Deep Neural Network Model. – Expert Systems with Applications, Vol. **486**, 2021, No 3, pp. 75-174.

Fast-track. Received: 16.09.2025, First revision: 20.10.2025,
Second Revision: 5.11.2025, Accepted: 6.11.2025