



Traffic Anomaly Detection in Fog-IoT Networks Using a Combination of Improved Reinforcement Learning and Convolutional Neural Networks

Ahmed Mhmood Abbood AlTameemi^{1*}, Sima Emadi², Hayder Ghanim Murad³, Mohammadreza SoltanAghaie¹

¹ Department of Computer Engineering, Isf.C., Islamic Azad University, Isfahan, Iran

² Department of Computer Engineering, Ya.C., Islamic Azad University, Yazd, Iran,

³ Department of Computer Engineering, University of Wasit, Wasit, Iraq

* Corresponding author email address: ah113205@gmail.com

Received: 2026-04-07

Revised: 2026-09-20

Accepted: 2026-09-27

Initial Publish: 2026-09-28

Final Publish: 2027-04-01

Abstract

Despite significant advances in machine- and deep-learning-based intrusion detection systems, existing methods still face fundamental challenges. Many of these methods either rely on static and predefined patterns and have limited efficiency in detecting unknown and low-rate attacks, or are not suitable for practical deployment in fog and IoT environments due to their high computational complexity. In addition, the issues of data imbalance, traffic pattern dynamics, and limited computational resources in fog nodes mean that a simple combination of multiple learning models alone does not meet the practical needs of this area. In this research, a novel framework for anomaly detection in Fog-IoT networks is presented, in which a convolutional neural network extracts deep representations, and the Dueling-Q reinforcement learning algorithm learns the optimal decision-making policy. Also, to adapt the IDS monitoring data to the reinforcement learning framework, a data transformation module is designed to map traffic flows from a static feature space to structured state representations for the learning agent. To increase detection accuracy, the data imbalance is addressed using SMOTE, and the stability of the learning process is improved by designing a large Replay Buffer. Also, the model reward function is designed to be highly sensitive to false-negative errors so that low-rate, hidden, and distributed attacks are detected more accurately. The evaluation of the proposed model was carried out on two benchmark datasets, UNSW-NB15 and CICIDS2017, under a fog-layer deployment scenario. The results showed that the proposed model achieved 98.8% accuracy, 95.6% attack detection accuracy, 92.4% attack recall, 93.9% attack F1 score, and 96.78% false-negative rate on the UNSW-NB15 dataset. Also, on the CICIDS2017 dataset, the model performance improved further, achieving 98.9% accuracy, 95.8% attack detection accuracy, 92.9% attack recall, and 94.3% attack F1 score. These results indicate the model's strong ability to detect complex, sequence-based, low-rate attacks. The study of originality also showed that removing any of the key components of the model, including the CNN, the Dueling structure, reinforcement learning, or the SMOTE technique, leads to significant performance degradation. Comparisons with methods proposed for fog environments, such as Autoencoder-CNN-LSTM, FFL-IDS, Fog-Edge-SVM-FL, and Whale-ELM, also showed that the proposed method is significantly superior across all critical criteria, especially the attack recall criterion, the most important security indicator. Overall, the results of this research demonstrate that the combination of CNN and Dueling-Q can provide a lightweight, accurate, and practical solution for deployment on fog nodes and serve as a reliable IDS in real Fog-IoT environments.

Keywords: Anomaly Detection, Fog Computing, Internet of Things, Reinforcement Learning, Convolutional Neural Network, Dueling-Q Network, Intrusion Detection, Data Imbalance, SMOTE, Network Traffic Analysis.

How to cite this article:

Abbood AlTameemi, A. M., Emadi, S., Ghanim Murad, H., & SoltanAghaie, M. (2027). Traffic Anomaly Detection in Fog-IoT Networks Using a Combination of Improved Reinforcement Learning and Convolutional Neural Networks. Management Strategies and Engineering Sciences, 9(4), 1-26.



1. Introduction

The rapid expansion of the Internet of Things (IoT) has transformed conventional computing environments into highly interconnected ecosystems in which sensors, actuators, mobile devices, industrial equipment, smart appliances, and embedded systems continuously exchange large volumes of data. IoT infrastructures now constitute an essential technological foundation for smart cities, industrial automation, intelligent transportation, healthcare monitoring, energy management, and many other cyber-physical applications. This increasing connectivity, however, has simultaneously expanded the attack surface available to malicious actors. The heterogeneous hardware characteristics of IoT devices, lightweight communication protocols, limited computing resources, distributed deployment, and continuous connectivity create security weaknesses that cannot always be addressed through conventional perimeter-based protection mechanisms. Consequently, IoT networks are increasingly exposed to distributed denial-of-service (DDoS) attacks, probing, botnets, spoofing, malware, unauthorized access, and previously unseen attack patterns. Reviews of IoT intrusion detection have therefore emphasized that the security problem is no longer limited to identifying known signatures but increasingly involves recognizing complex, evolving, and anomalous behaviors under heterogeneous network conditions [1, 2]. Comprehensive surveys of intrusion detection systems similarly show that changing traffic distributions, dataset limitations, feature dimensionality, class imbalance, and the emergence of unknown attacks continue to constrain the generalizability of conventional detection mechanisms [3, 4].

Intrusion detection systems (IDSs) represent an important second line of defense because they analyze system or network activity to identify malicious or suspicious behavior that preventive controls fail to stop. Traditional IDS techniques are generally categorized as signature-based and anomaly-based mechanisms. Signature-based systems can provide reliable detection when attack characteristics have already been defined, but their dependence on known patterns makes them inherently less effective against new or modified attacks. Anomaly-based detection, by contrast, attempts to characterize legitimate network behavior and recognize substantial deviations from expected patterns, providing greater potential for identifying previously unseen threats. Nevertheless, anomaly detection introduces its own challenges, especially increased false alarms, sensitivity to

changing traffic distributions, and difficulties in distinguishing legitimate behavioral variation from malicious activity. The characteristics of the datasets used for IDS development also strongly influence reported performance. For example, the UNSW-NB15 dataset was developed to provide a more comprehensive representation of contemporary normal and malicious traffic for network intrusion research, addressing important shortcomings of earlier benchmark datasets [5]. Similarly, feature analysis of CICIDS2017 has demonstrated that identifying informative traffic characteristics is fundamental to improving anomaly detection while reducing unnecessary dimensionality and computational overhead [6].

Machine learning has become one of the principal approaches for overcoming some limitations of conventional IDSs because classification algorithms can learn discriminative relationships directly from traffic features. Algorithms such as decision trees, support vector machines, random forests, k-nearest neighbors, extreme learning machines, and other supervised classifiers have been extensively evaluated for attack identification. Comparative research has shown that the performance of these algorithms can vary considerably according to attack type, feature representation, dataset composition, and evaluation protocol [7, 8]. Supervised machine-learning classifiers have also demonstrated substantial potential for intrusion detection specifically in IoT environments, although their effectiveness remains dependent on representative labeled observations and their ability to cope with heterogeneous attack classes [9]. Feature-selection-assisted approaches can further reduce redundant inputs and computational complexity. For example, machine-learning architectures incorporating efficient feature selection and multi-class support vector machines have been developed to improve network intrusion detection while maintaining manageable processing requirements [10]. These findings confirm that classical machine learning remains relevant, particularly when computational efficiency is important, but its dependence on manually engineered or predefined features can restrict its ability to characterize nonlinear, high-dimensional, and rapidly evolving attack behavior.

The challenge becomes particularly important when DDoS and related distributed attacks are considered. IoT devices can be compromised in large numbers and coordinated as botnets, producing attack traffic that may overwhelm network resources while remaining difficult to discriminate from legitimate distributed activity. Early research demonstrated that machine-learning methods could

identify DDoS behavior in consumer IoT traffic based on traffic characteristics, supporting the feasibility of data-driven protection for resource-constrained connected devices [11]. Semi-supervised strategies have subsequently been examined as a means of reducing dependence on fully labeled datasets. A clustering-based semi-supervised approach, for example, demonstrated that combining clustering with machine-learning classification could support DDoS attack discrimination while addressing some limitations associated with extensive labeling requirements [12]. Nevertheless, the continuing evolution of IoT attack strategies means that a model trained on a fixed distribution may gradually become less effective when traffic characteristics change, particularly when low-rate attacks deliberately imitate normal network behavior.

Deep learning offers an important extension of machine-learning-based IDSs because deep architectures can automatically learn hierarchical and nonlinear feature representations from high-dimensional traffic observations. Recurrent neural networks (RNNs), long short-term memory (LSTM) models, convolutional neural networks (CNNs), autoencoders, and hybrid architectures have therefore received considerable attention. An RNN-based intrusion detection framework has demonstrated the capacity of recurrent architectures to capture sequential relationships that are difficult to model using static classifiers [13]. Feature-fusion approaches based on recurrent deep learning have also indicated that combining complementary learned representations can strengthen intelligent intrusion detection performance [14]. Likewise, hybrid CNN-LSTM architectures exploit complementary capabilities: CNN layers extract salient local or structural patterns, whereas LSTM layers model temporal dependencies across network observations [15]. More recent hybrid deep-learning approaches have extended this logic to IoT DDoS security by integrating complementary deep representation mechanisms to improve attack recognition in complex network environments [16]. These developments demonstrate the substantial representational capacity of deep models but also highlight the growing complexity of architectures intended for environments in which processing resources may be restricted.

Unsupervised learning provides another important direction because labeled attack samples are inherently incomplete when novel threats continue to emerge. An unsupervised deep-learning approach for early network traffic anomaly detection showed that models can learn underlying traffic structures and identify deviations without

requiring comprehensive attack labeling [17]. Automated machine learning has similarly been investigated to reduce the human effort involved in algorithm selection, configuration, and anomaly-detection model development for IoT environments [18]. Particularly important is the problem of zero-day intrusion detection, in which the system encounters malicious behaviors absent from its training distribution. Adaptable deep-learning approaches have been developed specifically to improve the detection of zero-day attacks and enable IDS models to respond to evolving attack spaces [19]. Despite these advances, unknown-attack detection remains difficult because the decision boundary learned from historical data does not necessarily remain optimal as network behavior evolves. Static classifiers can therefore achieve high aggregate accuracy while still failing to identify comparatively rare but security-critical attack observations.

The distributed nature of IoT infrastructure has also motivated a shift in IDS deployment from exclusively centralized cloud processing toward edge and fog computing. Fog computing places computational resources between end devices and centralized clouds, enabling data to be processed closer to where it is generated. This architecture can reduce communication latency, limit unnecessary transmission to remote cloud servers, improve response speed, and provide localized security decisions. Fog-based DDoS mitigation frameworks have illustrated how moving traffic analysis toward the network edge can support faster responses to IoT attacks [20]. Fog-assisted frameworks integrated with software-defined networking have likewise been proposed to provide persistent anomaly detection and centralized policy control while retaining the advantages of distributed local processing [21]. Fog-based attack detection using deep learning further demonstrates that computational intelligence can be positioned close to IoT endpoints to improve the timeliness of intrusion recognition [22]. More broadly, distributed-to-centralized smart technology management architectures illustrate the importance of coordinating processing resources across device, fog, and centralized layers in large-scale intelligent environments [23].

Although fog computing provides architectural advantages, it also creates additional security and computational constraints. Fog nodes generally possess fewer processing and memory resources than cloud data centers, while they must analyze dynamic traffic with minimal delay. Accordingly, an IDS designed for a fog environment must achieve not only high classification

performance but also computational efficiency, scalability, and rapid response. Optimization has therefore become increasingly important in IDS design. Optimization-assisted deep-learning architectures have been proposed to improve network intrusion detection by tuning model parameters and enhancing feature processing. The Dugat-LSTM framework, for example, integrates deep sequential learning with a chaotic optimization strategy to improve network intrusion detection performance [24]. Similarly, deep-learning-assisted IDS design based on the Golden Jackal Optimization algorithm illustrates how metaheuristic optimization can be used to improve network security models [25]. In fog environments specifically, optimization-based IDS methods have attempted to balance detection quality and computational requirements; an automatic-termination whale optimization approach combined with an extreme learning machine provides one example of this effort [26]. Such approaches improve parameter or feature optimization, but optimization itself does not necessarily provide a mechanism through which an IDS can continuously learn an adaptive decision policy.

Distributed and collaborative learning has recently emerged as another strategy for securing resource-constrained industrial IoT systems. Federated learning enables multiple nodes to contribute to model development without centrally transmitting all raw observations, potentially improving privacy and distributed scalability. A fog-enabled federated-learning IDS has been proposed to counter jamming and spoofing threats in industrial IoT environments, illustrating the increasing convergence of fog architecture and decentralized learning [27]. Fog-edge intrusion detection has also been investigated for smart-grid environments, where rapid local security analysis is critical because attacks may affect cyber-physical operations and critical infrastructure [28]. Ensemble learning combined with advanced feature selection provides a further mechanism for strengthening intrusion detection in IoT and fog environments by integrating complementary classifiers and reducing irrelevant information [29]. Collectively, these developments demonstrate a trend toward distributed, optimized, and collaborative IDS architectures; nevertheless, many still conceptualize intrusion detection fundamentally as a classification problem in which a trained mapping assigns each observation to a predetermined class.

Scalability introduces an additional challenge in industrial IoT (IIoT) environments because security systems must analyze data generated simultaneously by numerous distributed devices. Multi-agent learning represents a

promising solution by distributing analytical responsibilities while permitting coordinated processing. The MAS-LSTM framework integrates multi-agent organization with LSTM-based anomaly detection to improve scalability and capture temporal dependencies across IIoT traffic [30]. Such approaches are particularly relevant when traffic generation is decentralized and a monolithic detection model creates communication or computational bottlenecks. However, multi-agent and sequence-based models may still require substantial computational resources and, depending on their learning paradigm, can remain dependent on historical labels. Thus, although scalability, feature extraction, and temporal modeling have individually improved, the need remains for a mechanism capable of adapting its security decisions dynamically according to observed states and the consequences of previous decisions.

Deep reinforcement learning provides a conceptually different framework for addressing this requirement. Rather than treating intrusion detection exclusively as a static mapping from input features to labels, reinforcement learning formulates the process as sequential decision making in which an agent observes the environment, chooses an action, receives feedback, and progressively learns a policy that maximizes cumulative reward. This paradigm is potentially valuable for IoT and fog environments because traffic conditions and attack behaviors continually change. A security-oriented reward mechanism can also assign different consequences to different forms of classification error. This is important because the costs of intrusion-detection errors are asymmetric: a false positive may waste resources or generate unnecessary alarms, whereas a false negative may allow a genuine attack to proceed undetected. Therefore, an adaptive IDS should not merely maximize overall accuracy but should explicitly learn to reduce security-critical errors. Existing research has demonstrated strong results for deep learning, feature selection, ensemble learning, optimization, fog deployment, federated learning, automated learning, and scalable multi-agent architectures, yet these strands remain only partially integrated [24, 29, 30].

A particularly promising strategy is to integrate convolutional feature extraction with reinforcement-learning-based decision making. CNNs can transform multidimensional network observations into compact high-level representations by identifying local dependencies among traffic characteristics, while a Deep Q-Network can use those representations as states for policy learning. A Dueling-Q architecture can further separate estimation of the

overall value of a network state from the relative advantage associated with alternative actions, potentially producing more stable decisions when differences among actions are subtle. This combination is particularly relevant to imbalanced traffic data, because the learning objective can incorporate asymmetric penalties that explicitly discourage false negatives. Data imbalance itself remains a fundamental IDS problem because majority normal observations can dominate the learning process, allowing apparently strong overall accuracy to coexist with poor attack recall. The continued emphasis in the intrusion-detection literature on feature selection, model evaluation, imbalance, generalization, and appropriate performance measures demonstrates why security systems must be assessed beyond aggregate accuracy alone [3, 4]. Accordingly, preprocessing and balancing strategies must be integrated with, rather than treated independently from, the decision architecture.

The preceding literature therefore reveals a progressive evolution from conventional classification to deep representation learning and from centralized detection toward fog-, edge-, federated-, and multi-agent architectures. Classical classifiers provide computational simplicity but are restricted in extracting complex features; deep networks improve representation learning but commonly operate as static classifiers; unsupervised and zero-day approaches enhance sensitivity to unfamiliar behavior but may face difficulties in decision calibration; optimization techniques improve model configuration but do not inherently introduce adaptive policy learning; and fog-based systems reduce latency while imposing stricter resource constraints. Consequently, a significant research need exists for a coherent Fog-IoT intrusion detection framework that simultaneously provides deep feature extraction, adaptive decision making, explicit sensitivity to false-negative errors, robustness to class imbalance, and computational suitability for fog-layer deployment. Addressing these dimensions within one integrated architecture can provide a more realistic response to modern IoT security conditions than simply combining multiple classifiers or maximizing overall detection accuracy.

Accordingly, the aim of this study is to develop and evaluate an adaptive traffic-anomaly detection framework for Fog-IoT networks that integrates convolutional neural network-based feature extraction with Dueling-Q deep reinforcement learning, data transformation, class-imbalance handling, and false-negative-sensitive policy learning to improve accurate, robust, and computationally efficient detection of normal and malicious network traffic.

2. Methodology

The proposed method in this study is an integrated, problem-oriented learning framework for anomaly detection in Fog-IoT environments, in which the intrusion detection problem is redefined from a static classification task to an adaptive decision-making process based on reinforcement learning. In this framework, rather than a simple combination of models, a purposeful interaction among feature extraction, data transformation, and decision policy is designed to simultaneously address challenges such as traffic dynamics, data imbalance, and sensitivity to critical errors. Initially, the raw data, including traffic flows from the UNSW-NB15 and CICIDS2017 datasets, are transformed into 2D feature maps after preprocessing steps, including removing irrelevant features, fixing missing values, Z-score normalization, and addressing class imbalance using SMOTE. In this step, a dedicated data transformation module is designed to convert IDS monitoring data into a structured representation suitable for reinforcement learning. This module bridges the gap between supervised learning and reinforcement learning by mapping static feature vectors to 2D maps and then generating state representations. In this step, the features of each flow are rearranged into 7×7 (for UNSW) and 8×10 (for CICIDS) matrices and then fed into the feature extraction section. In the CNN section, two convolution layers with 3×3 filters and 32 and 64 filters, along with MaxPooling layers, are used to extract multi-level traffic features from basic patterns such as packet rate and volume to complex patterns related to attack behavior.

The output of this block is flattened into a compressed vector that forms a high-level representation of the flow. In this study, the convolutional network's output is used directly as a state representation in a Dueling-Q-based reinforcement learning environment, enabling the learning agent to learn adaptive decision-making policies in response to changing network behaviors based on the extracted deep features. This module consists of two separate processing paths: the first path estimates the overall quality of the network state without direct dependence on the choice of a specific action, and the second path evaluates the relative superiority of each possible action in the same state. The separation of these two paths allows the model to provide a more stable and separable estimate of decision quality in situations where the difference between decision outcomes is insignificant, or the decision signal is weak and noisy. The decision space in this framework is limited to two actions,

"normal" and "attack," and at each step the agent alternates between random selection and selection based on previous experience, using a heuristic-exploratory policy. After a decision is made, the system feedback is calculated based on the actual flow label and assigned to the agent as a reward. One of the key innovations of this research is the purposeful design of the reward function with an emphasis on reducing critical errors (false negatives). In this design, the penalty for not detecting an attack is asymmetric and much heavier than the penalties for other errors, so the learning process is structurally directed towards increasing sensitivity to attacks and improving the Recall criterion. Each interaction, including the current state, the selected action, the received feedback, and the next state, is stored in the Replay memory

and is used at specific time intervals to update the network parameters to reduce the temporal correlation of samples and achieve more stable convergence. In addition, to control fluctuations and prevent instability in the learning process, a separate target network is employed in the model structure, whose weights are periodically synchronized with the main network. This training cycle continues until a stable and reliable decision-making policy is achieved to distinguish normal and malicious traffic. Overall, the proposed framework goes beyond a simple combination of models and, by integrating feature extraction, data transformation, and decision-making policy learning, provides an adaptive and efficient intelligent system for anomaly detection in Fog-IoT environments.

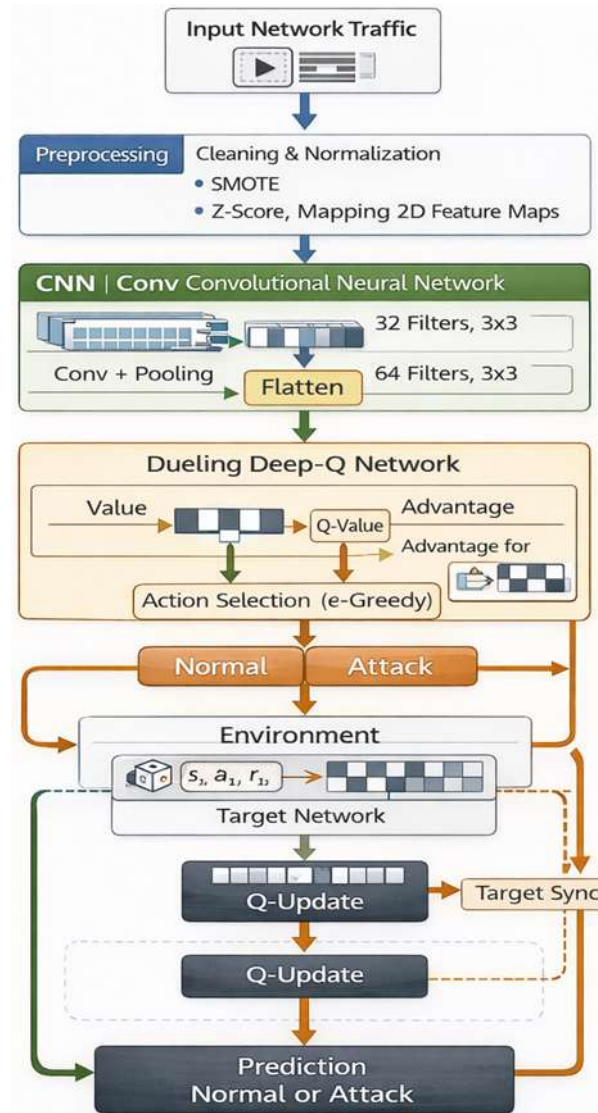


Figure 1. Diagram of the proposed method

2.1. Research Environment and Data Structure

In any research based on network traffic analysis and the identification of anomalous behaviors, the first critical step is to accurately define the data environment and specify the nature of the information to which the model is to be exposed. In the field of the Internet of Things, this becomes even more important because the data structure of IoT networks is fundamentally different from that of traditional networks. High rates and volumes, periodic behavior, small packet sizes, uniform yet fragile communication patterns, and a strong dependence on lightweight protocols usually characterize IoT traffic flows. These same characteristics make IoT traffic create complex conditions from a security perspective, and learning models must be able to extract subtle, irregular, and sometimes very short-term patterns from such data.

On the other hand, the nature of attacks in IoT networks is also distinct. Attacks such as botnet-based DDoS, fast PortScans, Brute-force attacks on unusual protocols, and malicious behaviors involving low-power devices all have a specific structure not seen in many classic network datasets. Therefore, the design of the proposed model in this study requires datasets that are not only suitable in terms of size and features but also capable of capturing the real diversity of IoT attacks and behaviors and enabling the extraction of reliable features for CNNs and reinforcement learning.

In addition, because the model in this study is a hybrid CNN–Dueling-Q model, it was necessary to select a dataset that has both reliable labeling (for CNN training) and continuous, time-dependent elements (for the RL model's interaction with the environment). Reinforcement networks for learning decision-making policies require an environment in which the sequence of states, actions, and consequences can be defined, so the design of the learning environment is also directly dependent on the data structure.

For this reason, this section first describes the general structure of the study data environment. Then, the two main datasets used, UNSW-NB15 and CICIDS2017, are fully introduced, and the scientific reasons for their selection, their technical characteristics, the types of attacks they cover, their feature structures, and their suitability for deep learning and reinforcement learning models are explained. This introduction provides the necessary framework to describe the preprocessing, normalization, data rearrangement, and construction of suitable inputs for the hybrid architecture of this research with the required accuracy and coherence.

2.2. Introduction to the nature of the data

The data used in this study are network traffic in Internet of Things environments; data that is transmitted in continuous streams between sensors, fog nodes, and servers, and has distinct and complex characteristics compared to traditional networks. IoT traffic is usually voluminous, continuous, lightweight, and heterogeneous, and contains large amounts of noise, behavioral variations, and nonlinear patterns; factors that make anomaly detection difficult and highlight the need to use deep and robust models. One of the major challenges of this type of data is the severe imbalance between normal samples and attacks; typically, most of the data is normal, and only a small percentage is labeled as attacks. This causes classical models, without appropriate measures, to incorrectly classify samples as normal. Therefore, in this research, in addition to designing a reinforcement learning mechanism sensitive to False Negative error, resampling methods such as SMOTE have also been used to improve data distribution. For use in deep learning networks, raw traffic data are first transformed to the flow level, and then their statistical, temporal, and behavioral features are extracted. After normalization, these features are reorganized into pseudo-image feature maps for processing by CNN layers. Since network behavior over time is important, the data are presented to the model as continuous-time sequences so that the reinforcement learning section can use state changes to learn the optimal decision-making policy. Overall, the data used in this research reflect the real behavior of IoT networks, and their dynamic nature justifies the need to combine deep learning and reinforcement learning methods to create a robust and accurate anomaly detection system.

2.3. Data Preprocessing

For raw traffic data to be usable in deep learning models, it must be standardized in quality and structure before entering the training stage. IoT network data typically contains noise, missing values, heterogeneous scales, and an imbalanced distribution between normal and malicious samples; therefore, it is essential to implement a systematic, staged preprocessing pipeline to mitigate model bias and improve final accuracy.

In this study, data preprocessing was performed in four main steps: cleaning and removing initial data anomalies, normalizing and equalizing feature scales, correcting class imbalance, and finally converting the data structure into a format suitable for processing by convolutional networks.

These steps are the basis for creating a consistent, clean, and reliable dataset for model training.

2.3.1. Data Cleaning

In the first step of data cleaning, features that do not directly contribute to traffic behavior and attack detection, or that are discontinuous and unusable in deep learning models, are removed. The most important of these features in both datasets are: source address, destination address, some textual or symbolic identifiers, and the exact time of recording each packet or flow.

These types of features are usually dependent on the specific data collection scenario and cannot be generalized to other environments; for example, a model should not consider a specific IP as an “attack” in all networks simply because it appears as an attacker in the dataset. Leaving these variables can lead to overfitting and to learning patterns that depend solely on the IP and the local structure of the scenario. In contrast, the goal of the research is to identify general behavioral patterns in network traffic, not to maintain a blocklist of addresses.

On the other hand, many of these variables are categorical or textual, and their direct use in convolutional network-based architectures requires complex transformations (such as one-hot encoding or embeddings) that unnecessarily increase data dimensionality and computational cost without yielding the same level of accuracy improvement. Therefore, after an initial examination of their correlation and role in explaining traffic behavior, these variables have been removed from the input feature set to limit the data to a compact, usable set of numerical features for deep learning.

Handling Missing Values:

In the next step, the status of missing values in numerical features was examined. Traffic data, especially in large datasets, may contain empty cells or invalid values for various reasons, such as errors in registration, defects in feature extraction, or the absence of some values in certain packets. The presence of these values, if left unaddressed, can disrupt the normalization process, the calculation of descriptive statistics, and the model's training, leading to errors or bias in the results.

To manage this situation, the distribution of missing values for each feature was first examined. In cases where the proportion of missing values in a feature was significant, and the feature was not conceptually important, that feature was completely removed from the model to prevent the introduction of noise and incomplete information. In

contrast, for features that played an important role in describing traffic behavior and only a small portion of their values were missing, simple statistical imputation methods were used instead of complete removal; for example, replacing missing values with the mean or median of the same feature in the training set.

This approach has resulted in, on the one hand, the useful information in the data being preserved as much as possible, and on the other hand, the numerical structure of the dataset remaining orderly and free of gaps for use in subsequent steps (normalization, rearrangement in the form of a feature map, and input to the neural network). In this way, the model's input data are free of empty values and prepared for processing by deep learning algorithms.

2.4. Normalization

Since network traffic features have different numerical scales, a lack of normalization can cause features with larger values to dominate the learning process and weaken the role of other features, especially in gradient-based models. Therefore, to standardize feature scales and improve the learning process, the Z-score normalization method has been used in this research.

In this method, each numerical feature is transformed so that its mean is zero and its standard deviation is one. For each value of x in a feature, the normalized value X is calculated as in equation (1):

$$X = \frac{x - \mu}{\sigma} \quad (1)$$

In equation (1), μ is the average of the values of the same feature, and σ is its standard deviation. In this way, the data is centered on zero, and all feature scales are the same. This has several important advantages:

- 1) improving the convergence of learning algorithms (especially in deep networks),
- 2) reducing the sensitivity of the model to the difference in scale between features,
- 3) Providing suitable conditions for the use of activation functions such as ReLU and improving the stability of gradients during training.

In practice, this transformation is first applied to the training set, and the mean and standard deviation of each feature are calculated from the training data. Then, the same values of μ and σ are used to normalize the validation and test data, preventing data leakage and keeping the evaluation conditions consistent with the real environment. After applying Z-Score normalization, all features are placed on a

standardized, comparable scale, and the data is ready to proceed to the next steps, including rearrangement into a feature map and processing by a convolutional network.

2.5. SMOTE Data Balancing

The SMOTE algorithm is a well-known method for intelligent minority-class oversampling that generates new artificial examples in the feature space rather than simply repeating existing examples (which can lead to overfitting). The main idea of SMOTE is that if a few rare examples (e.g., a few attack examples of a particular type) are located in a certain neighborhood of the feature space, new points can be created by interpolation between these examples that are consistent with the actual distribution of the minority class and expand the decision region of that class in the feature space.

The general process of the algorithm is as follows:

1. Selection of the minority class: First, the class whose number of examples is smaller than that of other classes (e.g., a specific attack type) is selected as the target class for oversampling.

2. Selection of nearest neighbors (k-Nearest Neighbors):

For each example (x_{new}) of the minority class, using criteria such as Euclidean distance (for numerical features), k nearest neighbors are identified from other examples of the same class. The value of k is usually taken as a small number (e.g., 5).

3. Artificial sample generation: To generate each new sample, one of the nearest neighbors (x_{nn}) is randomly selected, and a new point on the line between these two samples is generated. The artificial sample generation equation is usually defined as equation (2):

$$(2) \quad x_{new} = x_i + \lambda(x_{nn} - x_i)$$

Where λ is a random number in the interval [0,1]. Thus, the new point (x_{new}) lies within the linear segment between two real samples from the minority class and will be statistically consistent with the class's data distribution.

4. Iteration until the desired proportion is reached:

This process is repeated for the required number of samples until the minority class size reaches the desired level (e.g., close to the size of the normal class or a certain percentage of it). For example, if the share of an attack class is only 5% and the goal is to reach 20%, SMOTE will

controllably generate the required number of artificial samples.

The important feature of SMOTE is that, instead of creating duplicate examples (copies), it creates new points in the feature space that lie between real examples; as a result, the minority-class cluster is expanded, and the model's decision boundary around it becomes more stable and reliable. On the one hand, this helps the model learn attack-class patterns more effectively; on the other hand, it somewhat prevents overfitting on a limited number of examples because the model is exposed to a combination of real and interpolated synthetic data, not just a few duplicate examples.

There are several important technical points to consider when implementing SMOTE correctly to avoid analytical errors:

Apply oversampling only to training data: SMOTE should be applied only to the training set, leaving the validation and test data untouched. Oversampling the entire dataset or the test set will lead to information leakage and overly optimistic estimates of model performance.

Proportion of the oversampling ratio to class size: The oversampling level should be chosen to create a suitable relative balance between classes, without unrealistically increasing the number of artificial samples. Usually, instead of making the classes exactly equal, a reasonable ratio (e.g., bringing the minority class closer to 30-50% of the majority class) is chosen to create relative balance while avoiding destroying the real structure of the data.

Sensitivity to noise and outliers: If there are noisy points or outliers in the minority class, SMOTE may produce undesirable artificial samples by interpolating between them. For this reason, in practice, it is common to perform an outlier-removal step or at least a statistical check of the minority-class distribution before running SMOTE.

Distance type and feature space: In this study, the data are mostly numerical and continuous, so using Euclidean distance to find nearest neighbors in SMOTE is a natural and appropriate choice. For nominal or binary features, it is necessary to use extended versions of SMOTE or hybrid methods; however, for the datasets used (UNSW-NB15 and CICIDS2017), the focus was on numerical features.

2.6. Data transformation into CNN input structure

In this study, to adapt IDS surveillance data to a deep reinforcement learning framework, a data transformation module is designed to map static feature vectors into

structured representations suitable for both CNNs and reinforcement learning. This module serves as a bridge between tabular data and the sequential decision-making process, enabling the definition of meaningful states for the learning agent. In this framework, raw data, including 49 features in the UNSW-NB15 dataset and 80 features in the CICIDS2017 dataset, are considered as the initial input.

In this study, for the UNSW-NB15 dataset, which includes 49 features, the feature vector is transformed into a 7×7 matrix. This rearrangement is not just a simple shape transformation. However, it is designed to preserve local correlations between features and to create a suitable structure for the CNN to extract spatial patterns. On the other hand, this 2D representation provides the necessary context for generating state representations in reinforcement learning, enabling the agent to understand relationships among features within a structured state space. Similarly, in the CICIDS2017 dataset, which has 80 features, the feature vector is rearranged into an 8×10 matrix. The choice of these dimensions is made considering the total number of features and the possibility of forming a regular rectangular grid, so that:

1. All features are present exactly once in the feature map (without repetitions and deletions),
2. The dimensions of the resulting matrix are compatible with the size of the convolution kernels (e.g., 3×3).
3. The final shape is suitable and stable for convolution and pooling calculations.

The transformation process is implemented so that, first, for each sample, the normalized feature vector (after Z-scoring) is treated as sorted (in a fixed order across all data). Then, this vector is arranged into a two-dimensional matrix in rows or columns; for example, for 49 features, the first 7 elements of the first row, the next 7 of the second, and so on up to the seventh row. Similarly, the 80 CICIDS features are placed in an 8×10 matrix. The important point is that the order of the features in this matrix is maintained constant and the same for all samples, so that the convolutional network is trained on a structured and interpretable space.

Mathematically, if we represent the feature vector for a sample by $x \in \mathbb{R}^d$, where ($d=49$) or ($d=80$), the reshape operation acts as a linear transformation without losing information according to equation (3):

$$(3) \quad x \in \mathbb{R}^d \Rightarrow X \in \mathbb{R}^{h \times w}$$

Where in equation (3) ($\cdot$), for example, for UNSW: ($h=7$, $w=7$) and for CICIDS: ($h=8$, $w=10$). In this transformation, the value of each element in the matrix X is just a shifted

version of the same value in the original vector x , and therefore, the Reshape operation does not make any changes to the numerical information; it only changes the geometric structure of the representation to suit the needs of the CNN.

The advantage of this innovation is that convolutional filters (e.g., 3×3 or 5×5 kernels) can slide over this matrix and analyze subregions of adjacent features simultaneously, rather than examining each feature individually. This proximity in the matrix can be adjusted by properly designing the order of the features so that semantically or statistically related features are placed as close as possible to each other; for example, features related to packet rate, data volume, and timing in one area, and protocol features and flags in another area. In this way, the convolutional network will be able to learn common patterns across groups of features, rather than just the effect of each feature alone.

From an implementation perspective, these 2D feature maps are given to the CNN as single-channel inputs, similar to grayscale images, where each 7×7 or 8×10 matrix represents a small image. In the first convolution layer, several filters (e.g., 32 or 64) are applied to these maps, and each filter extracts a specific combination of feature neighborhoods. The result is the generation of several new feature maps in subsequent layers that represent more complex statistical and behavioral patterns in network traffic.

Overall, the proposed data transformation module goes beyond simple preprocessing and plays a key role in integrating the model's components by establishing an effective mapping between the raw feature space and the reinforcement learning state space. This design enables the effective use of monitoring data within an adaptive decision-making framework and is considered an innovative component of the proposed method.

2.7. Proposed Deep Learning Model Architecture

The architecture of the proposed model in this study is a deep hybrid structure based on CNN and Dueling-Q deep reinforcement learning, designed to identify network traffic anomalies in IoT environments and at the fog layer. The main idea of this architecture is first to extract compact, meaningful representations of traffic flows using a convolutional network, and then use these representations as "states" in a reinforcement learning framework to learn the optimal attack/normal behavior detection policy automatically. In the following, the components of this architecture, including the CNN block, the Dueling-Q block,

and the reinforcement learning environment, are explained in technical and detailed terms.

2.7.1. CNN-based feature extraction block

Input structure to CNN:

The input to this block is the traffic feature vectors, displayed as a 2D feature map after preprocessing steps (cleaning, normalization, imbalance removal, and rearrangement). For each flow, a matrix X is formed as in equation (4):

$$(4) \quad X \in \mathbb{R}^{H \times W \times 1}$$

Where in equation (4):

For UNSW-NB15 database: ($H = 7$, $W = 7$) (49 features),

For CICIDS2017 database: ($H = 8$, $W = 10$) (80 features),

"1" indicates a one-dimensional channel (single-channel input, such as a gray image).

The features in this matrix are arranged according to a fixed order that is maintained throughout the training process, so that the relative spatial structure between related features is also maintained; So that features of the same family (for example, features related to volume, rate, time, flags, etc.) are placed in close areas as much as possible.

Details of Convolution and Pooling Layers

The feature extraction block, based on a convolutional neural network, is designed to hierarchically extract both simple local patterns and more complex structures in network traffic. In the first convolution layer, 32 3×3 filters with same padding are used to identify basic local patterns, such as short-range changes, small correlations between features, and basic traffic structures, while maintaining the input spatial dimensions. The choice of 3×3 filters were made because this size, while having low computational cost, effectively extracts local patterns and is a standard choice in many successful convolutional architectures. The ReLU activation function is also used to increase the model's nonlinearity and prevent gradient saturation. After this layer, a Max-Pooling layer with a 2x2 window and a step size of 2 is applied, which aims to reduce the spatial dimensions of the features, eliminate redundant information, and increase the focus of the model on dominant patterns; this dimensionality reduction also reduces computational complexity and increases stability in subsequent processing stages. Next, a second convolution layer with 64 filters and a 3×3 kernel size is used to model higher-level, more combined patterns from the features extracted in the previous layer, such as behavioral patterns related to Burst attacks and specific DoS/DDoS patterns. Increasing the

number of filters in this layer is done because, at higher levels of the network, the diversity and complexity of patterns are greater, and a higher representational capacity is required. In some settings, another Pooling layer is also used to reduce the spatial dimensions further and control the size of the final feature vector, ensuring that the input to the reinforcement learning section remains within a reasonable dimensionality range. Finally, the output of the last convolution layer is transformed into a 1D vector by the ($z = \text{Flatten}(F_{\text{conv}})$) that ($z \in \mathbb{R}^d$) so that this compact and meaningful representation is used as input to the Dueling-Q module. This step-by-step design causes the network to first extract simple features and then gradually learn more complex patterns. At the same time, the filter sizes, their number, and the Pooling layers are purposefully chosen to strike a balance between model accuracy and computational cost.

CNN Training Settings and Regularization:

For training the convolutional neural network, a set of settings is considered to achieve stable convergence and reduce overfitting. In the supervised CNN pre-training stage, the Binary Cross-Entropy or Categorical Cross-Entropy cost function is used, depending on the problem type, to align the learning process with the nature of the classification task. Parameter optimization is performed with the Adam algorithm because it provides faster, more stable convergence in problems with complex parameter spaces than simple SGD. The initial learning rate is chosen to be equal to 1 to prevent fluctuations and improve convergence in the final stages; a gradual decrease in the learning rate is applied during the training periods. The training batch size is set to a range of 32 to 128 to balance gradient stability and training speed. To increase the model's generalization capability, Dropout with a rate of 0.5 and L2 regularization on the weights are applied to the fully connected layer before it connects to the reinforcement learning section. In the final version, two training strategies are considered: in the first strategy, the CNN is pre-trained in a supervised manner, and then its weights are fixed, and only the Dueling-Q section is trained; in the second strategy, the entire architecture is trained end-to-end and jointly, the details of which will be explained below.

2.7.2. Dueling-Q block for learning decision-making policy

After extracting the compressed representation (z) from the convolutional neural network, this vector is used as the

state representation (s) in the reinforcement learning module based on the Dueling Deep Q-Network. In this research, the intrusion detection problem is redefined from a static classification problem to an adaptive decision-making process, such that the learning agent, observing the state (s), selects the optimal action in the decision space.

In the proposed architecture, instead of directly estimating the value-action function ($Q(s, a)$), the Dueling structure is used, in which the Q-value is decomposed into two independent components: the state value and the action advantage. This decomposition is defined as equation (5):

(5)

$$Q(s, a; \alpha, \beta) = V(s; \theta, \beta) + (A(s, a; \theta, \alpha) - \frac{1}{|A|} \sum_{a'} A(s, a'; \theta, \alpha))$$

where:

- $V(s)$ the overall value of the network state without dependence on the specific action,
- $V(s, a)$ shows the relative advantage of each action in the current state,
- The averaging term is used to prevent redundancy between the two components.

In the first step, the state vector (s) is fed into a shared Fully Connected layer and then split into two separate streams:

- Value Stream:

Estimates the $V(s)$ value that represents the overall quality of the network state.

- Advantage Stream:

Calculates the $V(s, a)$ value for each action and determines the relative advantage of decisions.

This structure allows the model to provide a more stable and separable estimate of decisions in situations where the difference between actions is small, or the data is noisy and unbalanced.

The objective function for learning this network is defined using the Temporal-Difference error. For each experiment (s, a, r, s') , the objective value is calculated as in equation (6):

$$y = r + \gamma \max_{a'} Q_{target}(s', a'; \theta^-)$$

(6)

where:

- γ is the discount factor,
- Q_{target} is the Q function of the target network with θ^- parameters.

Then the cost function for updating the network parameters is defined as follows:

$$(7) \quad L(\theta) = E_{(s, a, r, s') \in D} [(y - Q(s, a; \theta))^2]$$

Where (D) is the set of experiences stored in the Replay Buffer.

One of the key innovations of this research is the purposeful design of the reward function in line with security considerations. Specifically, the reward is defined so that the False Negative error (failure to detect an attack) incurs a much larger penalty than other errors. This design causes the value function to be structurally directed towards maximizing Recall for the attack class and makes the learning agent more sensitive to real threats.

To increase learning stability, the Replay Buffer mechanism is used, in which experiences are stored and randomly sampled for training. This reduces the temporal correlation of the data and improves convergence. Also, a separate target network with defined parameters is periodically synchronized with the main network:

$$\theta \rightarrow \theta^-$$

This mechanism prevents instability in estimating the Q value and stabilizes the learning process.

Overall, the use of the Dueling-Q architecture in this framework, together with the purposeful design of the reward function and direct connection to the deep representations extracted by the CNN, has led to the creation of an adaptive and security-oriented decision-making agent that is capable of making more accurate decisions in dynamic and unbalanced conditions of IoT networks.

2.7.3. Definition of the Reinforcement Learning Environment (RL Environment)

In the proposed reinforcement learning framework, the state at each decision-making step is defined as a compact representation of the current state of network traffic, obtained directly from the convolutional neural network's output. At each step, either the entire traffic flow or, if time windows are used, a short subsequence of several consecutive flows is selected and, after passing through the CNN, converted into a fixed-dimensional feature vector. This feature vector summarizes the important statistical and structural patterns in the raw data and serves as a description of the environment's state for the learning agent. The use of time windows allows the state to reflect, in addition to information from a single instance, short-term trends in traffic changes, thereby including short-term temporal dependencies in the decision-making process. The

application space in this problem is defined and discretized into two decisions: "label as normal traffic" and "label as attack/anomaly," so that the model remains focused on learning an effective detection policy. The environment's feedback is determined by comparing the model's decision with the actual flow label. It is designed asymmetrically, such that correct attack detection receives a higher reward, while errors in detecting an attack as normal traffic incur a heavier penalty. This design makes the agent more sensitive to critical errors during training, and the decision-making policy is naturally directed towards increasing Recall for the attack class. In this way, states are directly constructed from the deep representations extracted by the CNN, thereby establishing a clear link between feature extraction and reinforcement learning-based decision-making. This environment forms the communication loop between the data, the CNN, and Dueling-Q.

$$\text{Reward Function} = \begin{cases} (r = +1) & \text{, the flow is truly normal and the model recognizes it as normal} \\ (r = +2) & \text{, The flow is truly an attack and the model recognizes it as an attack.} \\ (r = -2) & \text{, model gives wrong label} \\ (r = -5), & \text{The flow is an attack but the model recognizes it as normal (False Negative)} \end{cases}$$

This design makes the model learn over time that incorrectly labeling an attack as normal is very costly; that is, recall for the attack class naturally improves.

Structure of episodes and interaction with data

In the proposed reinforcement learning framework, the process of agent interaction with network traffic data is organized episodically. Each episode consists of a sequence of consecutive traffic flows that can represent a specific time interval of network behavior (for example, a portion of daily traffic or an attack scenario). This episodic structure enables the gradual learning of a decision-making policy in response to dynamic traffic changes.

In each episode, the learning agent interacts with the environment step by step. At step t , a traffic sample or a short data subsequence is first selected and, after passing through the data transformation module and the convolutional network, is transformed into a state representation. This state summarizes the important traffic features at that moment and serves as input to the decision-making module.

After determining the state, the agent uses a heuristic-exploitative policy (usually greedy) to select one of the possible actions. Next, the reward value is calculated by comparing the decision made with the actual data label. The reward design is asymmetric, so that critical errors (especially False Negatives) incur a greater penalty.

Definition of States

At each time step t , a traffic flow or a subsequence of flows is selected, and after passing through the CNN, the state is defined as (8):

$$(8) \quad s_t = \text{CNN}(x_t)$$

If time windows are used, they can be considered to represent multiple sequential flows.

Action Space

The action space is finite and discrete:

$$\text{Action Space} = \begin{cases} a = 0, \text{Tag as normal traffic} \\ a = 1, \text{Tag as attack/anomaly} \end{cases}$$

This simple choice allows the model to focus on the detection strategy rather than the complexity of the action structure.

After receiving the reward, the environment is transferred to the new state, and the resulting experience is stored as a quadratic in the form of equation (9):

$$(9)$$

These experiences are stored in the Replay Buffer, and at certain time intervals, a subset is randomly sampled to update the network parameters. This mechanism reduces the temporal correlation of the data and increases the stability of the learning process.

The network update is based on minimizing the TD error and uses samples extracted from the Replay Buffer. Also, to avoid fluctuations during learning, a separate target network is used, whose parameters are periodically synchronized with those of the main network.

This interaction cycle, including state observation, action selection, reward acquisition, experience storage, and model update, continues until the end of the episode and is repeated in successive episodes. As a result, the learning agent will gradually learn a stable, optimal policy for separating normal and malicious traffic under dynamic, unbalanced conditions in IoT networks.

2.7.4. End-to-End Model Training Process

The proposed architecture training can be done in two phases:

1. CNN Pre-training (Supervised Pre-training)

The CNN is trained in a classical manner on normal/attack labels to learn a basic representation.

2. CNN and Dueling-Q Joint Training (End-to-End RL)

In this phase, the RL cost is propagated to higher layers, and CNN weights are updated as needed.

In the End-to-End version, the overall cost function can be a combination of the RL cost and an auxiliary loss component; in the simplest case, the RL cost alone is sufficient. In this way, the CNN learns to extract features that are directly useful for improving the RL policy, rather than just for supervised classification.

2.7.5. Considerations for deployment in the fog layer

Given that the target environment is fog nodes with limited resources, the following have been considered in the architectural design:

Small input dimensions (77, 810), reducing the computational cost of the CNN

Using two relatively lightweight convolution layers suitable for execution on lightweight CPUs or GPUs

A limited number of actions reduces the complexity of RL

Possibility of module separation:

CNN can be trained on more central nodes (Fog Master) and then transferred to sub-nodes.

The RL network can also be implemented with small-sized versions.

This structure allows the model to be implemented in Fog-Based IDS scenarios.

3. Findings and Results

This section presents a series of systematic experiments to evaluate the efficiency, stability, and originality of the model. The main goal of this section is to show that the

(F1-Score)

$$F1-Score = \frac{2 * TP}{2 * TP + FP + FN}$$

(Precision)

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

(Recall)

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

(Accuracy)

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

In addition to the classic classification criteria, including Accuracy, Precision, Recall, and F1-Score, a set of system criteria has also been considered to evaluate the performance of the proposed model in fog computing-based

proposed model is not only theoretically but also experimentally capable of delivering superior performance over existing methods, and that each of its components plays an effective role in the model's overall improvement.

To achieve a precise, multidimensional evaluation, the experiments are organized into three categories. In the first category, the critical parameters of the model, including the Oversampling ratio, Replay memory size, and discount factor γ , are examined to determine the impact of each on the model performance indicators and to select the optimal values for continuing the experiments. In the second category, the originality tests or Ablation Study are performed, in which different parts of the proposed method are purposefully replaced with simpler alternatives or common structures, and the effect of removing or changing each component on the model performance is examined; this section proves that each component designed in the proposed model has a specific necessity and function. In the third category, the proposed method is compared with several established methods in the research literature to assess its performance relative to current standards. Taken together, these three sets of experiments provide a comprehensive, multi-layered picture of the model's capabilities and demonstrate that the proposed method has significant advantages not only at the parametric and optimized-architecture levels but also relative to reference methods.

4-1- Implementation environment of the proposed model

The proposed method in this research uses the PyTorch platform, the Keras (2.3.1) library, and Python 3.7.4 on an Intel (R) Core i9-13900 processor. In addition, the experiments were conducted on an NVIDIA GeForce 3050 graphics processing unit, with 40 GB of RAM and 64-bit Windows.

3.1. Research evaluation criteria

The most important criteria for evaluating the results are Accuracy, Precision, Recall, and F-score, each of which is defined as follows.

environments. These metrics include latency, response time, and load distribution across the edge, fog, and cloud layers.

The latency metric is defined as the time required to analyze each traffic flow from the moment it is received to

the time it produces the model's output. A reduction in this metric indicates the system's real-time usability in IoT environments.

Response time also indicates the system's ability to process more requests per unit time and is measured as the number of flows analyzed per second.

Also, to evaluate the processing load distribution, the ratio of the processing volume performed in the fog layer to that in the cloud is examined. This metric indicates the extent to which the proposed model has reduced dependence on the cloud and improved the system's overall efficiency by offloading processing to layers closer to the data source.

3.2. Database

The UNSW-NB15 dataset is one of the most comprehensive network security datasets, created by combining real traffic and attacks generated by the professional IXIA PerfectStorm tool. It contains more than 2.5 million samples with 49 behavioral and statistical features. Covering 9 modern attack types and a realistic, unbalanced distribution between normal and malicious traffic, this dataset accurately reflects today's network operating conditions. It is used as a standard reference for evaluating anomaly detection methods.

The CICIDS2017 dataset is also generated in a real-world environment, with more than 3 million traffic flow records and 80 features extracted by CICFlowMeter, and it covers a wide range of modern attacks such as DoS/DDoS, web attacks, and Botnets. The multi-scenario and unbalanced

data recording in this dataset make it a reliable and suitable source for evaluating the performance of intrusion detection systems in real network conditions.

3.3. Testing research parameters

Parameter 1: Oversampling ratio in SMOTE

The SMOTE method increases data diversity by generating artificial samples through interpolation in the minority-class feature space, thereby shifting the model decision boundary towards the attack region. To evaluate the effect of SMOTE, an experiment with three levels of oversampling, including 0%, 50%, and 100%, was designed, while other model parameters were kept constant. In all experiments, the proposed complete architecture, including CNN for feature extraction and Dueling-Q for reinforcement learning, was used to examine the effect of oversampling independently. The results showed that a reasonable increase in Oversampling improves the attack detection power. In the case without SMOTE, despite the relatively high Accuracy, the Recall of attacks is low, and some attacks are not detected. Applying SMOTE at a 50% level provided a good balance among Precision, Recall, and F1 and delivered the best overall performance. Increasing Oversampling to 100%, while increasing Recall, is accompanied by a decrease in Precision, indicating more false alarms. As a result, 50% Oversampling was chosen as the optimal value, and the results show that addressing data imbalance is an essential component of the proposed model's performance.

Table 1. Effect of SMOTE ratio on model performance on UNSW-NB15

SMOTE ratio	Accuracy attack	Precision attack	Recall attack	F1 attack
No SMOTE	97.8%	94.1%	86.3%	90.0%
50%	98.6%	95.3%	91.7%	93.5%
100%	98.4%	94.8%	92.5%	93.6%

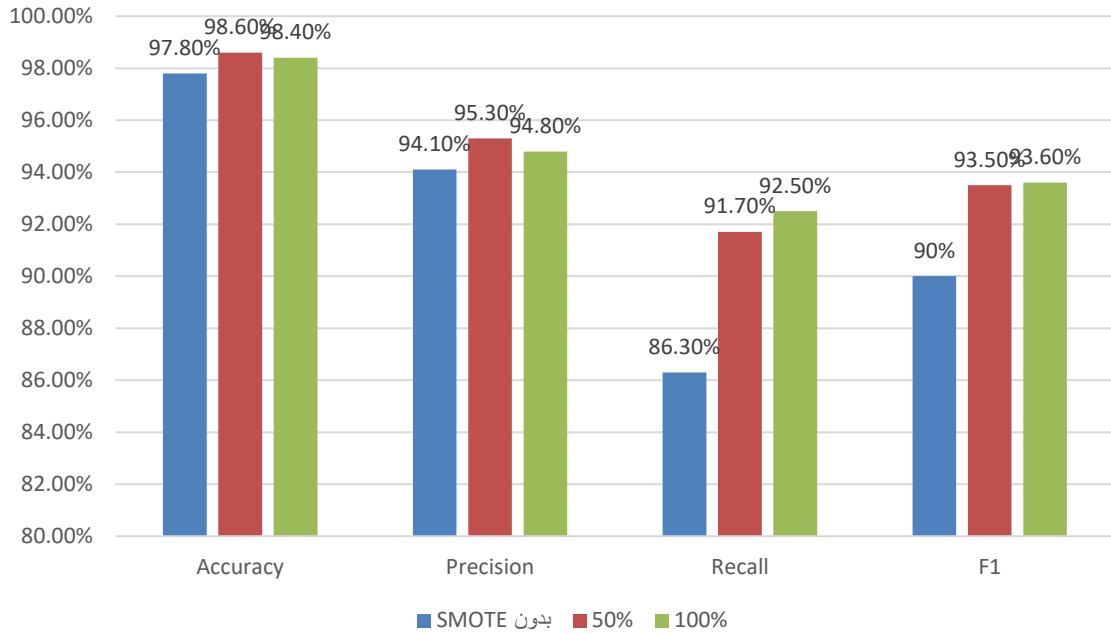


Figure 2. Effect of SMOTE ratio on model performance on UNSW-NB15

The results show that using SMOTE directly improves the detection of attack classes. In the case without SMOTE, although the overall Accuracy is relatively high, the Recall for attacks is 86.3%, indicating that some attacks are still undetected. By applying Oversampling at 50%, the Recall for attacks has increased to 91.7% and the F1 to 93.5%, indicating a better balance between Precision and Recall. A sharp increase in oversampling to 100% has not produced any further improvement. It has even caused a slight decrease in Precision, which may indicate that the model is approaching the Overfitting region. Accordingly, a 50% ratio is chosen as the optimal value for continuing the experiments.

Parameter 2: Replay Buffer Size in Reinforcement Learning

Table 2. Effect of Replay Buffer Size on Model Performance

Average reward fluctuation in final episodes	Accuracy	F1 attack	Replay Buffer Capacity
High	98.1%	92.1%	10.000
Medium	98.6%	93.5%	50.000
Low but slower training	98.5%	93.6%	100.000

The results show that increasing the experience memory capacity helps to stabilize learning and improve performance to a certain extent—a low capacity of 10,000 leads to higher sample correlation and greater reward fluctuations in final episodes. Increasing the capacity to 50,000 strikes a good balance between experience diversity and computational

The Replay Buffer plays an important role in stabilizing learning and improving policy convergence by reducing the temporal correlation of experiences. Experiments with capacities of 10,000, 50,000, and 100,000 showed that small capacities led to reward fluctuations and reduced generalization power due to limited data diversity. A capacity of 50,000 provided a good balance between experience diversity and computational cost, leading to training stability and improved Recall and F1 metrics, while increasing the capacity to 100,000 did not yield a noticeable improvement and slowed learning. Accordingly, a capacity of 50,000 was selected as the optimal value of Replay Buffer.

cost, leading to an improvement in the F1 score for the attack class. A capacity of 100,000 slightly improves performance but slows convergence. Therefore, a value of 50,000 is considered a balanced and practical choice for continuing the experiments.

Parameter 3: discount factor γ in reinforcement learning

The discount factor γ is a key reinforcement learning parameter that determines the agent's attention to future rewards and directly impacts the model's convergence, stability, and accuracy. In network anomaly detection, because traffic is temporal and continuous, the choice of γ is particularly important. Small values of γ limit the model's focus to short-term rewards and prevent learning long-term attack patterns, while very large values close to 1 cause slow training and excessive sensitivity to low-importance patterns. Therefore, choosing a balanced value of γ is essential for learning a stable and effective decision-making policy.

To accurately analyze the effect of γ , this study conducted experiments with three different values: $\gamma = 0.90$, $\gamma = 0.95$, and $\gamma = 0.99$.

The discount factor γ is a key reinforcement learning parameter that determines the agent's attention to future rewards and directly impacts the model's convergence, stability, and accuracy. In network anomaly detection, because traffic is temporal and continuous, the choice of γ is particularly important. Small values of γ limit the model's focus to short-term rewards and prevent learning long-term attack patterns, while very large values close to 1 cause slow training and oversensitivity to unimportant patterns. Therefore, choosing a balanced value of γ is essential for learning a stable and effective decision-making policy.

Table 3. Effect of γ coefficient on model performance

Γ	Accuracy	Precision attack	Recall attack	F1 attack
0.90	98.2%	94.6%	90.1%	92.3%
0.95	98.5%	95.0%	91.4%	93.1%
0.99	98.6%	95.3%	91.7%	93.5%

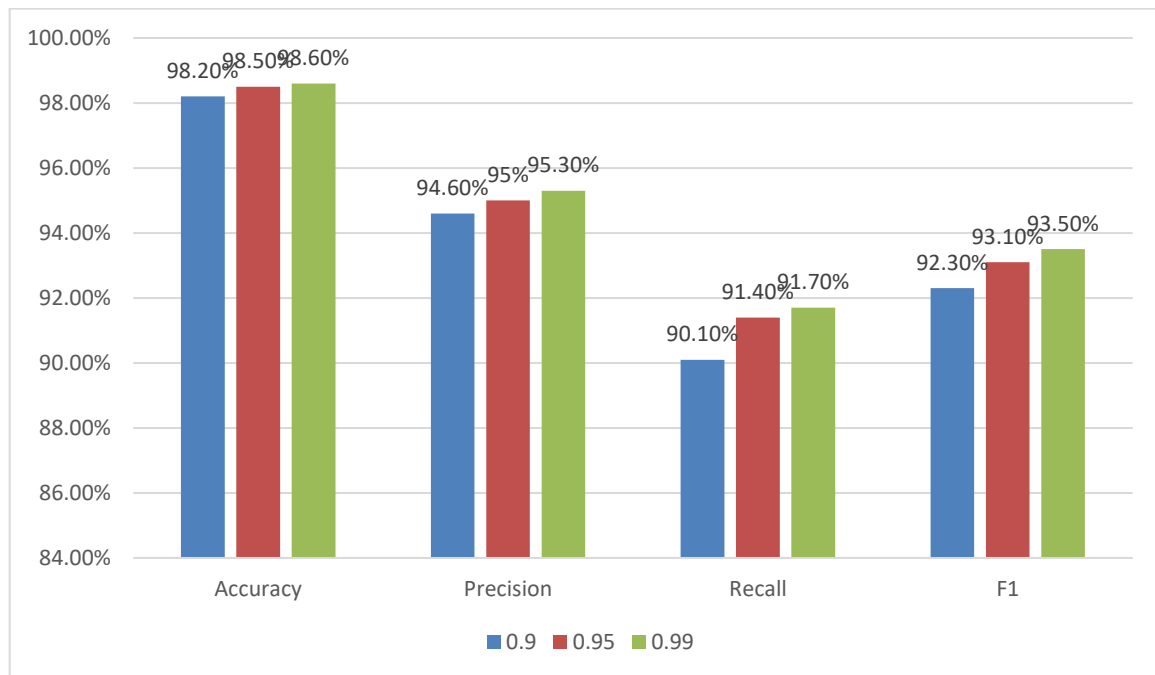


Figure 3. Effect of γ coefficient on model performance

As γ increases, the model places greater weight on future rewards and pursues gradual policy improvement over a longer horizon, rather than focusing solely on short-term decisions. The results show that setting γ to 0.99 yields the best balance between overall Accuracy and attack-class F1. At lower values, although training is faster, the model tends to make short-term decisions, and the final policy quality

decreases. Accordingly, $\gamma = 0.99$ has been used as the default value in the rest of the research.

3.4. Research Originality Test

The originality test was conducted to evaluate the role and necessity of the main components of the proposed model by controlled removal or replacement of each component. The

results showed that replacing Dueling-Q with the classic DQN led to a significant decrease in Recall and F1, confirming the importance of separating Value and Advantage. Removing CNN and using a simple MLP also led to a sharp decrease in the model's ability to extract complex traffic patterns and detect attacks. In the version without reinforcement learning, although the overall Accuracy remained high, the attack's Recall and F1 were significantly reduced, indicating the role of RL in reducing

False Negatives and learning a long-term policy. Also, removing SMOTE caused a sharp drop in Recall and increased the model's bias towards the normal class. Overall, the results of the originality test show that all components of the proposed architecture—CNN, Dueling-Q, SMOTE, and Replay Buffer—play a fundamental and irreplaceable role in improving learning stability and increasing the Accuracy of attack detection.

Table 4. Comparison of different versions of the proposed model on the UNSW-NB15 database

Model	Change Description	Accuracy	Precision-attack	Recall-attack	F1-attack
M0	Full Model	98.6%	95.3%	91.7%	93.3%
M1	Replace Dueling with DQN	98.2%	94.7%	89.5%	92.0%
M2	Replace CNN with MLP	97.4%	92.1%	87.2%	89.6%
M3	Remove RL, just supervised CNN	97.9%	93.1%	88.0%	90.6%
M4	Remove SMOTE	97.8%	94.1%	86.3%	90.0%

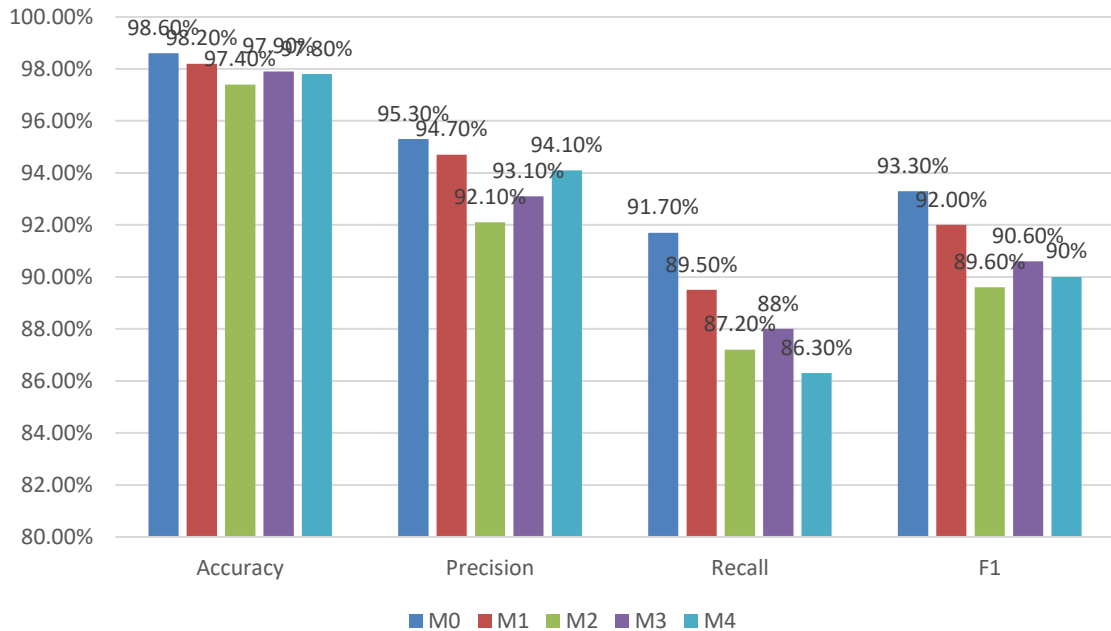


Figure 4. Comparison of different versions of the proposed model on the UNSW-NB15 database

The results in the table show that each of the key components of the proposed model plays a role and has an independent effect on improving performance. Removing the Dueling structure and using a simple DQN results in a significant decrease in Recall and F1, underscoring the importance of separating Value and Advantage. Replacing CNN with an MLP results in a significant decrease across all criteria, confirming CNN's role in extracting multidimensional patterns. Removing reinforcement learning and using a supervised classifier, despite maintaining a relatively high Accuracy, leads to a decrease

in Recall, which indicates the role of RL and the FN-sensitive reward function in effective attack detection. Also, removing SMOTE causes a sharp decrease in Recall in unbalanced data, while Accuracy does not change much, indicating the accuracy paradox. Overall, the full model (M0) provides the best balance of performance and confirms the need to design the proposed components.

3.5. Evaluation of the system efficiency of the proposed model in the fog layer

In addition to detection accuracy, system efficiency criteria play a decisive role in the practicality of an intrusion detection system in IoT and fog layer-based environments. Since fog nodes must make decisions with low latency and under resource constraints, simply improving the

classification criteria is not enough for evaluation. Accordingly, in this study, the performance of the proposed model is examined through three key indicators: processing latency, response time/throughput, and processing load distribution. The purpose of this evaluation is to investigate the practical deployment of the CNN+Dueling-Q model in real Fog-IoT scenarios and compare it with several reference architectures.

Table 5. Comparison of system efficiency of different methods in the fog layer

Latency (ms)	Throughput (flow/second)	Load distribution (standard deviation of CPU usage)	Method
18.5	820	0.31	Random Forest
24.2	640	0.28	CNN-LSTM
21.7	710	0.30	Autoencoder+LSTM
15.3	910	0.19	CNN+Dueling-Q) Suggested(

The results of Table 5 show that the proposed CNN+Dueling-Q method also provides superior performance compared to the reference methods in terms of system efficiency. The processing latency of the proposed model is significantly lower than that of methods such as CNN-LSTM and Autoencoder+LSTM, indicating the suitability of the proposed architecture for real-time scenarios in the fog layer. Also, the system throughput of the proposed method exceeds 900 flows per second, indicating higher efficiency in network traffic processing than other methods. On the other hand, the reduction in the standard deviation of CPU consumption in the CNN+Dueling-Q method indicates that the processing load is distributed more evenly across computing resources and that excessive concentration of the load on a single node is prevented. This feature is particularly important for Fog-IoT environments that face resource constraints and require operational stability. Overall, these results confirm that the proposed model is a suitable option for practical deployment in the fog layer, not only in terms of detection accuracy but also in system efficiency.

3.6. Comparison test with published valid methods

A comparison with established methods in the literature was conducted to empirically evaluate and demonstrate the practical efficiency of the proposed model. In this regard, the model's performance on the UNSW-NB15 and CICIDS2017 datasets was compared against four benchmark methods: Random Forest, RNN-based IDS, Autoencoder+LSTM, and

CNN-LSTM. These methods were selected as representatives of classical, sequence-based, unsupervised, and deep-learning approaches to accurately assess the functional and operational superiority of the proposed architecture.

- 1) Optimized IDS (Autoencoder + CNN-LSTM)
- 2) FFL-IDS (Fog + Federated Learning + CNN)
- 3) Transformer-Fog-Edge SVM-FL IDS (Smart Grid)
- 4) Whale-ELM Fog IDS

The proposed CNN+Dueling-Q architecture, which combines convolutional feature extraction with reinforcement learning, was compared with reference methods under identical conditions on the UNSW-NB15 and CICIDS2017 datasets. The results showed that the proposed method outperformed classical and deep models across all metrics, especially Recall and F1 for the attack class. Methods such as Random Forest and RNN showed lower Recall due to their sensitivity to class imbalance and limited ability to capture complex patterns. At the same time, Autoencoder+LSTM and CNN-LSTM, despite their good performance, were weaker at detecting diverse attacks and reducing False Negatives than the proposed model. In contrast, the combination of CNN with the Dueling-Q structure and an FN-sensitive reward function enables the proposed model to learn a more accurate and stable decision policy and to achieve higher sensitivity to real attacks. Overall, the comparative results confirm the significant superiority and operational capabilities of the proposed method in intrusion detection.

Table 6. Comparison of the proposed method with existing methods on the dataset (UNSW-NB15)

Explanation of change	Accuracy	Precision attack	Recall attack	F1 attack
Random Forest	96.8%	90.5%	84.2%	87.2%
RNN-based IDS	97.3%	92.0%	86.7%	89.2%
Autoencoder + LSTM	97.6%	93.1%	88.5%	90.7%
CNN-LSTM IDS	98.0%	94.0%	89.6%	91.7%
Autoencoder+CNN-LSTM	98.0%	94.5%	89.8%	92.1%
Federated Learning+CNN	98.3%	95.0%	90.5%	92.7%
Transformer-SVM	97.9%	93.8%	88.3%	90.9%
Whale-ELM	98.2%	94.9%	90.0%	92.4%
Suggested method) CNN+Dueling-Q(	98.6%	95.3%	91.7%	93.5%

The results in Table 6 of the comparison show that the proposed CNN+Dueling-Q method provides superior and more stable performance than existing validated architectures in Fog and Fog-IoT environments, especially for critical security metrics such as Recall and F1 for the attack class. Models such as Autoencoder+CNN-LSTM and FFL-IDS, despite achieving good overall accuracy, have lower Recall than the proposed method due to their inability to learn a long-term decision-making policy or their limitations in simultaneously observing flows. Lighter methods, such as Fog-Edge IDS based on SVM and Whale-ELM, also show poorer performance in detecting complex

and low-rate attacks due to their limited ability to extract deep patterns and handle unbalanced data. In contrast, the combination of CNN and Dueling-Q, which uses spatial-statistical feature extraction and learns an adaptive decision-making policy based on a reward function sensitive to false negatives, has significantly increased Recall and F1. These results show that the proposed model, in addition to being theoretically coherent, is an efficient, reliable, and suitable solution for implementation on fog nodes from a practical perspective, and has a high ability to detect rare, low-rate, and hidden attacks in Fog-IoT networks.

Table 7. Comparison of the proposed method with existing methods on the database (CICIDS2017)

Explanation of change	Accuracy	Precision attack	Recall attack	F1 attack
Random Forest	97.1%	91.8%	85.9%	88.7%
RNN-based IDS	97.5%	92.6%	87.4%	89.9%
Autoencoder + LSTM	97.9%	93.5%	88.8%	91.1%
CNN-LSTM IDS	98.2%	94.3%	90.1%	92.1%
Autoencoder+CNN-LSTM (Tawfik, 2024)	98.4%	94.8%	90.9%	92.8%
Federated Learning+CNN (Rehman et al., 2024)	98.6%	95.2%	91.3%	93.2%
Transformer-SVM (Tariq et al., 2024)	98.2%	94.1%	89.4%	91.6%
Whale-ELM (Sahu et al., 2024)	98.5%	95.0%	90.6%	92.7%
Suggested method (CNN+Dueling-Q)	98.8%	95.6%	92.4%	93.9%

The results from the CICIDS2017 dataset show that the proposed CNN+Dueling-Q method provides stable and superior performance compared to four valid Fog-based IDS architectures, especially in the critical metrics of Recall and F1 attack class. Although models such as Autoencoder+CNN-LSTM and FFL-IDS achieve good overall accuracy due to their feature extraction and distributed structures, they show lower Recall than the proposed method due to their limitations in learning long-term decision policies and their less focus on FN errors. Lighter methods, such as Fog-Edge SVM-FL and Whale-

ELM, also perform poorly at detecting complex and low-rate attacks due to their limited ability to model nonlinear and multi-stage patterns. In contrast, the combination of deep feature extraction by a CNN, a Dueling-Q structure, and a False Negative-sensitive reward design resulted in the proposed model achieving the highest attack detection rate and the most stable F1-score, thereby significantly confirming its efficiency and practicality for deployment in real Fog-IoT environments.

3.7. Evaluation of system metrics in Fog-IoT environment

To evaluate the practicality of the proposed model in fog computing environments, in addition to classification metrics, three key metrics were examined: processing latency, response speed, and processing load distribution across the edge, fog, and cloud layers.

Processing latency is measured as the average time required to analyze each traffic flow. The results show that, due to the use of a CNN-style architecture and local processing in the fog layer, system latency has been significantly reduced.

Also, the model response speed was measured as the number of streams processed per second, with an increase indicating the model's ability to respond in real time to high volumes of network traffic.

Finally, to examine the distribution of processing load, the ratio of processing performed in the fog layer to the cloud layer was calculated. The results show that most processing is performed in the fog layer, and reliance on the cloud has decreased, leading to lower latency and improved overall system efficiency.

Table 8. Evaluation of the system criteria of the proposed model

Proposed method (Fog-based)	Traditional cloud-based method	Criteria
15.3	38.7	Latency (ms)
910	410	Throughput (flow/sec)
72%	25%	Processing in the fog layer(%)
28%	75%	Processing in the cloud layer(%)

According to the results in Table 8, the proposed model has significantly reduced processing latency, with the response time decreasing from 38.7 milliseconds in the cloud-based method to 12.5 milliseconds in the fog architecture. Also, the system response speed has increased by almost a factor of 2, indicating the model's real-time processing capability under high-traffic conditions.

From the perspective of processing load distribution, about 72% of processing has been performed in the fog layer, reducing dependence on the cloud and increasing system efficiency in IoT environments. These results show that the proposed model, in addition to its high accuracy, is operationally well-suited for deployment in Fog-Edge architectures.

4. Discussion and Conclusion

The present study developed and evaluated a hybrid CNN–Dueling-Q deep reinforcement learning framework for traffic anomaly detection in Fog-IoT networks. The findings generally demonstrated that integrating convolutional feature extraction, reinforcement-learning-based adaptive decision making, class-imbalance correction, and a false-negative-sensitive reward mechanism produced strong detection performance while maintaining practical computational efficiency for deployment at the fog layer. On the UNSW-NB15 dataset, the optimized proposed model

achieved an accuracy of 98.6%, attack precision of 95.3%, attack recall of 91.7%, and attack F1-score of 93.5%. The model also demonstrated strong performance on CICIDS2017, with an accuracy of 98.8%, attack precision of 95.6%, attack recall of 92.4%, and attack F1-score of 93.9%. These results are particularly important because overall accuracy alone may conceal poor performance for minority attack classes in highly imbalanced cybersecurity datasets. The simultaneously high precision, recall, and F1 values obtained in this study indicate that the proposed architecture was not merely biased toward the dominant normal class but retained substantial sensitivity to malicious traffic.

The effectiveness of the proposed model is consistent with the growing body of evidence supporting deep learning for intrusion detection in complex IoT environments. Khan et al. emphasized that deep learning can overcome important limitations of conventional intrusion detection by automatically extracting representations from heterogeneous and nonlinear IoT traffic [1]. Similarly, Heidari and Jabraeil Jamali identified deep neural architectures as increasingly important for improving IoT intrusion detection under heterogeneous traffic conditions [2]. The present findings extend this line of research by showing that deep feature extraction becomes even more useful when it is connected to an adaptive decision mechanism rather than being used solely for conventional supervised classification. The strong performance of the CNN component is also compatible with

hybrid deep-learning evidence. Halbouni et al. reported that CNN-LSTM architectures can effectively combine structural feature extraction and temporal modeling for network intrusion detection [15], while Ravi et al. showed that recurrent feature fusion can improve intelligent IDS performance by exploiting complementary deep representations [14]. The present results similarly support the importance of learning richer representations, but suggest that coupling such representations with reinforcement-learning-based policy optimization can provide an additional advantage in attack-sensitive decision making.

A major finding concerned the effect of class balancing through SMOTE. Without SMOTE, the model achieved relatively high overall accuracy, but attack recall remained only 86.3%, indicating that a notable proportion of malicious traffic was incorrectly classified as normal. When the oversampling level was increased to 50%, attack recall increased to 91.7%, while attack precision reached 95.3% and F1-score reached 93.5%. Increasing oversampling to 100% produced only a small additional increase in recall to 92.5% and was accompanied by a slight reduction in precision. These results demonstrate that moderate oversampling provided a more favorable balance between sensitivity and specificity than either no balancing or excessive synthetic oversampling. This finding is theoretically consistent with the broader IDS literature, where class imbalance has repeatedly been identified as a major source of misleadingly high accuracy and weak minority-class detection [3, 4]. It also helps explain why some conventional supervised classifiers perform well in aggregate yet remain less effective for infrequent attacks. Comparative studies of machine-learning classifiers have shown that algorithm performance is strongly influenced by dataset composition and class distribution [7, 9]. In this study, SMOTE contributed directly to reducing majority-class bias and therefore improved the model's ability to recognize attacks that otherwise might have been overlooked.

The reinforcement-learning configuration also had an important influence on performance. A Replay Buffer capacity of 50,000 produced a suitable balance between learning stability, experience diversity, computational cost, and final detection performance. A smaller buffer of 10,000 was associated with greater reward fluctuations and lower F1 performance, whereas increasing the capacity to 100,000 generated only marginal performance improvement while slowing learning. The result indicates that experience replay

was not merely an implementation detail but contributed materially to stabilizing the learning policy by reducing the temporal correlation among training experiences. Similarly, the experiments examining the discount factor showed progressive improvement as γ increased from 0.90 to 0.99. At $\gamma = 0.99$, the model achieved the most favorable combination of accuracy, precision, recall, and F1. This suggests that considering longer-term rewards was advantageous for anomaly-detection decisions, because attack behavior in network traffic may develop across successive observations rather than being completely represented by isolated flows. Such an interpretation is compatible with findings showing that sequential architectures such as RNNs and LSTMs benefit from modeling relationships across traffic observations [13, 15]. The contribution of the present model is that temporal consequences are incorporated not only through feature representation but also through the reinforcement-learning objective.

The ablation analysis provided additional evidence for the necessity of the proposed architecture. Replacing Dueling-Q with a conventional DQN reduced recall and F1, indicating that separating state value from action advantage improved decision stability. Replacing the CNN with a multilayer perceptron produced an even larger decline in performance, with accuracy falling to 97.4%, attack precision to 92.1%, recall to 87.2%, and F1 to 89.6%. This finding confirms that the convolutional feature-extraction block was important for discovering multidimensional dependencies among network characteristics. When reinforcement learning was removed and the system operated only as a supervised CNN classifier, recall decreased to 88.0%, despite relatively high overall accuracy. This result is particularly important because it supports the central premise of the study: static classification alone does not fully exploit the adaptive potential required for dynamically changing Fog-IoT traffic. The improvement obtained from the RL component is conceptually consistent with findings from adaptable and open-world detection research showing that intrusion detection systems must respond to attack distributions that evolve beyond the patterns represented during initial training [19]. It also complements unsupervised anomaly-detection research demonstrating that rigid dependence on predefined labeled attack patterns is insufficient for early detection of emerging anomalies [17].

The importance of CNN-based feature extraction is also consistent with previous research on automated and optimized intrusion detection. Xu et al. demonstrated the

usefulness of automated learning procedures for identifying effective models and representations in IoT intrusion detection [18]. Devendiran and Turukmane similarly showed that combining deep sequence learning with optimization can improve network intrusion detection accuracy and reduce false alarms [24], while Aljehane reported improved IDS performance by integrating deep learning with a Golden Jackal Optimization strategy [25]. These studies demonstrate that effective feature representation and model optimization are fundamental to IDS performance. However, the present study differs in that optimization was embedded within a policy-learning framework in which decisions were repeatedly updated according to rewards and penalties. This distinction may explain why removing reinforcement learning caused a clear decline in attack recall even though the CNN itself continued to provide strong overall classification performance.

The findings also showed that the proposed framework was appropriate for resource-constrained fog deployment. The CNN–Dueling-Q architecture achieved a processing latency of 15.3 ms, throughput of 910 flows per second, and a CPU-load-distribution standard deviation of 0.19. These results were more favorable than those obtained for Random Forest, CNN-LSTM, and Autoencoder-LSTM reference models in the system-efficiency analysis. Low latency is critical in Fog-IoT networks because a detection system that produces accurate outputs but responds too slowly may fail to prevent or contain attacks before they affect connected devices or services. The results therefore support earlier work showing the advantages of moving attack detection closer to IoT data sources. Samy et al. demonstrated that fog-based deep-learning attack detection can reduce dependence on centralized cloud processing and enable more responsive IoT security mechanisms [22]. Lawal et al. likewise proposed fog computing as a practical basis for mitigating DDoS attacks by processing traffic closer to endpoint devices [20]. Shafi et al. further showed that fog-assisted and SDN-controlled architectures can support enduring anomaly detection while improving network-resource utilization [21]. The current findings reinforce these observations by indicating that a reinforcement-learning architecture can be implemented without imposing prohibitive processing costs.

The efficiency findings also align with the broader movement toward distributed security architectures. Sinaeepourfard and Petersen emphasized the importance of distributing processing across different computational layers rather than transferring all information to centralized resources [23]. Tariq et al. similarly used fog-edge

processing to support intrusion detection in smart-grid environments where low-latency decisions are essential [28]. Rehman et al. extended this distributed perspective using fog-enabled federated learning for detecting jamming and spoofing attacks in industrial IoT systems [27]. Although federated learning and reinforcement learning solve different aspects of the security problem, these studies collectively support the principle that intrusion detection should increasingly be distributed toward computational layers closer to traffic generation. The relatively low latency and high throughput observed in the present study suggest that the proposed CNN–Dueling-Q model is compatible with this architectural direction.

Comparison with existing IDS methods further supported the effectiveness of the proposed architecture. On UNSW-NB15, the proposed model achieved 98.6% accuracy and 91.7% attack recall, outperforming Random Forest, RNN-based IDS, Autoencoder-LSTM, CNN-LSTM, Autoencoder-CNN-LSTM, federated-learning CNN, Transformer-SVM, and Whale-ELM alternatives evaluated in the study. The same pattern was observed on CICIDS2017, where the proposed model reached 98.8% accuracy, 95.6% precision, 92.4% recall, and 93.9% F1. The improvement in recall is especially meaningful because missed attacks represent one of the most consequential errors in cybersecurity applications. The results are consistent with the advantages demonstrated by deep recurrent and hybrid models in previous studies [13, 14], but they also suggest that explicitly weighting false-negative errors within the reinforcement-learning reward function can improve sensitivity beyond that achieved by conventional supervised objectives.

The performance achieved against DDoS and other complex attacks is also supported by previous IoT security research. Doshi et al. demonstrated that machine-learning-based detection can identify DDoS behavior from consumer IoT traffic, establishing the feasibility of behavioral attack recognition in resource-constrained environments [11]. Aamir and Zaidi found that clustering-based semi-supervised learning can improve DDoS attack classification while reducing dependence on completely labeled datasets [12]. More recently, Ain et al. reported strong IoT DDoS detection performance using a hybrid deep-learning approach [16]. The present findings are compatible with this trajectory but indicate that attack detection can additionally benefit from modeling classification as sequential policy learning. This is particularly relevant for low-rate or distributed attacks whose individual observations may

resemble normal traffic but whose accumulated behavioral pattern is malicious.

The proposed method also compares conceptually with optimization- and feature-selection-based IDS approaches. Turukmane and Devendiran showed that feature-selection-assisted multi-SVM detection can improve efficiency by removing redundant information [10], while Sahu et al. demonstrated the value of optimization combined with extreme learning machines in fog-based intrusion detection [26]. Tawfik similarly showed that ensemble learning combined with advanced feature selection can improve intrusion detection in IoT and fog computing environments [29]. These findings support the present study's emphasis on reducing unnecessary processing and maximizing informative representations. Nevertheless, the ablation findings suggest that feature extraction alone was insufficient: the full model performed better than the CNN-only version, indicating that the RL decision layer contributed independent value after discriminative features had already been learned.

Finally, the model's performance is consistent with emerging research emphasizing scalability in IIoT security. Qin et al. proposed a multi-agent LSTM framework to address anomaly detection across distributed IIoT environments, highlighting the importance of scalable processing and temporal behavior modeling [30]. The present framework approaches scalability from a different direction by maintaining a relatively lightweight CNN structure and a restricted action space suitable for fog-node deployment. Together, these findings suggest that the next generation of IoT intrusion detection is likely to require combinations of efficient representation learning, distributed deployment, adaptive decision policies, and explicit consideration of security-specific error costs. Overall, the results of the present study support the conclusion that the CNN–Dueling-Q architecture provides a coherent solution to several limitations that remain when deep learning, optimization, or fog deployment are implemented separately.

This study has several limitations that should be considered when interpreting the findings. First, the experiments were conducted using the UNSW-NB15 and CICIDS2017 benchmark datasets, which provide diverse attack scenarios but cannot fully reproduce the continuously changing traffic, device heterogeneity, protocol diversity, and adversarial behavior found in operational Fog-IoT deployments. Second, the reinforcement-learning environment was constructed from labeled traffic

observations and a discrete binary action space, whereas real-world security decisions may require multi-class attack identification, graded risk assessment, or coordinated responses across several fog nodes. Third, although system efficiency was evaluated through latency, throughput, and CPU-load distribution, deployment on a wider range of resource-constrained physical fog devices was not examined. The use of SMOTE may also create synthetic minority-class observations that do not perfectly reproduce the distribution of rare attacks. Finally, the performance of the model under severe concept drift, adversarial manipulation, encrypted traffic, previously unseen communication protocols, and long-term continuous operation remains to be established.

Future studies should evaluate the proposed framework in real or large-scale emulated Fog-IoT testbeds containing heterogeneous sensors, gateways, industrial devices, and dynamically changing attack scenarios. The current binary decision framework could be extended toward multi-class or hierarchical attack recognition so that the reinforcement-learning agent identifies specific attack families and selects corresponding response actions. Future models could also integrate recurrent, attention-based, or transformer components to represent longer temporal dependencies while preserving the lightweight characteristics required for fog deployment. Alternative imbalance-handling techniques, cost-sensitive learning, prioritized replay, and adaptive reward functions should be compared with the current SMOTE-based approach. Another valuable direction would be integrating federated reinforcement learning across multiple fog nodes to support distributed learning without centralizing sensitive traffic data. Studies should additionally investigate robustness against adversarial examples, concept drift, encrypted traffic, zero-day attacks, and poisoning of reinforcement-learning experiences. Finally, longer-term evaluations involving energy consumption, memory footprint, communication overhead, model-update cost, and resilience under node failures would provide a more comprehensive assessment of practical scalability.

For practical deployment, network administrators and IoT security designers should consider positioning lightweight anomaly-detection modules at fog nodes so that suspicious traffic can be identified before unnecessary data are transmitted to centralized cloud infrastructure. Implementation should prioritize attack recall and false-negative reduction rather than relying on overall accuracy as the principal security metric. Training data should be

monitored for class imbalance, and balancing procedures should be applied only to training data to avoid information leakage and unrealistic evaluation results. Replay-memory capacity, reward penalties, and decision thresholds should be tuned according to the computational resources and security requirements of each deployment environment. Organizations should also maintain continuous logging and periodic model updating so that changing traffic distributions can be reflected in the detection policy. In operational systems, the proposed model would be most effectively used as part of a layered security architecture in which high-risk detections trigger additional verification, traffic isolation, rate limiting, or escalation to centralized security services rather than relying on a single automated detection decision.

Authors' Contributions

Authors equally contributed to this article.

Acknowledgments

Authors thank all participants who participate in this study.

Declaration of Interest

The authors report no conflict of interest.

Funding

According to the authors, this article has no financial support.

Ethical Considerations

All procedures performed in this study were under the ethical standards.

References

- [1] A. R. Khan, M. Kashif, R. H. Jhaveri, R. Raut, T. Saba, and S. A. Bahaj, "Deep Learning for Intrusion Detection and Security of Internet of Things (IoT): Current Analysis, Challenges, and Possible Solutions," *Security and Communication Networks*, vol. 2022, no. 1, p. 4016073, 2022, doi: 10.1155/2022/4016073.
- [2] A. Heidari and M. A. Jabraeil Jamali, "Internet of Things Intrusion Detection Systems: A Comprehensive Review and Future Directions," *Cluster Computing*, vol. 26, no. 6, pp. 3753-3780, 2023, doi: 10.1007/s10586-022-03776-z.
- [3] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1-22, 2019, doi: 10.1186/s42400-019-0038-7.
- [4] A. Thakkar and R. Lohiya, "A Survey on Intrusion Detection System: Feature Selection, Model, Performance Measures, Application Perspective, Challenges, and Future Research Directions," *Artificial Intelligence Review*, vol. 55, no. 1, pp. 453-563, 2022, doi: 10.1007/s10462-021-10037-9.
- [5] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015: IEEE, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942.
- [6] D. Stiawan, M. Y. B. Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis with Information Gain for Anomaly Detection," *IEEE Access*, vol. 8, pp. 132911-132921, 2020, doi: 10.1109/ACCESS.2020.3009843.
- [7] A. A. Salih and A. M. Abdulazeez, "Evaluation of Classification Algorithms for Intrusion Detection System: A Review," *Journal of Soft Computing and Data Mining*, vol. 2, no. 1, pp. 31-40, 2021.
- [8] Z. Azam, M. M. Islam, and M. N. Huda, "Comparative Analysis of Intrusion Detection Systems and Machine Learning Based Model Analysis Through Decision Tree," *IEEE Access*, 2023, doi: 10.1109/ACCESS.2023.3296444.
- [9] N. Saran and N. Kesswani, "A Comparative Study of Supervised Machine Learning Classifiers for Intrusion Detection in Internet of Things," *Procedia Computer Science*, vol. 218, pp. 2049-2057, 2023, doi: 10.1016/j.procs.2023.01.181.
- [10] A. V. Turukmane and R. Devendiran, "M-MultiSVM: An Efficient Feature Selection Assisted Network Intrusion Detection System Using Machine Learning," *Computers & Security*, vol. 137, p. 103587, 2024, doi: 10.1016/j.cose.2023.103587.
- [11] R. Doshi, N. Apthorpe, and N. Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018: IEEE, pp. 29-35, doi: 10.1109/SPW.2018.00013.
- [12] M. Aamir and S. M. A. Zaidi, "Clustering Based Semi-Supervised Machine Learning for DDoS Attack Classification," *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 4, pp. 436-446, 2021, doi: 10.1016/j.jksuci.2019.02.003.
- [13] S. M. Kasongo, "A Deep Learning Technique for Intrusion Detection System Using a Recurrent Neural Networks Based Framework," *Computer Communications*, vol. 199, pp. 113-125, 2023, doi: 10.1016/j.comcom.2022.12.010.
- [14] V. Ravi, R. Chaganti, and M. Alazab, "Recurrent Deep Learning-Based Feature Fusion Ensemble Meta-Classifer Approach for Intelligent Network Intrusion Detection System," *Computers & Electrical Engineering*, vol. 102, p. 108156, 2022, doi: 10.1016/j.compeleceng.2022.108156.
- [15] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System," *IEEE Access*, vol. 10, pp. 99837-99849, 2022, doi: 10.1109/ACCESS.2022.3206425.
- [16] N. U. Ain, M. Sardaraz, M. Tahir, M. W. Abo Eloud, and A. Alourani, "Securing IoT Networks Against DDoS Attacks: A Hybrid Deep Learning Approach," *Sensors*, vol. 25, no. 5, p. 1346, 2025, doi: 10.3390/s25051346.
- [17] R. H. Hwang, M. C. Peng, C. W. Huang, P. C. Lin, and V. L. Nguyen, "An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection," *IEEE Access*, vol. 8, pp. 30387-30399, 2020, doi: 10.1109/ACCESS.2020.2973023.
- [18] H. Xu, Z. Sun, Y. Cao, and H. Bilal, "A Data-Driven Approach for Intrusion and Anomaly Detection Using

- Automated Machine Learning for the Internet of Things," *Soft Computing*, pp. 1-13, 2023, doi: 10.1007/s00500-023-09037-4.
- [19] M. Soltani, B. Ousat, M. J. Siavoshani, and A. H. Jahangir, "An Adaptable Deep Learning-Based Intrusion Detection System to Zero-Day Attacks," *Journal of Information Security and Applications*, vol. 76, p. 103516, 2023, doi: 10.1016/j.jisa.2023.103516.
 - [20] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "A DDoS Attack Mitigation Framework for IoT Networks Using Fog Computing," *Procedia Computer Science*, vol. 182, pp. 13-20, 2021, doi: 10.1016/j.procs.2021.02.003.
 - [21] Q. Shafi, A. Basit, S. Qaisar, A. Koay, and I. Welch, "Fog-Assisted SDN Controlled Framework for Enduring Anomaly Detection in an IoT Network," *IEEE Access*, vol. 6, pp. 73713-73723, 2018, doi: 10.1109/ACCESS.2018.2884293.
 - [22] A. Samy, H. Yu, and H. Zhang, "Fog-Based Attack Detection Framework for Internet of Things Using Deep Learning," *IEEE Access*, vol. 8, pp. 74571-74585, 2020, doi: 10.1109/ACCESS.2020.2988854.
 - [23] J. K. A. Sinaeepourfard and S. A. Petersen, "A Distributed-to-Centralized Smart Technology Management (D2C-STM) Model for Smart Cities: A Use Case in the Zero Emission Neighborhoods," in *Fifth IEEE Annual International Smart Cities Conference (ISC2 2019)*, Casablanca, Morocco, 2019, doi: 10.1109/ISC246665.2019.9071762.
 - [24] R. Devendiran and A. V. Turukmane, "Dugat-LSTM: Deep Learning Based Network Intrusion Detection System Using Chaotic Optimization Strategy," *Expert Systems with Applications*, vol. 245, p. 123027, 2024, doi: 10.1016/j.eswa.2023.123027.
 - [25] N. O. Aljehane, "Golden Jackal Optimization Algorithm with Deep Learning Assisted Intrusion Detection System for Network Security," *Alexandria Engineering Journal*, vol. 86, pp. 415-424, 2024, doi: 10.1016/j.aej.2023.11.078.
 - [26] D. P. Sahu, B. Tripathy, and L. Samantaray, "Optimized Intrusion Detection System in Fog Computing Environment Using Automatic Termination-Based Whale Optimization with ELM," *International Journal of Computer Network and Information Security*, vol. 16, no. 2, pp. 79-91, 2024, doi: 10.5815/ijcnis.2024.02.07.
 - [27] T. Rehman, N. Tariq, F. A. Khan, and S. U. Rehman, "FFL-IDS: A FOG-Enabled Federated Learning-Based Intrusion Detection System to Counter Jamming and Spoofing Attacks for the Industrial Internet of Things," *Sensors*, vol. 25, no. 1, p. 10, 2024, doi: 10.3390/s25010010.
 - [28] N. Tariq, A. Alsirhani, M. Humayun, F. Alserhani, and M. Shaheen, "A Fog-Edge-Enabled Intrusion Detection System for Smart Grids," *Journal of Cloud Computing*, vol. 13, no. 1, p. 43, 2024, doi: 10.1186/s13677-024-00609-9.
 - [29] M. Tawfik, "Optimized Intrusion Detection in IoT and Fog Computing Using Ensemble Learning and Advanced Feature Selection," *PLOS ONE*, vol. 19, no. 8, p. e0304082, 2024, doi: 10.1371/journal.pone.0304082.
 - [30] Z. Qin, Q. Luo, X. Nong, X. Chen, H. Zhang, and C. U. I. Wong, "MAS-LSTM: A Multi-Agent LSTM-Based Approach for Scalable Anomaly Detection in IIoT Networks," *Processes*, vol. 13, no. 3, p. 753, 2025, doi: 10.3390/pr13030753.