Discover Applied Sciences

https://doi.org/10.1007/s42452-026-08563-8

Article in Press

DDoS attack detection and classification for the MQTT-IoT protocol using LSTM models

Received: 4 May 2025

Accepted: 13 March 2026

Published online: 24 March 2026

Cite this article as: Negesse D.B., Gemeda K.A. & Gianini G. DDoS attack detection and classification for the MQTT-IoT protocol using LSTM models. *Discov Appl Sci* (2026). https://doi.org/10.1007/s42452-026-08563-8

Deribew Beko Negesse, Ketema Adere Gemeda & Gabriele Gianini

DDOS ATTACK DETECTION AND CLASSIFICATION FOR THE MQTT-IOT PROTOCOL USING LSTM MODELS

Deribew Beko Negesse[1], Ketema Adere Gemeda[*1], and Gabriele Gianini[2]

[1]Department of Computer Science and Engineering, School of Electrical Engineering and Computing, Center for Electrical System and Electronics, Adama Science and Technology University, Adama, Ethiopia

[2]Department of Informatics, Systems and Communication (DISCo) University of Milano-Bicocca, 20126, Milano (MI), Italy

deribest23@gmail.com;ketema.adere@astu.edu.et; gabriele.gianini@unimib.it
Corresponding Author: *ketema.adere@astu.edu.et

ABSTRACT: The Message Queue Telemetry Transport (MQTT) protocol serves as a vital publish-subscribe messaging standard, enabling seamless communication across critical Internet of Things (IoT) infrastructures. However, the widespread adoption of MQTT has heightened vulnerability to cybersecurity threats, notably to Distributed Denial of Service (DDoS) attacks. These attacks overwhelm MQTT brokers with malicious traffic, leading to service disruptions. In this study, we developed a deep learning model to detect DDoS attacks within MQTT-IoT networks, comparing several candidate architectures: Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Multi-Layer Perceptron (MLP). Model evaluation utilized a publicly available, real-world MQTT dataset containing both DDoS attacks and normal traffic. The experiment result illustrated that our proposed LSTM attained 99.53% F1-score, outperforming the best models from the literature. This aligns with the observation that MQTT-based attacks are primarily sequential anomalies, where the spatial structure has a lower importance, and where the LSTM can take advantage of its ability to model temporal attack signatures.

Keywords: IoT, MIoT, MQTT protocol, DDoS attacks, MQTTset, MLP, RNN, LSTM

1. INTRODUCTION

IoTs have transformed the communication technology landscape enabling seamless device and sensor connectivity to the World Wide Web (WWW) with minimal human intervention [1], [2]. This interconnectivity allows IoT nodes, or smart devices, to gather, process, and share data, paving the way for promising future communication advancements. Analysts estimate that the number of connected IoT devices will rapidly increase. Sensors drive this revolution, collecting real-time environmental data, such as humidity, temperature, light, and motion across critical infrastructures. Consequently, IoT applications have become integral to various aspects of life, including smart cities, healthcare, and agriculture [3], [4], [5], [6]. The three main components of IoT are the perception, network, and application layers, that forming a network of connected items providing services to customers via sensors, actuators, and gateways. As device interconnectivity increases, data transfer speeds accelerate. Due to the resource constraints and limited computing capabilities of most IoT devices, lightweight and reliable communication protocols are essential, Popular options include MQTT, Constrained Application Protocol (CoAP), and Advanced Message Queue Protocol (AMQP) [2], [3]. MQTT is the most widely adopted protocol in the IoT environment due to its lightweight nature, low bandwidth requirements, asynchronous operation, minimal memory usage, and reduced packet loss [8], [9]. MQTT operates over Transmission Control Protocol/Internet Protocol (TCP/IP), using a

publish/subscribe message communication pattern. The architecture of MQTT comprises three primary components: publisher, subscribers, and brokers, as depicted in Figure 1 [8].

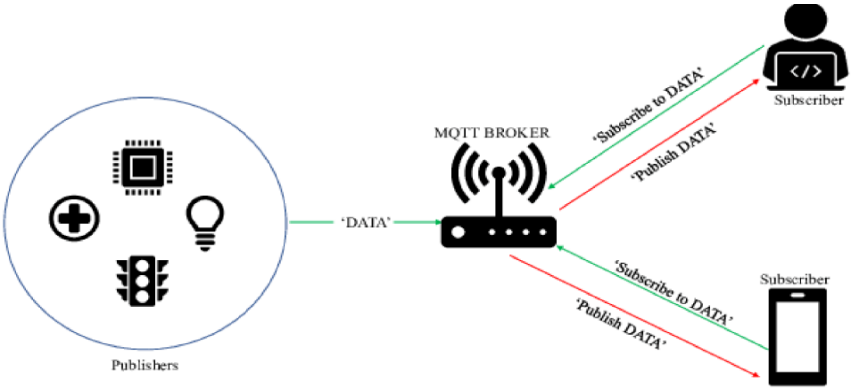

Figure 1: MQTT architecture

Data/topic is a string stored on the central MQTT broker, using a forward slash (/) delimiter for message addressing. The MQTT broker efficiently disseminates data between physical IoT devices, such as sensors and actuators. Typically, sensors convert physical attributes into electrical signals before publishing, while actuators/subscribers convert electrical signals into physical actions understandable by the end-user. This mechanism ensures that clients receive only relevant messages, minimizing network traffic and resource consumption, which is critical given the limited resources (computing capability, memory storage, and energy supply) of small sensor devices [10], [11], [12]. Communication is achieved through the exchange of control packets, each containing a standardized two-byte protocol header [13]. This pattern allows a client to write data to topics, and other devices can subscribe to that topic via a broker to receive updates and facilitate further processing. This reliance on a central broker with limited storage capability creates a weak point for security risks. DDoS attackers exploit this by generating large-sized message packets with high Quality of Service (QoS) levels, overwhelming the broker and causing service unavailability. The target system becomes incapacitated, communication is disrupted, and system queues fill with forged packets, depleting resources and blocking new incoming requests [13].

The MQTT threat model includes various attack vectors targeting MQTT protocols, such as DDoS, identity spoofing, information disclosure, privilege escalation, and data tampering [4]. DDoS attacks are one of the most common types of cyberattacks that can disrupt services to compromise MQTT network security [4], [5]. These attackers aim to process malicious packets containing an unexpectedly larger payload than the normal behavior with a high QoS level to a centralized MQTT broker. This causes the target to be incapacitated or forced out of service, disrupting communication between IoT devices and filling the system queues with forged or bogus packets, which depletes system resources and renders the broker incapable of handling new incoming requests. Ensuring scalable security measures, particularly effective detection mechanisms, is crucial for securing MQTT networks against DDoS attacks in order to effectively utilize IoT applications.

Traditional intrusion detection systems (IDS), such as rule-based and signature-based have been considered possibly in order to establish secure communication systems [14], [15]. They are useful in small network data scenarios and their usage is more viable in well-known DDoS trend attacks. The fuzzy rule works on the strategy of the IF-THEN statement approach to identify DDoS flows based on the state of the rule base. The primary limitation of such a traditional approach is that

they can only effectively identify known assaults. This necessitates frequent updates and makes pattern learning difficult. They are also computationally expensive, making them unsuitable for resource-constrained devices. Anomaly-based is also another technique used for the detection of DDoS attacks by establishing a baseline to identify anomalous network traffic patterns [16]. Processed tree proposed using the principle of tree graph structure through observing the similarity of packet sequences in order to observe network events to identify DDoS attacks [20]. These techniques are susceptible to noise and can generate false alarms due to ever-changing network environments, particularly in dynamic and heterogeneous IoTs-based systems. False rate rises as a result of dynamic network behavior and scalability scenarios, thus it is effectively identified through integrated popular technology trends such as machine learning [15]. The system aims to learn a pattern of normal behavior and attacks behavior based on the network packet patterns. As MQTT-IoT scales increase, conventional linear machine learning methods struggle to detect evolving threats in real environments due to the volume and heterogeneity of network data. Recently developed deep learning based techniques have the capability of dealing with massive volumes of data and identifying any indications of network penetration [17], [18], [19]. Deep learning, a subset of machine learning, can understand complex patterns and process large network data to identify anomalies that deviate from normal behaviors.

Based on the limitations of existing techniques, challenges remain in detecting and classifying DDoS attacks within MQTT-IoT ecosystems, such as publish-subscribe protocol and minimal packet header) [41], [42], [43]. Deep learning-based models offer a more viable solution by autonomously learning and identifying attack patterns. However, designing a high-performance deep learning model that generalizes well across diverse attack scenarios remains an open challenge. Many existing approaches do not integrate systematic feature selection, leading to suboptimal detection accuracy and increased computational overhead. Integrating an efficient feature selection mechanism into deep learning models can enhance detection accuracy, reduce misclassification errors, and decrease the time required for attack detection. Several prior studies evaluate their models using simulated DDoS attacks in controlled MQTT-IoT environments. However, real-world network behavior is highly dynamic, and models trained on synthetic data may not perform effectively in practical deployments. Thus, validating detection systems on real-world datasets is critical to ensuring their reliability and robustness. Furthermore, less attention has given for challenges raised by the concept drift when the statistical properties of the target attackers change in unforeseen ways, recently incremental on-line learning has investigated [7]. Addressing the above stated challenges will enhance the accuracy, efficiency, and practical applicability of MQTT-IoT ecosystems. The current work proposed deep learning models with feature selection mechanisms that can help to advance the security level of IoT applications. The developed model evaluates the feature relevance based on statistical properties before training and testing since effective and efficient techniques for protecting networks against these types of attacks. The increasing research importance and trends of shifting deep learning inference from the cloud to resource-constrained edge hardware devices are facing challenges due to model complexity, hyperparameter tuning, performance, maintainability, and resource utilization [12].

Accordingly, the main contribution of the study is the development and comparative analysis of suitable deep learning models: RNN, LSTM, and MLP to enhance DDoS attack detection. Then, we accompanied Pearson correlation and chi-Square (X^2) feature selection methods in order to comprehensively evaluate feature relevance by measuring (eg., computing weighted score) both linear

relationships and statistical dependence of each feature with the target variable. We have introduced novel MQTT flow features such as Mqtt.msgid, Mqtt.qos, and etc. which have exhibit strong statistical ties to the target (DDoS attacks) for optimizing model precision and reducing computational overhead through targeted feature selection. As a result, among those deep learning models, the LSTM enhanced DoS attack detection accuracy while reducing classification errors toward ensuring reliable threat identification for MQTT-IoT networks. Finally, we propose an incremental learning framework integrated with the established LSTM model to enable adaptive learning from evolving data streams. Its implementation is reserved for future work.

The remainder of this manuscript is organized as follows: Section 2 reviews related works from the literature. Section 3 details the research methodology and materials used. The proposed architecture for DDoS attack detection and classification is presented in Section 4, followed by its performance evaluation and discussion in Section 5. Finally, Section 6 concludes the paper.

2. LITERATURE REVIEW

The MQTT protocol, widely used in IoT applications, remains vulnerable to various internal and external actors [4], [5]. Among these, DDoS attacks are especially disruptive, aiming to exhaust network resources through employing tactics such as bandwidth saturation, malicious payload injection, and abuse of QoS mechanisms. These attacks often leverage large numbers of compromised devices to overwhelm MQTT brokers with high-volume traffic. Due to the diversity of adversarial behaviors, continued research is vital in this regard. Numerous studies have explored DDoS detection and classification in MQTT networks using a variety of algorithmic methods [8], [14]. As IoT advances toward industrial-scale adoption [42], the lack of proactive security measures in protocol design, especially the MQTT protocol, poses significant risks. Exploitation of these vulnerabilities raises critical concerns about the growing impact of cyberattacks on human well-being in our increasingly connected world [21].

Alzahrani and Aldhyani, [8], leveraged AI models to identify and categorize MQTT protocol IoT attacks. They investigated the use of K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), CNN, and CNN-LSTMs for this purpose. The study achieved high accuracy, up to 98.94% using a train-test (0.3) split and 93.22% with 5-fold cross-validation (CV) on the CNN-LSTM model. However, it has been noted that the lack of a feature selection mechanism has increased training time, decreased model robustness, and led to increased misclassification errors. In the end, very limited theoretical insights were provided.

Haripriya and Kulothungan., [14] developed a fuzzy-rule approach to identify DDoS attacks from normal ones in MQTT-IoT network flows. The authors presented an MQTT detection system that analyzes publish-subscribe node behavior and network features, focusing on connect and connack control packets. The key advantage is the periodic updating of fuzzy rules, which boosts intrusion detection precision to around 90%. However, it has been noted that considering only connection request rates is insufficient for detection, as DDoS attacks can manifest in various characteristics, including overwhelming the broker with publish messages or exploiting resources to disrupt communication, even at high QoS levels. To develop a stronger detection mechanism, it has been suggested to analyze additional MQTT packet headers and payload data beyond the limited control packet features (connect and connack). Such analysis can account for the variable nature of DoS attacks and provide a more comprehensive approach to securing MQTT-based IoT applications.

Process-tree-based developed to detect DDoS attacks in MQTT-IoT systems with the MQTT protocol [20]. Such an approach partially checks the MQTT network behavior to identify small deviations in the log file, achieving an 83.8% detection rate. While efficient in reducing network overhead, this partial observation can miss attack details and increase false positives. To fully understand the attack scope, analyzing the complete MQTT network behavior and selecting significant parameters may be necessary to improve the detection method.

A Deep Neural Network (DNN)-based intrusion detection system for MQTT protocols has been proposed to investigate publish-subscribe security vulnerabilities [23]. By using the MQTT-IoT-IDS2020 dataset, the authors achieved 97.13% detection accuracy through the comparative evaluation of multiple machine learning approaches. While demonstrating strong performance against the examined threat vectors, the work acknowledges a critical limitation: the evaluation environment excluded DoS attack patterns, among the most prevalent and high-impact threats in MQTT-IoT ecosystems. Such gaps are explicitly identified as a priority for future research, emphasizing the need to expand testing to include DoS and other advanced attack scenarios. Such enhancements would validate the system's robustness for safeguarding real-world MQTT deployments against comprehensive security threats. The study in [24] focused on a similar dataset, the MQTT-IoT-IDS2020 dataset, in order to explore the use of machine learning for detecting intrusions in IoT networks that use the MQTT protocol. In this, the article investigated the application of different machine learning techniques to classify normal and malicious MQTT traffic (aiming to identify attacks like brute-force and denial-of-service).

Syed et al. [25] studied attack detection through machine learning for IoT and modeled algorithms to identify and classify DDoS attacks. The paper discusses messages published/subscribed to specific topics via valid authorization and authentication mechanisms, as well as Connect Flooding and Invalid Subscription Flooding attacks, and assesses the performance of brokers. The study achieved a high attack detection accuracy of around 95.94% and a training time of 847.41 seconds using an MLP model. However, it has been noted that the detection accuracy and training time still need further improvement, considering resource constraints on IoT devices and potential delays in detection. In such a case, it might be referred to as a lag between when an attack is launched and when it is detected by the system. This delay could allow an attack to inflict more damage before being mitigated. The paragraph suggests this is an area that needs improvement in the proposed approach. Additionally, the use of an MQTT dataset led to evaluating the model's performance, which was not balanced and resulted in biased results due to the frequently changing nature of attack behaviors.

Dikii and Tikhomirov, [26] studied the identification of DoS attacks that exploit the MQTT protocol's subscribe messages. They focused on using machine learning algorithms such as decision tree (DT) and support vector machine (SVM) to detect anomalous behavior in MQTT-IoT networks, particularly related to subscribe messages. However, the authors acknowledge that it may not address other potential attack vectors or vulnerabilities within the MQTT protocol or IoT networks, such as those targeting the published messages. Publisher message-based attacks could involve malicious activities aimed at overwhelming the MQTT broker by flooding the network with unauthorized or spurious publisher messages, potentially disrupting the normal operation of the IoT network and compromising its availability and performance. The researcher used a simulated dataset to evaluate the model performance, and the authors noted that due to the variance in specific IoT networks and attack types, the real-world dataset is more viable for model evaluations. Mihoub et al. [27] studied the detection and mitigation of DoS attacks for IoT using a looking-back-enabled ML approach to identify attack

scenarios using Bot-IoT datasets with various protocols, including Hyper Text Transfer Protocol (HTTP), Transport Control Protocol (TCP), and User Datagram Protocol (UDP). However, the demonstrated approach was not based on MQTT protocol analysis, which limits the details on protocol-specific parameters, and thus may not be effective for detecting MQTT-based attacks that are the focus of the current research.

Imran et al. [28] proposed a feature engineering approach for anomaly detection, focusing on DoS attacks in MQTT-IoT networks using machine learning. The authors suggested that considering the MQTT network control packet flow characteristics would be more effective. However, their approach relied on socket features, such as source addresses, which may not be compatible with dynamic network environments, as Internet Protocol (IP) addresses can vary. This could increase the potential for misclassification and model bias. Instead, the authors suggest that considering the MQTT network control packet flow characteristics would be more effective.

Mosaiyebzadeh et al [49] developed an IDS for MQTT attack detection using three deep learning models (DNN, LSTM, and CNN-RNN-LSTM). While the hybrid CNN-RNN-LSTM model achieved impressive results (92.04% accuracy, 100% recall, and 98.33% F1-score) on the MQTT-IoT-IDS2020 dataset, the research lacks transparency regarding feature selection methodology and presenting a notable gap in the experimental design. Due its complexity, this architecture likely requires cloud or edge server deployment for practical implication which is not possible on the IoT device itself (i.e., creating network latency and potential single points of failure). Security challenges of RPL (Routing Protocol for Low-Power and Lossy Networks) with regard to 6LoWPAN networks (stand for IPv6 over Low-Power Wireless Personal Area Networks) which forms the backbone of many IoT systems studied in [50]. Security framework has designed to protect vulnerable IoT networks from multiple routing attacks by leveraging distributed training (Federated Learning) to obtain promising results. A recent study [51] developed multi-class IDS for IoT-DDoS attacks using CNN, DNN, and Transformer models on the CIC-IoT 2023 dataset. The models achieved high accuracy (~99% binary, ~100% 3-class, and ~93% 12-class), with DNNs performing best on complex tasks. However, the research's relevance is limited for MQTT-based systems, as it does not evaluate MQTT-specific threats or datasets.

Furthermore, the challenge of distinguishing malicious from legitimate traffic is compounded when they share network-level identifiers. Elsayed et al. [29] highlighted this issue by noting that intruders and legitimate users can share the same IP address, which complicates accurate classification. An analysis of the literature reveals that this remains a persistent limitation across many existing models and approaches, which often struggle to disentangle such overlapping behaviors. The challenge of concept drift, where a model's target domain changes in unforeseen ways, is often addressed through incremental online learning [52]. A comprehensive study in this area has analyzed and compared various algorithms based on accuracy, convergence speed, and model complexity. Building upon this foundation, recent work has explored these concepts in the context of IoT security [7], [53]. A modified version of Long Short-Term Memory (LSTM) called the LSTM Drift Detector (LSTMDD) is proposed, where it outperforms the other existing drift detection techniques detectors in detecting both sudden and gradual drifts [7]. Although the later study [53] is limited to image data; constraint given that IoT often involves time-series or sensor data, it makes a significant contribution by highlighting a critical vulnerability in incremental learning systems.

In order to ensure reliable communication between IoT devices, the targeted DoS attacks were studied using the UNSW-NB15 dataset in [31]. The study in [45] used

MQTT traffic to train classifiers like logistic regression, decision trees, and deep neural networks, specifically focused on DDoS attacks. Identifying and classifying DoS attacks against MQTT sensor networks has also been studied in [46]. By using MQTT datasets, only three types of attacks, including DoS, Brute force, and Malformed, were studied [47]. In [48], MQTT traffic is considered for detecting DoS and brute force attacks using feature engineering and an ensemble learning algorithm to enhance and identify malicious activities. Despite significant progress in IoT security research, still several gaps remain to be opened for the researchers. Existing intrusion detection systems often struggle to identify rare or less frequent attack types and limiting their effectiveness in real deployments due to computationally heavy (not lightweight for IoT) [54]. Class imbalance within datasets such as CICIoT2023 was studied for minority attack categories and highlighting the need for improved data preprocessing and balancing strategies [55]. Conversely, the advent of quantum computing introduces new threats to IoT infrastructures, as highlighted in [56], which evaluates the vulnerabilities of IoT cryptographic protocols within the evolving attack landscape and proposes corresponding countermeasures.

The quality of the utilized datasets for the model training, validation, and testing process determines the capability of the intended solution. The collection of such datasets can be created either from scratch or collected from existing sources, whereas publicly available data sources are widely acceptable considering their standardization and reproducibility. Many of the currently developed detection systems only examine partial MQTT network flow behaviors, which can result in missed attacks or increased false positives. A partial observation may incorrectly flag legitimate activities as malicious, leading to unnecessary resource exhaustion. Therefore, a thorough analysis of MQTT features to identify the most relevant parameters for attack detection is essential to improve detection efficacy [41], [42]. Table 1 summarizes some of the key gaps and limitations identified from the prior proposed schemes in terms of objective, methods (i.e., model architectures, feature selection techniques, detection approaches), and experimental activities (i.e., datasets used and evaluation metrics).

Furthermore, several publicly available datasets, such as Knowledge Discovery and Data Mining Cup competition 99 (KDDCU99) [30], University of New South Wales - Network Based-15 (UNSW-NB15) [31], and Network-Based Internet of Things (N_BaIoT) [32], are commonly used to evaluate attack detection models in conventional ICT networks. However, these datasets lack IoT-specific MQTT attack scenarios. For instance, in our recent work, we have used CIC-DDoS2019 and ICICDDoS2017 public datasets in order to early detect and classify DDoS attacks in a multi-controller structure of Software Defined Networks (SDNs) [33]. These are datasets commonly used to evaluate attack detection models in conventional networks; they lack IoT-specific MQTT attack scenarios. While datasets like BoT_IoT and ToN_IoT include IoT-based DoS and DDoS attacks, they also omit MQTT protocol-specific attacks that their applications limit [4], [5]. The MQTT-IoT-IDS2020 dataset covers MQTT-related attacks, namely aggressive scanning, UDP scanning, MQTT brute-force, and SSH brute-force, but limited scenarios are included with regard to specific DoS attacks [23], [24]. MQTTset dataset provides comprehensive MQTT-specific attack scenarios, including DoS, brute force, SlowITe, and malformed data attacks [34], [35]. Table 2 compares the effectiveness of existing datasets (N_BaIoT, BoT_IoT, ToN_IoT, MQTT-IoT-IDS2020, and MQTTset) for MQTT-based DDoS attacks, considering three key criteria [4], [5], [23], [24], [31] – [35]. (i) Dataset relevance to MQTT-specific DDoS attacks: protocol focus - whether the dataset is tailored to MQTT, MQTT-based DDoS inclusion - explicit inclusion of MQTT-specific attacks, and attack diversity - coverage of different DDoS attack types; (ii) realism and data quality: traffic source

- origin of the data (e.g., simulated or real-world), labeling accuracy - reliability of ground-truth labels, and normal/attack ratio - balance between benign and malicious traffic; and (iii) feature suitability for deep learning: packet-level features - availability of raw packet details, flow-based features - presence of aggregated flow statistics, and preprocessing needed - level of required data preparation. As one can observe, MQTTset outperforms the others for MQTT-targeted DDoS studies, due to its protocol specialization, attack diversity, and reproducibility when compared to others, including the MQTT-IoT-IDS2020 (limited variants). This dataset offers rich MQTT feature parameters tailored to our research goals, as detailed in section 3.1

Table 1: Comparison of previously proposed schemes in terms of objective, experimental activities, and gaps identified.

Authors, Year, and Ref.	Objective	Method	Experimental	Gap identified
Dikii and Tikhomirov., [26]	– Detect DoS attacks in IoT networks exploiting MQTT SUBSCRIBE messages using ML classifiers.	– **Model architectures:** MLP, RF, and SVM (linear/RBF). – **Feature selection:** time interval, message count, and mean time between messages. – **Detection approach:** anomaly-based (behavioral analysis).	– **Datasets:** custom-generated (10K attacks + 5K legitimate logs). – **Evaluation metrics:** accuracy and F1-score.	– Small-scale environment, no comparison with public datasets (e.g., MQTTset). – Unbalanced dataset. – Limited to SUBSCRIBE messages; other MQTT attacks not explored.
Syed et al., 2020 [25]	– To propose a machine learning-based framework for detecting application-layer DoS attacks targeting MQTT brokers in IoT networks.	– **Model architectures:** AODE, C4.5, and MLP. – **Feature selection:** MQTT protocol metadata (e.g., packet counts, and field lengths). – **Detection approach:** anomaly-based (ML classifiers).	– **Datasets:** custom-generated (real IoT testbed with MQTT traffic). – **Metrics:** accuracy (95.94%,MLP)	– Limited to single-source attacks (no DDoS evaluation). – Class imbalance issues. – Consume large training (late detection took around 842.1 seconds). – Accuracy further refined or improved.
Alzahrani and Aldhyani., 2022 [8]	– Develop an intrusion detection system (IDS) for MQTT-based IoT networks using AI algorithms.	– **Model architectures:** CNN (64/128 filters), LSTM (memory gates), KNN, LDA, and hybrid CNN-LSTM. – **Feature selection:** normalization and one-hot encoding. – **Detection approach:** anomaly-based (focus on MQTT-specific attacks).	– **Datasets:** MQTTset (330,936 samples, and 5 attack types). – **Evaluation metrics:** accuracy, precision recall, F1-score, MAE, and MSE.	– Without an appropriate features selection mechanism, it leads to increased training time, misclassification errors, late detection, and decreased model performance. – Hybrid CNN-LSTM has high computational cost (eg., training time 130.30s and memory requirements 16GB RAM); may hinder deployment on edge devices due to their resource constraint characteristics – Hybrid CNN-LSTM is empirically
Imran et al., 2024 [28]	– Enhance anomaly detection in MQTT-IoT networks by incorporating the 'source' attribute (IP addresses) from PCAP files and improving detection accuracy using feature engineering and decision tree variants.	– **Model architectures:** Decision Tree variants (ID3, C4.5, CART, XGBoost, LightGBM, and CatBoost). – **Feature selection:** hand-crafted feature engineering (hexadecimal conversion, label encoding, handling missing values). – **Detection approach:** anomaly-based (focus on DoS attacks).	– **Datasets:** MQTTset (reduced and modified versions). – **Evaluation metrics:** accuracy, F1-score, and training/testing time.	– No discussion on adversarial attacks or model robustness. – Lack of comparison with deep learning models (e.g., LSTM and MLP), which are increasingly used for anomaly detection. – Did not apply valid feature selection mechanisms rather manual crafting

Rahmeh Fawaz et al., 2022 [37]	– To develop a decentralized DDoS prevention system for IoT using the Ethereum blockchain, focusing on device authentication and malicious IP tracking.	– **Model architecture:** Ethereum smart contracts with ECDSA for authentication. – **Detection approach:** anomaly-based (gas limit monitoring). – **Feature selection:** not considered (blockchain-native).	- **Dataset:** custom simulation - **Metrics:** authentication time, standard deviation, and gas limit efficiency.	– Narrow scope: focuses solely on application-layer DDoS attacks, ignoring volumetric or protocol-based (MQTT) attacks. – Simulated datasets may not reflect real-world IoT traffic. – Scalability issues due to public blockchain. – No comparison with ML-based detection methods. – High efficiency in authentication but
Al-Fayoumi and Al-Haija, 2023 [38]	– To develop a lightweight ML-based detection system for LR-DDoS attacks targeting MQTT protocol in SD-IoT, focusing on minimal feature sets and high-speed detection.	- **Model architectures:** decision tree classifier (DTC), multilayer perceptron (MLP), artificial neural networks (ANN), and naïve Bayes classifier (NBC). - **Feature selection:** PCA (reduced to 2 features). - **Detection approach:** anomaly-based (ML-driven).	– **Dataset:** LRDDoS-MQTT-2022 (200K samples and balanced). – **Metrics reported:** Decision Tree Classifier (DTC) - accuracy (99.5%), F1-score (99.35%), and False Negative Rate (FNR) -0.6%	– Limited to low-rate DDoS (LRDDoS) attacks based MQTT protocol; other DDoS variants (e.g., SYN floods) not tested. – Limited applicability of classical machine learning models for complex datasets and large traffic generated by IoT devices. – Its real applicability has strongly limited to only two features (CSUM and SRC_PORT)
Álvaro et al, 2024 [39]	– To develop a visual tool using unsupervised projection techniques for human experts to detect MQTT-based attacks in IoT networks.	– **Model architectures:** unsupervised Beta Hebbian Learning (BHL), t-distributed stochastic neighbor embedding (t-SNE), and ISOMAP – **Feature selection:** 41 MQTT protocol features – **Detection approach:** anomaly-based (visual clustering).	– **Datasets:** custom-generated (real MQTT traffic + Sybil attacks). – **Metrics:** qualitative visualization (no quantitative metrics like accuracy/F1).	– No quantitative detection metrics (e.g., accuracy or False Positive Rate (FPR)). - Limited to Sybil attacks; other MQTT attacks (e.g., DoS) not tested. – Computational cost of Isometric feature mapping (ISOMAP) ISOMAP not addressed.
Dongdong, and Ji, 2024 [40]	– To propose a peer-to-peer (P2P) two-factor authentication (2FA) protocol for IoT	– **Model architectures:** P2P Chord architecture, AES encryption, and XOR-based key derivation. – **Feature selection:** packet size, transmission time, and pre-shared coefficients – **Detection approach:** anomaly-based (implicit in authentication failure handling).	– **Datasets:** simulated IoT traffic (no public dataset used). – **Metrics:** response time, memory usage, and connection latency.	– No comparison was conducted with deep learning-based authentication methods. – Limited to CoAPMicro – Simulated environment lacks real-world IoT heterogeneity.
Al-Haija and Droos, 2025 [41]	– To provide an extensive overview of current research on deep learning (DL)-based intrusion detection systems (IDS) for IoT. It examines DL	– **Model architectures**: discusses CNN, LSTM, Autoencoders, DNN, and hybrid models (e.g., CNN-LSTM). – **Feature selection**: highlights techniques like PSO (Particle Swarm Optimization) but does not deeply compare methods like	– **Datasets:** reviews NSL-KDD, Bot-IoT, CIC-IDS2017, and N-BaIoT but does not specifically mention MQTT-based datasets. – **Evaluation metrics**: commonly use accuracy,	– **Scope**: broad coverage of IoT IDS but lacks depth in MQTT-specific security (e.g., MQTT-based DDoS attacks). – **Datasets**: most datasets are general IoT traffic; MQTT-specific attacks (e.g., QoS exploits) are not addressed. – Limited discussion on deploying DL-IDS in resource-constrained IoT edge

	architectures, datasets, challenges, and future directions to enhance IoT security.	chi-square or mutual information. – **Detection approaches**: focus on anomaly-based DL methods, with limited discussion on signature-based techniques.	precision, recall, and F1-score. No statistical significance testing – **Real-time performance**: mentions computational efficiency but lacks detailed latency analysis.	devices.
Sujitha and Santhi, 2025 [42]	– Investigate how advanced features of MQTT v5.0 can be exploited to launch DDoS attacks in IoT deployments and evaluate their impact.	– **Model architectures:** not applicable (attack modeling, but not detection). - **Feature selection:** protocol feature analysis (MQTT v5.0 properties). - **Detection approaches:** not discussed (focus on attack feasibility).	– **Datasets:** real-world IoT testbed (healthcare monitoring system). – **Evaluation metrics:** CPU/memory utilization, and E2E delay. – **Real-time performance:** attack impact measured, but no detection rates reported and analyzed.	– Limited to only four MQTT brokers including Mosquitto, EMQX, HiveMQ, and VerneMQ. – No comparison with MQTT v3.1.1 attacks.

Table 2: Dataset comparison for DDoS on MQTT

Dataset	**Relevance to MQTT specific DDoS attacks**			**Realism and data quality**			**Feature suitability for deep learning**			**Key insight**
	Protocol focus	**MQTT-based DDoS inclusion**	**Attack diversity**	**Traffic source**	**Labeling accuracy**	**Normal/attack ratio**	**Packet-level features**	**Flow-based features**	**Preprocessing needed**	
N_BaIoT	General IoT (HTTP and TCP)	No	Mirai and Bashlite	Real IoT botnets	High (manual verification)	Highly imbalanced (attack-heavy)	Yes (time-series)	No	High (normalization)	Focused on Wi-Fi communication; lacks MQTT context but suits general anomaly detection
BoT_IoT	Mixed (HTTP, MQTT, and CoAP)	Partial (limited MQTT focused)	DDoS, DoS, and Recon	Simulated + real	Moderate (auto-labeled)	Balanced	Partial	Yes	Moderate	Strong for general IoT attack detection but lack native MQTT traffic: limited its direct

ToN_IoT	Multi-protocol (MQTT and [illegible])	Yes (but sparse MQTT-DDoS)	Flooding and Brute-force	Hybrid (testbed + synthetic)	High (multi-source)	Slightly attack-heavy	Yes	Yes	Moderate	Authentication and disconnection phase not found
MQTT-IoT-IDS2020	Native (MQTT-only)	Yes	Flooding and Spoofing	Synthetic (testbed)	High (scenario-based)	Balanced	Yes (MQTT headers)	No	Low	Built for MQTT environments with limited attack variants; doesn't contain flow-based features for
MQTTset	Native (MQTT-only)	Yes (optimized)	Flooding, SlowITe, and Payload Injection	Testbed + real devices	Very high (manual + PCAP-based validation)	Realistic imbalance	Yes (full payload + metadata)	Partial	Low (preprocessed)	Built for MQTT environments; but doesn't update with new attack techniques or protocol

3. RESEARCH METHODOLOGY

This research aims to detect and classify security risks by effectively capturing and analyzing complex communication patterns in MQTT network flows. This facilitates proactive reduction of attack impact, timely mitigation, minimized service downtime, and improved overall system performance. The collected datasets were preprocessed, and appropriate feature engineering was performed before training the models. The proposed solution was compared against a baseline approach to demonstrate improved accuracy. Figure 2 illustrates the adopted methodology.

3.1. MQTTset dataset description

Among publicly available datasets (discussed under Table 2), the MQTTset was identified as suitable being it combines real device traffic with rigorous labeling in order to enhance realism for deep learning development [34]. MQTTset uses an experimental setup with Eclipse Mosquitto and represents real-world scenarios by considering eight diverse simulated IoT sensors (temperature, humidity, CO2 levels, motion, smoke, doors, and fans) across two virtual rooms over ten days, generated with the IoT-Flock tool [35]. MQTTset is extensively utilized in smart environments such as homes and offices, where traffic generated using MQTT version 3.1.: a lightweight publish/subscribe protocol is recorded. As such it served as the primary data source which comprising a diverse range of network behaviors including legitimate traffic (approx. 11,000,000 instances) and various attack vectors: Flooding DoS (130,233), Brute Force (14,501), Malformed Data (10,924), SlowITe (9,202), and MQTT Publish Flood (613). The raw network traffic was originally captured in Packet Capture (PCAP) format and subsequently processed to extract 34 discriminative features essential for cybersecurity analysis.

To ensure computational feasibility and model specialization, we applied a strategic filtering process [46-48]. From the extensive pool of legitimate traffic, a representative subset of 165,475 legitimate instances was selected for further analysis. Furthermore, this study focuses exclusively on Flooding DoS attacks by utilizing 130,233 samples explicitly labeled as 'Flooding DoS' in the original Comma Separated Values (CSV) files. These samples were validated based on their volumetric signatures characterized by high-frequency CONNECT and PUBLISH sequences, which clearly distinguishes these attacks from low-rate and resource-exhaustion anomalies such as SlowITe. The attack targets the MQTT broker by rapidly establishing multiple connections and saturating its processing capacity. This criteria-based filtering facilitates a robust binary classification framework (Legitimate vs. DDoS) aligning with established benchmarking standards in IoT security research [35], [45].

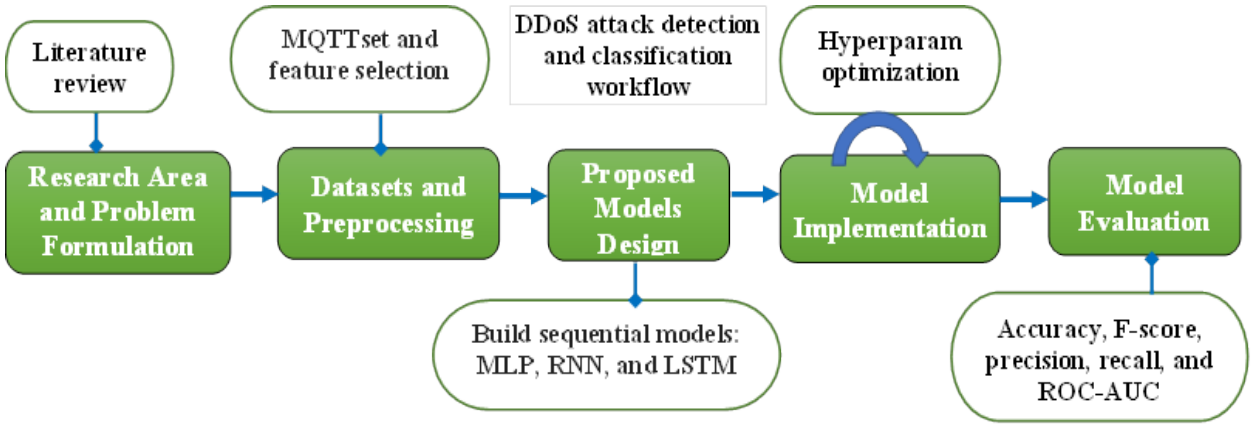

Figure 2: Schematic diagram of the research methodology

3.2. Data preprocessing

Data preprocessing is essential for transforming raw network traffic into a clean, consistent format suitable for deep learning models. This step enhances data quality, removes noise and redundancies, and ensures optimal model performance and learning efficiency. The MQTTset dataset contains 34 features originally captured in PCAP format and exported to CSV. Prior to training, the data underwent the following preprocessing pipeline:

- **Data cleaning:** Redundant and irrelevant features (e.g., TCP header fields) were removed to reduce dimensionality. Missing values were handled, and data types were standardized across all records. As shown in Table 3, three distinct data formats were harmonized into consistent numerical and categorical types to prevent parsing errors during training.
- **Feature normalization:** Numerical features exhibited varying scales and ranges, which could bias the learning process. To mitigate this, all features were scaled to a common range using MinMaxScaler, preventing features with larger magnitudes from dominating the model and accelerating convergence. Converting the data to a consistent format as the datasets contained three different data type formats, as represented in Table 3. Unlike generic IoT DDoS detection, we focus on protocol-specific exploits unique to MQTT (e.g., CONNECT flood malformations) with this work to correlate QoS levels (0-2) with attack detectability. Filtering the most prevalent and critical attack instances from the MQTTset dataset is a crucial step for our targeted analysis or model training. To ensure data quality and model coherence, standard criteria of class filtering considered [45] - [48]: included minimizing class imbalance, simplifying training (binary classification), and ensuring deployment practicality (focusing on high-impact attacks), and maintaining feature consistency (DDoS and legitimate).

Table 3: Characteristics of MQTTset features

No.	Features name	Data types	Description
1	tcp.flags	Object	TCP flag status
2	tcp.time_delta	Float64	Time between TCP stream
3	tcp.len	Int64	TCP segment length
4	mqtt.conack.flags	Object	Acknowledged flags status by broker and status of MQTT connection.
5	mqtt.conack.flags.reserved	Int64	Reserved
6	mqtt.conack.flags.sp	Int64	Indicate session present.
7	mqtt.conack.val	Int64	Return code value.
8	mqtt.conflag.cleansess	Int64	Clean session flag either 0 or 1
9	mqtt.conflag.passwd	Int64	Password flag
10	mqtt.conflag.qos	Int64	Indicate QoS level value.
11	mqtt.conflag.reserved	Int64	Reserved
12	mqtt.conflag.retain	Int64	Will retain value
13	mqtt.conflag.uname	Int64	User name flag
14	mqtt.conflag.willflag	Int64	Will flag to indicate if the connection terminates abnormally.
15	mqtt.conflags	Object	Connect flag
16	mqtt.dupflag	Int64	Duplicate flags
17	mqtt.hdrflags	Object	Header flag in MQTT messages.
18	mqtt.kalive	Int64	Keep-alive
19	mqtt.len	Int64	MQTT Message length

20	mqtt.msg	Object	MQTT message
21	mqtt.msgid	Int64	Message identifier
22	mqtt.msgtype	Int64	MQTT message type
23	mqtt.proto_len	Int64	Protocol name length
24	mqtt.protoname	object	Protocol name
25	mqtt.qos	Int64	Indicate QoS level
26	mqtt.retain	Int64	Retain
27	mqttt.sub.qos	Int64	Requested QoS by subscriber
28	mqtt.suback.qos	Int64	Granted QoS by broker
29	mqtt.ver	Int64	Indicate the version of the protocol
30	mqtt.willmsg	Int64	Will message
31	mqtt.willmsg_len	Int64	Will message length
32	mqtt.willtopic	Int64	Will topic
33	mqtt.willtopic_len	Int64	Will topic length
34	Target	object	Category value either **DDoS** or **legitimate**

- **Label encoding:** For binary classification, categorical labels were encoded numerically: 0 for legitimate traffic and 1 for DoS attacks. This enables the model to interpret class distinctions during supervised training. These steps were implemented using Python libraries (e.g., pandas and scikit-learn) to ensure reproducibility. The final preprocessed dataset is balanced, normalized, and structurally consistent, providing a reliable foundation for training and evaluating deep learning-based intrusion detection models.
- **Imbalance handling**: The considered MQTT dataset exhibited class imbalance, with 165,475 legitimate samples (majority) and 130,233 DoS attacks (minority), which leading to potential model bias toward the majority class. We employed a hybrid resampling approach to balance the classes while maintaining computational efficiency: well-established strategy in IoT security [12], [43]. Undersampling (lightweight design) was first applied to reduce the majority class and minimizing training time and memory footprint. Synthetic Minority Oversampling Technique (SMOTE) was then used to generate synthetic minority samples by ensuring effective attack detection without compromising lightweight design. Undersampling: reduced legitimate traffic from 165,475 to 90,883 (random selection), and also applied the same ratio (0.55) to DoS attacks, downsizing them from 130,233 to 71,628 samples. SMOTE: considered to mitigate the minority bias (DoS attacks) by increasing DoS attacks from 71,628 to 91,166 through generating 19,538 synthetic samples via interpolation (k=5 neighbors). As result, synthetic oversampling prevented overfitting and improved minority-class generalization. As balanced class distribution reported in the final outcome, legitimate (90,883) and DoS (91,166), model performance enhanced by mitigated bias and ensuring robust detection of both classes. Class distributions of MQTT dataset before and after balanced has shown in Figure 3. To prevent data leakage and ensure unbiased performance estimation, all synthetic samples generated via SMOTE were restricted to the training set, while the test set remained entirely original (non-synthetic).

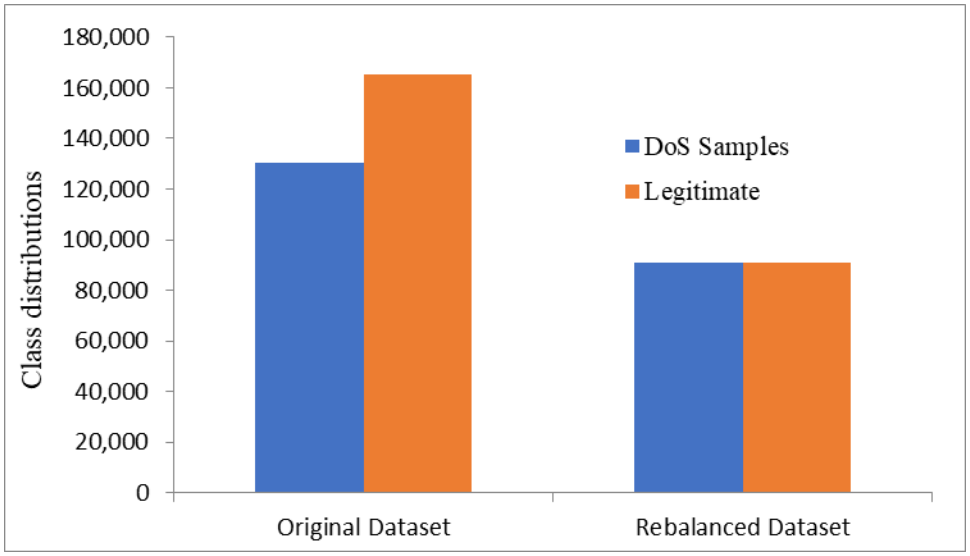

Figure 3: Class distribution before and after balanced

3.3. Feature selection

Feature selection plays a critical matter in ensuring both the robustness and efficiency of candidate models for MQTT-based IoT intrusion detection. Initially several established techniques were considered, including information gain, embedded methods such as random forest, correlation analysis, and chi-square (χ^2) tests [12], [22], [42], [44]. While information gain and random forest are widely used, they are often biased toward high-cardinality features and computationally expensive when applied to high-dimensional IoT traffic. Mutual information was also inspected but excluded due to its sensitivity to small sample sizes in minority attack classes, which could distort feature relevance rankings.

Given the heterogeneous nature of MQTT features consisting of numerical attributes (e.g., message rate and payload size) and categorical attributes (e.g., QoS level and message type), we selected a hybrid approach using Pearson correlation and chi-square tests. This combination balances statistical rigor with computational efficiency by ensuring that selected features are both relevant and interpretable in the context of MQTT traffic analysis. Previous studies have separately demonstrated the effectiveness of Pearson correlation for numerical features [36] and chi-square tests for categorical features [33] in evaluating feature relevance and selection outcomes. Instead of separating, the integration of Chi-square and Pearson correlation leverages complementary strengths as stated in prior studies [22], [44]. This hybrid approach captures both categorical dependencies (chi-square) and linear relationships (Pearson) while remaining less computationally intensive than mutual information and more generalizable than model-specific embedded methods. The hybrid scoring function (**using 0.6*χ^2 + correlation*0.4**) was adopted to ensure a balanced representation of both categorical and numerical data types. Following established feature selection practices in IoT security research, a higher weight **(60%)** was assigned to the Chi-square component to prioritize the categorical significance of MQTT protocol headers, which are highly discriminative for detecting volumetric flooding behaviors [47], [48]. Simultaneously, Pearson correlation (40%) was utilized to identify linear dependencies in numerical flow metrics. This weighted integration ensures that the selected features capture both protocol-level violations

and statistical traffic anomalies through providing a more comprehensive input for the classification model compared to single-metric filters.

Figure 4 describes the proposed hybrid feature selection procedure. The process comprises four main stages: (1) initial evaluation of 34 candidate features; (2) feature ranking using a weighted combination of Pearson correlation and chi-square statistics; (3) prioritization based on the computed score; and (4) final selection of 13 MQTT protocol attributes.

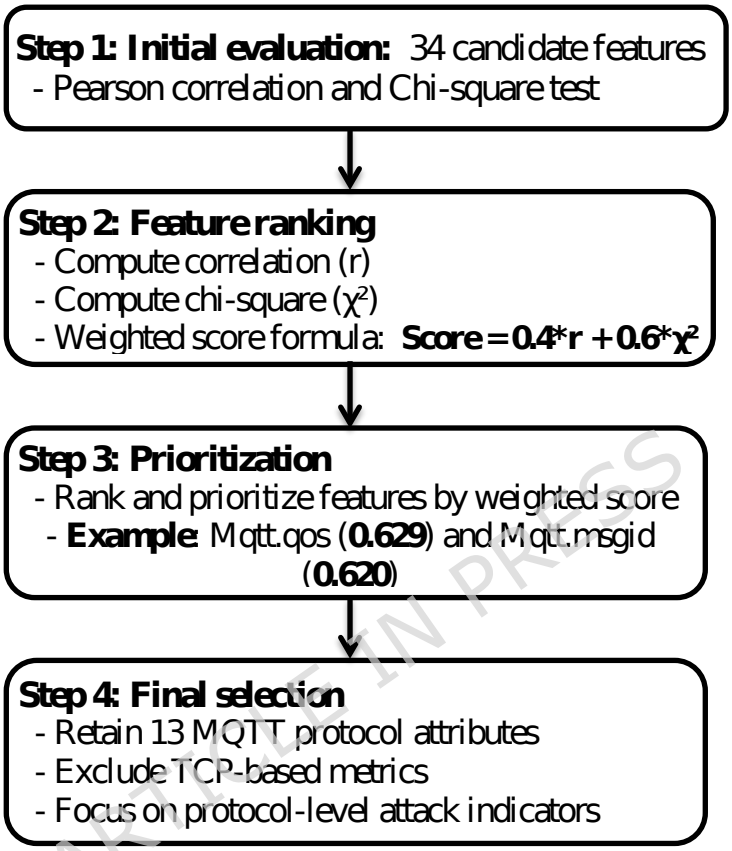

Figure 4: Flow diagram for hybrid feature selection process

To ensure comparability between Pearson correlation coefficients and chi-square statistics, each metric was normalized to a [0,1] scale using min–max normalization. A weighted score was then computed for each feature that empirically optimized on a validation set with the (0.6 and 0.4) combination yielding the highest F1-score during our systematic experimentation. This hybrid weighting approach align with other studies, eg., the precedent of ensemble feature selection [48], which balances statistical significance across different data types (numerical and categorical) to ensure a more robust feature subset for binary classification. Features with the highest weighted scores were retained and resulting in a final subset of 13 MQTT protocol attributes, as illustrated in Figure 5.

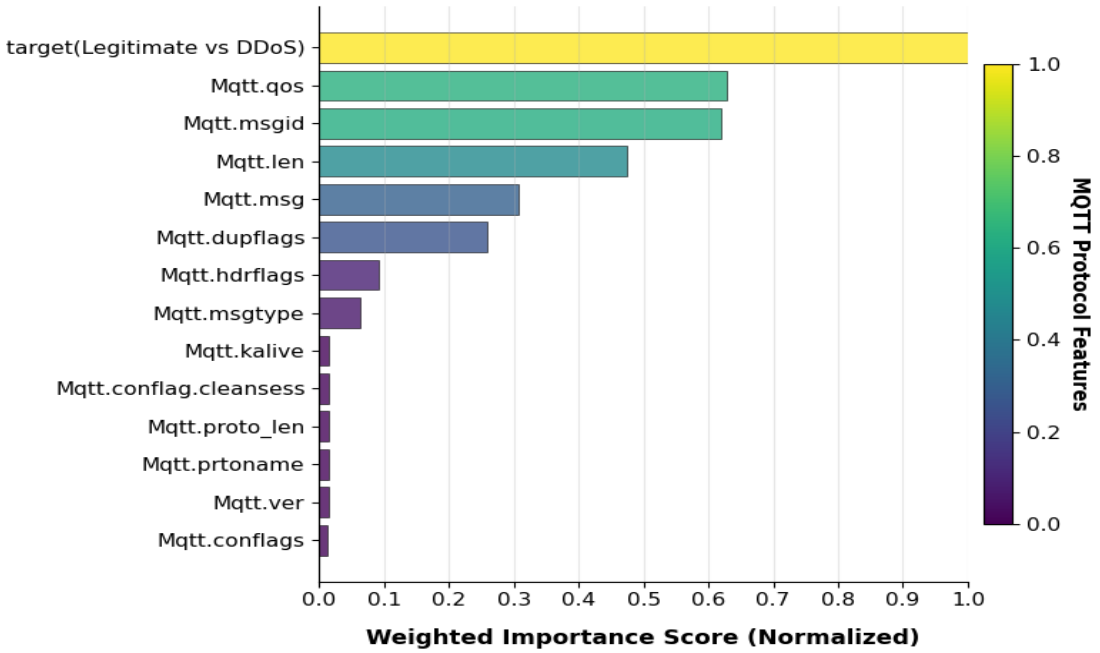

Figure 5: Feature importance ranking for MQTT intrusion detection

The weighted scores reveal critical insights into MQTT features that effectively distinguish legitimate traffic from DDoS attacks: eg., eg., Mqtt.qos (0.629), Mqtt.msgid (0.620), Mqtt.len (0.476), Mqtt.msg (0.308) and etc.. Mqtt.qos and Mqtt.msgid are the most influential, suggesting DDoS attacks often manipulate QoS levels or reuse message IDs. Similarly, Mqtt.len and Mqtt.msg are moderately useful, as attacks may alter packet sizes or payload content. The hierarchy of feature importance demonstrates that attackers primarily exploit protocol-level characteristics (QoS and message IDs) in addition to packet and payload content that should be guiding more effective detection strategies. On the other hand, MQTT-focused specific features (e.g., Mqtt.msg and Mqtt.len) better to capture attack patterns than the generic TCP metrics. At the end, these three TCP-based features were excluded from our final decision in order to only maintain 13 MQTT protocol attributes (as validated by normalized importance scores). In essence, the hybrid Pearson–chi-square based selection process ensures statistical rigor by evaluating numerical and categorical features appropriately and also maintains domain specificity by focusing on MQTT protocol attributes/semantics rather than others. This approach achieves efficiency through a weighted scoring system that yields a compact and discriminative feature set. This improves detection performance, balances computational efficiency, and interpretability, as the selected features directly reflect protocol behaviors exploited in attacks.

3.4. Model selection

Selecting a deep learning model for DDoS attack detection and classification in IoT devices requires careful consideration of multiple criteria due to the unique challenges posed by IoT environments (e.g., limited resources, high attack variability, and large-scale traffic). Further, MQTT specific attack patterns and model capabilities to generalize well on unseen data and handle huge volumes of data were the central pillars applied for considering the candidates including MLP, RNN, and LSTM. RNN provides basic sequence modeling, LSTM introduces long-term memory mechanisms, and MLP offers a non-sequential baseline. This comparative framework

allows us to determine whether attack patterns are better captured through temporal dependencies (RNN/LSTM) or through static feature correlations (MLP).

3.5. Model training and validation

To optimize model performance and ensure robustness, we implemented a comprehensive training and validation framework. Hyperparameters were systematically tuned with final configurations designed for IoT efficiency (detailed in Table 4). The dataset was split into 70% for training and 30% for testing, with an additional hold-out validation set used for early stopping. We also employed 5-fold cross-validation to enhance generalizability across dynamic network flows. Model performance was evaluated using accuracy, precision, recall, F1-score, false positive rate (FPR), false negative rate (FNR), and misclassification error, supported by confusion matrix visualization and ROC curve analysis. The proposed LSTM was comparatively assessed against RNN and MLP baselines under identical conditions to isolate the benefits of temporal modeling.

4. PROPOSED ARCHITECTURE FOR DDOS ATTACK DETECTION AND CLASSIFICATION

The main aim of this study is the detection and classification of DDoS attacks in the MQTT-IoT ecosystem using deep learning models. It consists of three main phases: at the initial stage we assume that the MQTT network datasets were generated by IoT devices, then relevant features were identified by computing the weighted score obtained from correlation and chi-square in order to reduce computational resources and misclassification error [44]. Final, the pinpoint features fit into the model with appropriate hyperparameter settings for training and testing the candidate deep learning.

4.1. Proposed solution architecture

Figure 6 presents the proposed end-to-end architecture which is systematically organized into four stages: data preprocessing, feature selection, model building and training, and performance evaluation. It is systematically divided into four core stages: data preprocessing, feature selection, model training and evaluation, and model deployment for inference. The flow ensures raw MQTT traffic is transformed into actionable security intelligence through a structured pipeline optimized for deep learning. The preprocessing stage transforms raw MQTT network traffic into a clean, structured dataset suitable for machine learning. This involves data cleaning to remove noise, duplicates, and missing values; data type conversion for consistency across features; normalization to scale numerical values to a common range (e.g., [0,1]), preventing feature dominance and improving convergence; label encoding to convert class labels into numerical values (0 for legitimate and 1 for DoS); and data balancing using random under sampling and SMOTE to mitigate class imbalance. Collectively, these steps ensure the data is reliable, standardized, and optimized for training robust detection models.

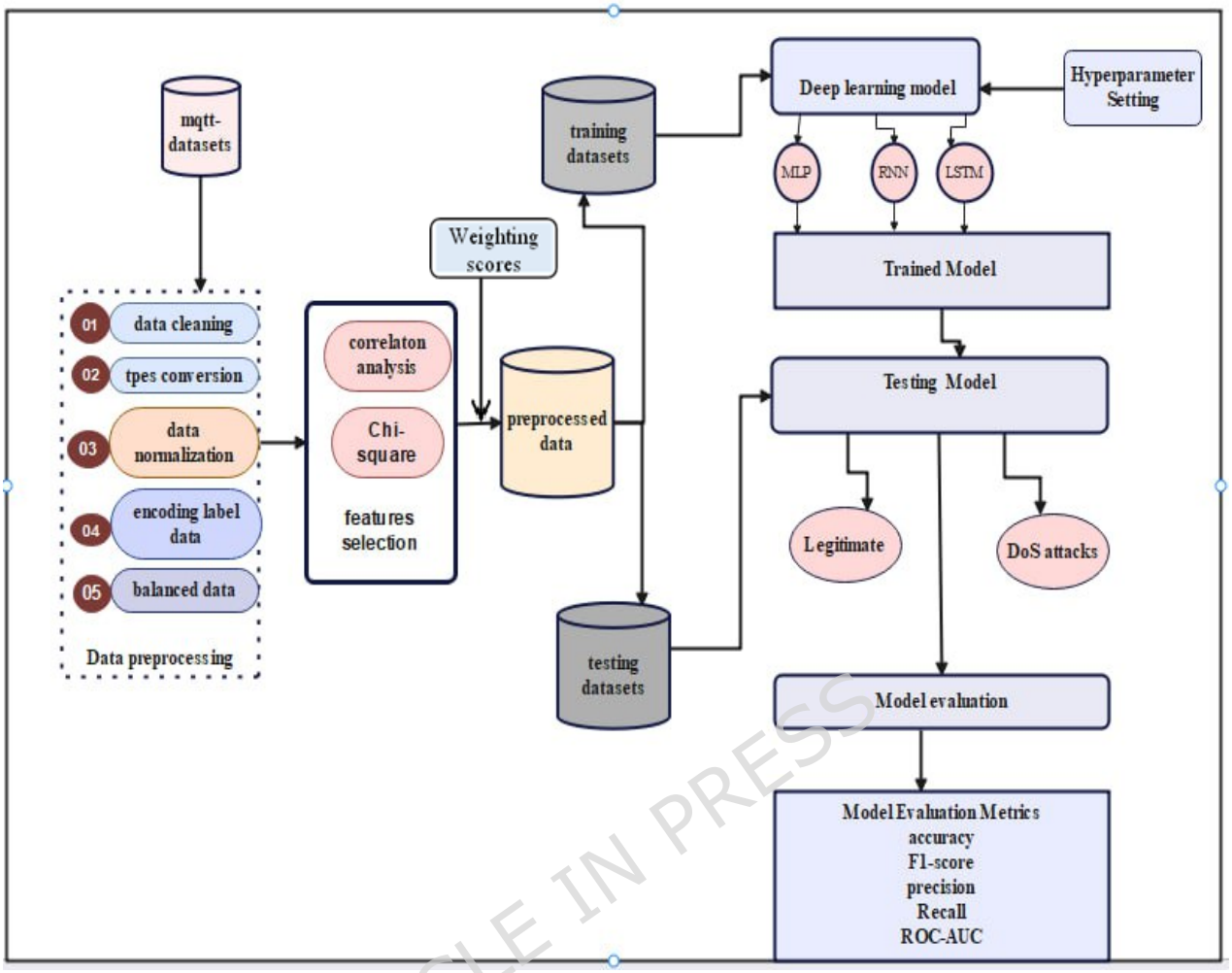

Figure 6: The proposed deep learning architecture illustrating the end-to-end process from input features to DDoS classification

Following preprocessing, the feature selection enhances model efficiency and interpretability through a dual-criteria approach: Pearson correlation eliminates redundant features by identifying their high multicollinearity, while the Chi Square test evaluates categorical feature relevance to the target variable (attack vs. legitimate). Then, normalizing the important feature scores was employed to select the optimal subset that split into training and testing datasets by establishing a robust evaluation framework.

In the model development stage, three deep learning architectures are implemented for comparative analysis: a MLP as a non-sequential baseline, a RNN for basic temporal dependencies, and a LSTM network designed to capture long range sequential patterns inherent in MQTT attack flows. Hyperparameters are systematically tuned to optimize learning rate, batch size, layer configuration, and regularization to yield the final trained model. Performance is evaluated using metrics including accuracy, precision, recall, F1 score, and ROC AUC by providing a comprehensive assessment of detection capability. The validated model is subsequently prepared for operational deployment in a simulated or real IoT-MQTT environment, where it functions as a lightweight Intrusion Detection System (IDS). It analyzes live or batched MQTT traffic in near-real-time by classifying packets as legitimate or DDoS attacks thereby enabling proactive network security.

This proposed architecture provides a systematic, reproducible framework that moves from loading raw MQTT data to a deployable security solution. It emphasizes robust preprocessing, intelligent feature selection, and a comparative evaluation of advanced deep learning models (with a focus on LSTM's strengths for sequential data). Lastly, the LSTM-based IDS framework that mitigates concept drift through incremental online learning is integrated for safeguarding IoT-MQTT infrastructures against DDoS threats.

4.2. Algorithm flows of the proposed model

To illustrate the algorithmic flow, we describe the sequential operations of each deep learning model. As this is a binary classification task, binary cross-entropy is used as the loss function. The general framework for the selected deep learning architecture:

Algorithm: deep learning model **(LSTM, MLP, or RNN)**
Input: x,y MQTT network (MQTTset datasets) flows.
Output: A map object with the identified labels for every MQTT network flow, indicating if the flow is a DoS attack or not.

Begin()

Define deep learning types:
- Build a sequential model.
- Add an input layer with 14 neuron units with an activation function of ReLu, and an input shape of x_train[1],1).
- Add a dense layer of 7 neuron units with an activation function ReLu.
- Add a dense layer of 1 unit with a sigmoid function.
- Add a dropout rate of [0.001 to 0.002].

Compile the defined model:
- Add an optimizer of 'Adam' with a learning rate of [1e-1 to 1e-2].
- Set the loss function to binary_crossentropy.
- Set 'accuracy' as evaluation metrics.

Training the defined model:
- Fit the model applying x_train and y_train with a train split hold-out and 5-fold CV.
- Incorporate callbacks of early stopping to monitor validation loss.
- Set an epoch size to 100.
- Set a batch size of 1000.

Testing the defined model:
- Evaluate the model performance by applying test split with hold-out and 5-fold CV.
- X_test or unseen data.

Return detection_classification{}

End()

4.3. Model building

This study presents an optimized model for effectively detecting and classifying DoS attacks in MQTT-IoT networks. Our model selection prioritizes IoT deployability through architectural minimalism. We evaluate three computationally distinct approaches: MLP for maximum efficiency, RNN for lightweight temporal awareness, and LSTM for optimal accuracy-complexity balance by using carefully designated libraries and hyperparameters for optimal performance. The detailed model configurations, including the critical parameter settings, are illustrated in Figures 7-9.

```
# Build the MLP model for binary classification
model = Sequential()
model.add(Dense(14, input_dim=x11_train.shape[1], activation='relu'))
model.add(Dense(7, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.add(Dropout(0.001))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=8, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.01)
# Train the MLP model
history=model.fit(x11_train, y11_train, epochs=100, batch_size=1000,
        validation_split=0.3, callbacks=[early_stopping, reduce_lr])
```

```
.9951 - lr: 0.0010
Epoch 95/100
90/90 [==============================] - 0s 2ms/step - loss: 0.0139 - accuracy: 0.9949 - val_loss: 0.0083 -
.9950 - lr: 0.0010
Epoch 96/100
90/90 [==============================] - 0s 3ms/step - loss: 0.0148 - accuracy: 0.9951 - val_loss: 0.0083 -
.9950 - lr: 0.0010
Epoch 97/100
90/90 [==============================] - 0s 3ms/step - loss: 0.0149 - accuracy: 0.9952 - val_loss: 0.0083 -
.9955 - lr: 0.0010
Epoch 98/100
90/90 [==============================] - 0s 3ms/step - loss: 0.0166 - accuracy: 0.9948 - val_loss: 0.0084 -
.9951 - lr: 0.0010
Epoch 99/100
90/90 [==============================] - 0s 3ms/step - loss: 0.0141 - accuracy: 0.9949 - val_loss: 0.0085 -
.9951 - lr: 0.0010
Epoch 100/100
90/90 [==============================] - 0s 3ms/step - loss: 0.0148 - accuracy: 0.9950 - val_loss: 0.0085 -
.9951 - lr: 0.0010
```

Figure 7: MLP model implementation

```
# Build the RNN  Model for Binary classification
rnn_model = Sequential()
rnn_model.add(SimpleRNN(14, input_shape=(X_train_rnn.shape[1],1), activation='relu'))
rnn_model.add(Dense(7, activation='relu'))
rnn_model.add(Dense(1, activation='sigmoid'))
rnn_model.add(Dropout(0.002))
rnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=8, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.001)
# Train the RNN model
history11=rnn_model.fit(X_train_rnn, y1_train, epochs=100, batch_size=1000, validation_split=0.3,
                    callbacks=[early_stopping, reduce_lr])
```

```
.9792 - lr: 0.0010
Epoch 14/100
90/90 [==============================] - 0s 5ms/step - loss: 0.0570 - accuracy: 0.9832 - val_loss: 0.0205 -
.9919 - lr: 0.0010
Epoch 15/100
90/90 [==============================] - 0s 5ms/step - loss: 0.0345 - accuracy: 0.9931 - val_loss: 0.0119 -
.9949 - lr: 0.0010
Epoch 16/100
90/90 [==============================] - 0s 5ms/step - loss: 0.0261 - accuracy: 0.9945 - val_loss: 0.0097 -
.9955 - lr: 0.0010
Epoch 17/100
90/90 [==============================] - 0s 5ms/step - loss: 0.0226 - accuracy: 0.9951 - val_loss: 0.0084 -
.9960 - lr: 0.0010
Epoch 18/100
90/90 [==============================] - 0s 5ms/step - loss: 0.0254 - accuracy: 0.9951 - val_loss: 0.0084 -
.9955 - lr: 0.0010
Epoch 19/100
90/90 [==============================] - 0s 5ms/step - loss: 0.0239 - accuracy: 0.9951 - val_loss: 0.0083 -
.9960 - lr: 0.0010
```

Figure 8: RNN model implementation

```
[3]: lstm_model = Sequential()
lstm_model.add(LSTM(14, input_shape=(X_train_lstm.shape[1], 1), activation='relu'))
lstm_model.add(Dropout(0.001))

lstm_model.add(Dense(7,activation='relu'))
lstm_model.add(Dropout(0.001))

lstm_model.add(Dense(1, activation='sigmoid'))
lstm_model.add(Dropout(0.001))
lstm_model.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=8, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.001)
# Train the LSTM model0
history = lstm_model.fit(X_train_lstm, y_train, epochs=100, batch_size=1000,
                         validation_split=0.3,callbacks=[early_stopping, reduce_lr])
```

```
.9962 - lr: 0.0010
Epoch 65/100
90/90 [==============================] - 1s 9ms/step - loss: 0.0135 - accuracy: 0.9956 - val_loss: 0.0063 - val_accuracy: 0
.9957 - lr: 0.0010
Epoch 66/100
90/90 [==============================] - 1s 9ms/step - loss: 0.0155 - accuracy: 0.9951 - val_loss: 0.0063 - val_accuracy: 0
.9962 - lr: 0.0010
Epoch 67/100
90/90 [==============================] - 1s 8ms/step - loss: 0.0125 - accuracy: 0.9956 - val_loss: 0.0063 - val_accuracy: 0
.9955 - lr: 0.0010
Epoch 68/100
90/90 [==============================] - 1s 8ms/step - loss: 0.0145 - accuracy: 0.9953 - val_loss: 0.0063 - val_accuracy: 0
.9955 - lr: 0.0010
Epoch 69/100
90/90 [==============================] - 1s 9ms/step - loss: 0.0136 - accuracy: 0.9954 - val_loss: 0.0063 - val_accuracy: 0
.9962 - lr: 0.0010
Epoch 70/100
90/90 [==============================] - 1s 9ms/step - loss: 0.0128 - accuracy: 0.9956 - val_loss: 0.0063 - val_accuracy: 0
.9962 - lr: 0.0010
```

Figure 9: LSTM model implementation

4.4. Hyperparameter tuning strategy and process

Following the foundational principles outlined in Sections 3.1 and 3.2, the hyperparameter tuning process for this study was carefully designed to balance model performance with the stringent computational constraints of IoT deployment environments such as limited memory/storage, compute power, and energy budgets. Although conventional methods such as Grid Search or Bayesian Optimization often yield the theoretical global optimum, they are computationally prohibitive and impractical for resource-constrained scenarios where models are tuned offline and deployed with fixed parameters [12]. Therefore, a manual, iterative tuning approach was adopted, guided by empirical evidence and a deep understanding of each model's architecture. This method is not only computationally efficient but also provides invaluable insights into model behavior and facilitating tailored adjustments that automated searches might overlook. The primary objective was to identify a set of robust hyperparameters that ensure generalization, prevent overfitting, and enable feasible deployment, rather than pursuing marginal gains at excessive computational cost. Our tuning process focused on several key hyperparameters, as summarized in Table 4. The selection was driven by their proven impact on model convergence, stability, and final performance.

The hyperparameters for each model were finalized through a principled, iterative tuning process, which was governed by the following rationale:

- Number of hidden layers and hidden units: For IoT security applications, the architecture of models must balance detection accuracy with computational

tractability. The model's compact structure leverages 14 units for scalable capacity with a 7-unit bottleneck serving as a constraint to enhance generalization and prevent overfitting. The design of all models was guided by IoT resource constraints, avoiding overfitting on given data, empirical performance validation, and unified experimental design [12]. Deeper or wider networks increase parameter counts, memory usage, and inference latency which make them unsuitable for resource-constrained edge nodes. By keeping the number of layers and hidden units identical across the models, we are ensuring that any performance differences observed are due to the inherent architectural design of each model type, and not simply because one model was larger or had more parameters.

- Architecture-informed learning rates: The learning rate was empirically tuned to suit each model's characteristics. Recurrent networks such as RNN and LSTM are sensitive to gradient instability (vanishing or exploding gradients). A conservative learning rate of 0.001 was chosen to ensure stable weight updates. In contrast, the simpler feed-forward architecture of the MLP allowed for a faster learning rate of 0.01, which is promoting quicker convergence without compromising stability.
- Targeted regularization for overfitting control: The significant performance discrepancy for the MLP observed between hold-out and 5-fold CV (as discussed in Section 5.2) underscores the critical role of regularization. While early stopping was universally applied and dropout rates were set minimally. The RNN's slightly higher dropout (0.002) aims to lightly regularize its susceptibility to overfitting on sequential data without destroying important temporal dependencies. The LSTM's inherent gating mechanism and the MLP's simpler structure warranted an even lower dropout (0.001) to primarily avoid the risk of underfitting, which is a genuine concern in highly constrained models.
- Consistency in optimization and evaluation**:** Using the Adam optimizer and Binary Cross-Entropy (BCE) loss across all models ensured a consistent and efficient training process. Furthermore, maintaining an identical batch size and using the same early stopping criterion (monitoring validation loss) guaranteed that the models were compared under equivalent training conditions. This isolation of architectural effects is crucial for attributing performance differences to the model designs themselves rather than to disparate tuning strategies

Table 4: Hyperparameter configuration for candidate models

Hyperparameter	**RNN**	**LSTM**	**MLP**	**Justification for IoT/Model context**
Architecture	SimpleRNN(14) → Dense(7) → Dense(1)	LSTM(14) → Dense(7) →	Dense(14) → Dense(7) →	Minimal layers/units for edge efficiency
Parameters	337	1,009	309	Resource-efficient model for lightweight deployment
Output activation	Sigmoid	Sigmoid	Sigmoid	Standard for binary classification; provides probabilistic output
Loss function	BCE	BCE	BCE	Standard for binary tasks with probability based
Optimizer	Adam	Adam	Adam	Adaptive learning rate enables/efficient

				convergence with minimal tuning
Learning rate	0.001	0.001	0.01	Lower rate for RNN/LSTM stabilizes training; higher rate speeds up MLP convergence
Batch size	1000	1000	1000	Consistence in large batch size with hardware-appropriate for stable gradient estimation
Dropout rate	0.002	0.001	0.001	Minimal dropout to prevent overfitting while preserving temporal features (RNN/LSTM) and avoiding underfitting (MLP)
Early stopping	Yes	Yes	Yes	Prevents overfitting by ensuring generalization and saving computational resources

This hyperparameter tuning strategy successfully identified the configurations that yielded highly performing models, as evidenced by the results in Section 5. More importantly, it aligned the model development process with the real resource-constraints of IoT systems, where offline tuning and operational efficiency are paramount. The excellent and stable performance of the LSTM and RNN under these tuned parameters confirmed, as effectively validated by 70-30 hold-out split and 5-fold stratified.

4.5. LSTM-based conceptual framework for concept drift handling in IDS

The present evaluation is confined to only static datasets; however, we have proposed a conceptual framework that leverages the established LSTM architecture to facilitate continuous learning from dynamic data streams. It is a multi-faceted strategy to ensure a long-term model sustainability of our proposed LSTM architecture against concept and data drift in dynamic IoT environments. This includes continuous performance monitoring using drift detection methods (e.g., DDM) to trigger automated adaptation mechanisms. These mechanisms encompass periodic retraining LSTM architecture with online learning capabilities for incremental updates and an ensemble approach for robust voting on novel attacks. It supported by a continuous data pipeline and model version control. Accordingly, the framework is designed to maintain detection efficacy against an evolving threat landscape. A feasible strategy is to adopt a co-design approach to practically deploy this LSTM-based conceptual framework on resource-constrained devices. This could include hardware-software optimization for efficient LSTM inference or a split-computing architecture where lightweight preprocessing occurs on the device, and complex model inference is offloaded to an edge server. Figure 10 illustrates a proactive conceptual framework for handling concept drift in IoT environments using an online LSTM-based ISD.

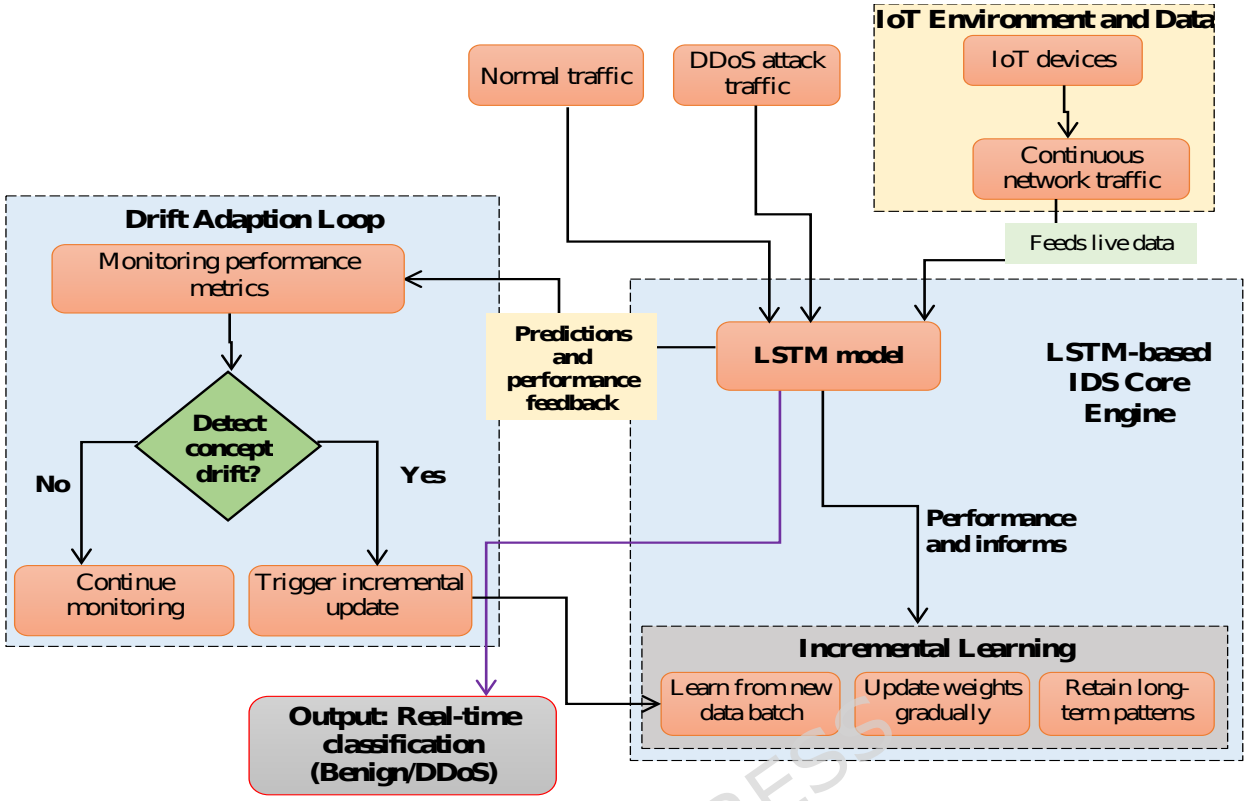

Figure 10: A conceptual framework for handling concept drift in IoT environments using an online LSTM-based IDS

The system continuously monitors incoming network traffic from IoT devices, when its performance metrics are tracked in real-time. When drift is detected; it triggers an incremental update using mini-batch gradient descent rather than a full retrain. The LSTM model learns from data batches, updates its weights gradually using, and retains long-term patterns to adapt to evolving attack behaviors: such as DDoS traffic (without retraining from scratch). The output is a real-time classification of traffic as either benign or malicious, enabling adaptive and resilient security in dynamic IoT settings. To balance responsiveness and stability, model updates are performed on small batches. In stable environments, updates may occur within certain allocated time frame while during attack surges the updates could be triggered.

It is deployed and managed in IoT settings by taking the following implementation actions: i) automate for dynamic networks - continuously monitor and classify live traffic from a mix of devices (sensors, cameras, and gateways) without manual retuning when new devices join or behavior shifts; ii) combat evolving threats proactively - automatically trigger model updates when new attack patterns emerge by ensuring the IDS stays effective against novel DDoS methods without waiting for manual intervention; and iii) deploy at minimal scale overhead - implement incremental updates that require less computational power and storage by enabling the system to run on edge devices or in the cloud across large and distributed IoT networks. Considering this framework, the security teams can maintain a resilient and self-adapting defense layer that reduces response time to emerging threats and lowers operational burden in constantly changing IoT environments. While a comprehensive analysis for real-time deployment on heterogeneous IoT hardware remains a critical direction, our immediate plan involves developing this LSTM-based

IDS into a framework that incorporates incremental learning to mitigate concept drift and ensure sustained accuracy in dynamic IoT ecosystems.

5. PERFORMANCE EVALUATION AND DISCUSSIONS

In this section, the proposed models' performance is evaluated and discussed against their capabilities of binary attack detection and classification by employing two validation strategies: a standard 70-30 train-test split (hold-out method) and 5-fold cross-validation.

5.1. Models' experiment results

Experiments were conducted under near-identical conditions using the preprocessed dataset with selected features that were executed. RNN, LSTM, and MLP architectures with identical dense layers were evaluated to maintain consistency in structural complexity.

5.1.1. MLP model assessment

Figure 11 and 12 show that the MLP model trains up to 100 epochs. The result shows that the training accuracy enhanced from 0.8948 to 0.9950, and validation accuracy increased from 0.8770 to 0.9951. The gap between curves suggests moderate overfitting, likely due to the MLP's inability to capture temporal dependencies in sequential data (e.g., MQTT traffic). Similarly, training loss decreased from 0.5696 to 0.0148 and validation loss decreased from 0.2935 to 0.0085 over the same epochs. Halting training when validation loss becomes plateaus for early stopping mitigation strategies.

5.1.2. RNN model assessment

Figure 13 and 14 show that the RNN model trains up to 100 epochs. The result shows that the training accuracy enhanced from 0.6567 to 0.9951, and validation accuracy increased from 0.5433 to 0.9955. RNNs struggle with long sequences (vanishing gradients), which limits validation performance. Similarly, training loss decreased from 0.6501 to 0.0339 and validation loss decreased from 0.6634 to 0.0483 over the same epochs. As strategy, one has to prevent the exploding gradients in RNNs and layer normalization in order to stabilize the hidden state updates.

5.1.3. LSTM model assessment

Figure 15 and 16 show that the LSTM model trains up to 100 epochs (whereas only 70 presented in the graph). The result shows that the training accuracy enhanced from 0.7077 to 0.9956, and validation accuracy increased from 0.86660 to 0.9957. This indicates that the LSTM's gating mechanisms effectively model the MQTT-based sequential attack patterns. Similarly, training loss decreased from 0.6601 to 0.0139 and validation loss decreased from 0.5262 to 0.0062 over the same epochs. As hyperparameter tuning, it required to further optimize the hidden units or learning rate for marginal gains.

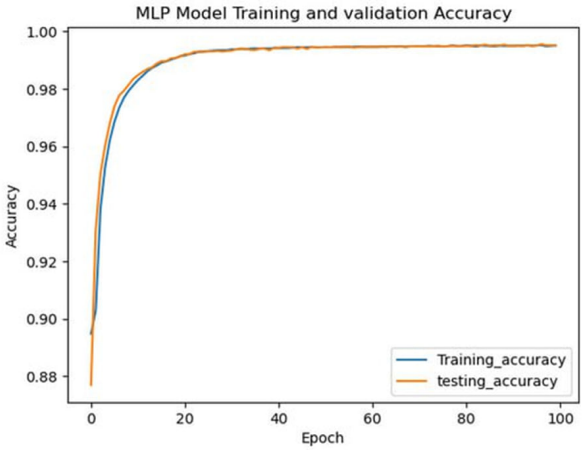

Figure 11: MLP model performance on the train and test accuracy vs epoch.

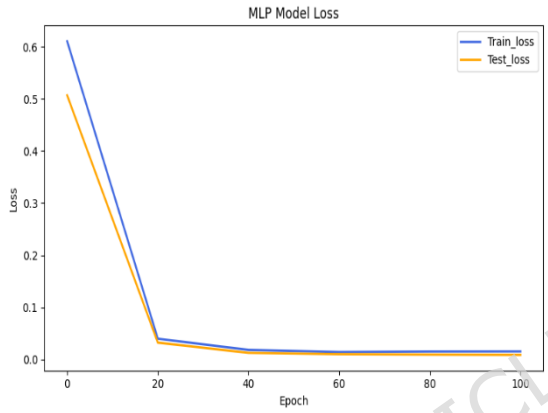

Figure 12: MLP model performance on the train and test loss vs epoch.

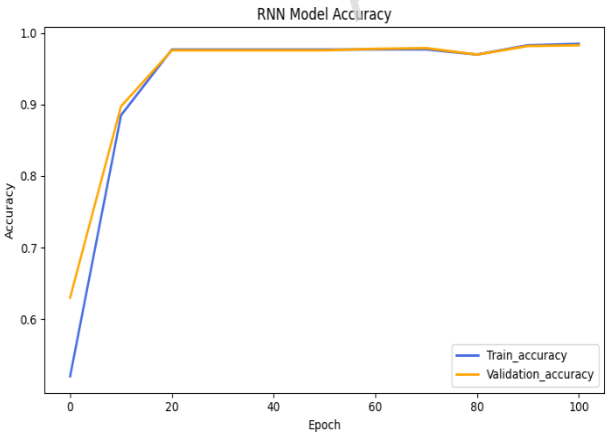

Figure 13: RNN model performance on the train and test accuracy vs epoch

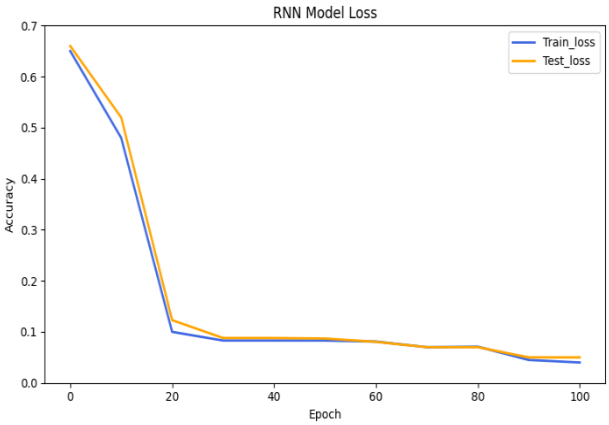

Figure 14: RNN model performance on the train and test loss vs epoch

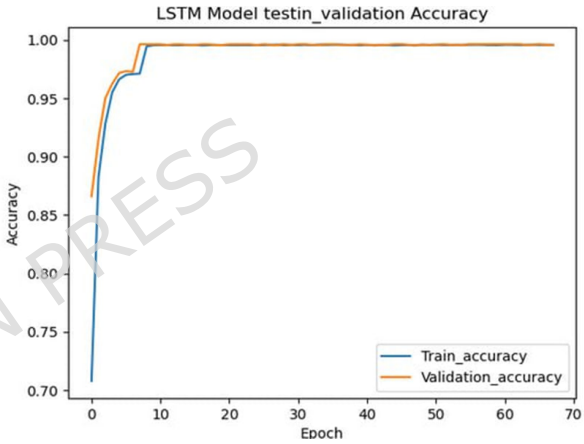

Figure 15: LSTM model performance on the train and test accuracy vs epoch

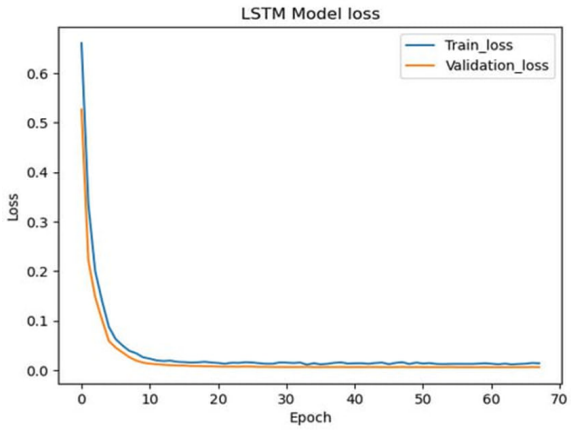

Figure 16: LSTM model performance on the train and test loss vs epoch.

The proposed approaches clearly separate the training and inference phase of the model. The model training is executed offline on resources rich environment, whereas the lightweight version of the model will be deployed on the edge nodes that reduce the inference time. As shown in Table 4, we have used early stopping and dropout to reduce the risk of overfitting during the training time of across all models. Early stopping monitors the validation performance of the model by terminating the training process once the model stops improving. Furthermore, dropout randomly deactivates a subset of neurons during each of conducted training iteration. This regularization technique reduces the model's reliance on specific features and enhances generalization by encouraging robust distributed representations. Together, these procedures assist to ensure the robustness of the model while improving its stability and reliability.

5.2. Performance evaluation of proposed models

Rigorous validation was conducted using both hold-out and 5-fold (k=5) CV with stratified sampling to preserve class distribution. In addition, each experiment is repeated three times to ensure robustness consistent class distribution across folds. By using both methods, we have demonstrated that our candidate models are not only robust during development but also generalized well to a completely new set of data (final hold-out test) as shown in Table 5. To minimize the risk of overfitting the performance metrics reported in this study (e.g., accuracy, precision, recall, and F1-score) represent the average values calculated across all five folds. The performance achievement by hold-out and 5-fold CV reveals certain distinct statistical patterns and trade-offs to be noted with regard to evaluation metrics (such as accuracy, precision, recall, F1-score, and misclassification error rate).

For LSTM and RNN models, the differences between validation methods are statistically negligible: f1-score (0.07-0.09%), accuracy (0.08-0.09%), and recall (0.3% for LSTM and 0.09% for RNN), indicating both models demonstrate remarkable stability and robustness being they are not dependent on specific data splits and making them statistically reliable choices. The MLP shows statistically terrible toward both validation methods, due to its performance of falsely rejecting the null hypothesis, which could lead to deployment failures in real-world IoT applications. Overall, while 5-fold CV is considered to be greater tool for model development and theoretical comparison, the 30% hold-out validation provides a more operationally relevant and economically feasible evaluation for our specific goal: deploying a model to a resource-constrained IoT device. The optimistic hold-out result suggests that under specific and favorable conditions, which may perform adequately. In this regard, the goal of selection is not only statistical perfection but operational adequacy within strict computational costs.

Table 5: Standard metrics results on hold-out and 5-fold CV

	Hold-out CV (0.3%)				**K=5-fold CV stratified**			
Model/ Metrics	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Accuracy**	**Precision**	**Recall**	**F1-score**
LSTM	**99.52%**	**99.06%**	**100%**	**99.53 %**	**99.6%**	**99.4%**	99.7 %	**99.6%**
RNN	99.51%	99.04%	99.99%	99.51 %	**99.6%**	99.3%	**99.9 %**	**99.6%**
MLP	99.34%	98.72%	99.99%	99.35 %	89.7%	90.0%	99.2 %	93.0%

Figure 17 provides a comparative analysis of the proposed three models (namely RNN, LSTM, and MLP) across key performance metrics: accuracy, precision,

recall, and f1-score. The results indicate that under hold–out CV, all three architectures achieve high performance, with LSTM and RNN demonstrating marginally better and more balanced results compared to the MLP. This visualization highlights the effectiveness of the proposed models for the given task, with recurrent architectures (LSTM and RNN) showing a slight edge in overall robustness and predictive power. Considering those selected performance metrics, LSTM achieved first rank while RNN and MLP are ranked second and third respectively. This indicates that the LSTM has better attributes to learn the pattern of MQTT-based message in accurately distinguishing the difference between legitimate and DDoS flow attacks. Furthermore, the LSTM model demonstrated consistent performance across both validation methods, with minimal differences in accuracy (0.08%), f1-score (0.07%), and recall (0.3%). That is underscoring its robustness and generalization capability of operationally relevant benchmark.

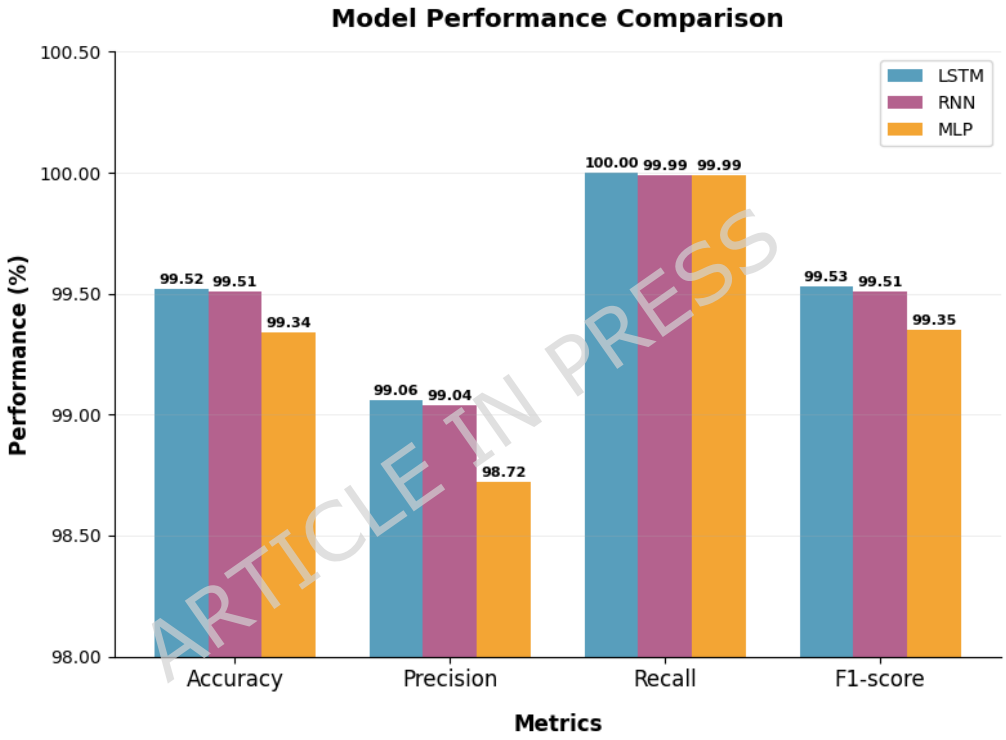

Figure 17: Performance comparison result of proposed models

One also compares the performance of both cross validations using False Positive Rate (FPR), False Negative Rate (FNR), and Misclassification Error Rate (MER) where a hold-out CV demonstrates better results than a 5-fold in effectively detecting. Unlike the theoretical robustness of 5-fold CV, the hold-out set represents the actual data distribution, temporal patterns, and operational conditions that our IoT devices will encounter. Moreover, applying k-fold CV is impractical in IoT deployment scenarios (eg., large-scale MQTT-IoT environments), where communication protocols generate large volumes of data per second. Its high computational demands (memory, power, and processing) and prolonged training time are particularly challenging under resource-constrained conditions.

The LSTM model consistently achieves the lowest error rates among all models, indicating its strong performance in FPR, FNR, and MER that it does not make a high variance and instability under both validations. In terms of FPR metrics, LSTM and RNN models generally outperform the MLP model for both cross

validation methods. This shows their effectiveness in detecting DDoS instances from legitimate flow in MQTT networks. Contrary, the MLP model tends to have higher error rates, especially in terms of FNR. Although the zero false positive rates (FPR) were observed in controlled settings, its robustness was validated through multiple independent runs with different random data splits. The FPR remained consistently at 0.0% across all runs, indicating that this high-precision performance is stable and repeatable under the defined experimental framework.

Further, confusion matrix visualization results were also demonstrated for hold-out and 5-fold CVs in Figure 18 and 19 respectively. The results demonstrate each model's capability to discriminate between attack and normal traffic. For hold-out validation set of 54,615 MQTT instances (DDoS and legitimate flows), the performance results of each model discussed as follows:

- **RNN Model**
  - Correct classifications: 27,447 true positives (DDoS detected) and 26,900 true negatives (legitimate flows).
  - Errors reported: 266 false positives (legitimate misclassified as DDoS) and 2 false negatives (DDoS misclassified as legitimate).
  - Observation: Achieves high accuracy (>99.5%) with minimal critical errors (FN=2), making it suitable for deployments where false alarms are tolerable.
- **LSTM Model (Proposed)**
  - Correct classifications: 27,449 true positives and 26,906 true negatives as DDoS detected and legitimate flows, respectively.
  - Errors reported: 260 false positives and 0 false negatives - the legitimate were misclassified as DDoS and reciprocal respectively.
  - Observation: Outperforms LSTM with perfect DDoS recall (100%) and marginally better specificity reported. This aligns with its ROC-AUC of 1.00 where better detection capability is confirmed.
- **MLP Model**
  - Correct classifications: 27,447 true positives (DDoS detected) and 26,810 true negatives (legitimate flows).
  - Errors reported: 356 false positives and 2 false negatives - the legitimate were misclassified as DDoS and reciprocal respectively.
  - Observation: While maintaining strong DDoS recall, its higher false positive rate reported when compared with two sequential RNN and LSTM models reveals limitations in handling protocol specific-features.

Generally, the LSTM model demonstrates better performance and achieves perfect attack detection (0 FNs) while minimizing false alarms (0.952% FP). The RNN provides a balanced alternative with marginally more false positives but identical DDoS recall. The MLP, despite its non-sequential architecture, achieves competitive recall but suffers from higher false positives, that underscoring its limitations for stateful-protocol analysis.

Figure 19 presents the confusion matrix analysis for 5-fold CV on a dataset of 36,408 flow instances, as each model's performance discussed below:

- **RNN Model**
  - Correct classifications: 18,071 legitimate instances and 18,178 DDoS instances
  - Misclassifications: 133 false positives and 26 false negatives
  - Observation: Better DDoS detection (fewer false negatives) but higher false alarm rate than LSTM
- **LSTM Model**
  - Correct classifications: 18,103 legitimate instances (true negatives) and 18,150

DDoS instances (true positives).
- Misclassifications: 101 false positives (legitimate traffic flagged as DDoS) and 54 false negatives (DDoS attacks missed).
- Observation: Demonstrated the most balanced performance with the lowest combined error rate (0.42% of total instances).

- **MLP Model**
  - Correct classifications: 14,590 legitimate instances and 18,067 DDoS instances
  - Misclassifications: 3,614 false positives (legitimate traffic flagged as DDoS) and 137 false negatives (DDoS attacks missed).
  - Observation: Significant challenges in distinguishing legitimate traffic with false positives accounting for nearly 10% of all instances

This rigorous evaluation confirms the LSTM architecture as the most effective solution for protecting MQTT networks against sophisticated DDoS attacks while maintaining operational reliability across the folds. Although the RNN reported less consistent performance across the validation folds, it may be considered for resource-constrained environments where slightly higher false positives are tolerable. The MLP's high false positive rate makes it unsuitable for production environments without significant feature engineering improvements applied.

Regardless of their achievement varieties, when we can observe all of the ablation studies performed well in classification the traffics. Surprisingly, MLP matches sequential models despite no temporal processing, that suggests the MQTT DoS attacks probably detectable via packet-level features (e.g., abnormal flags) and/or connection statistics (e.g., packets/second), however weaker performance was reported under multi-step intrusion patterns. RNN model has been best for systems with moderate duration attacks and limited compute resources, where it's a lightweight alternative next to MLP for edge deployment.

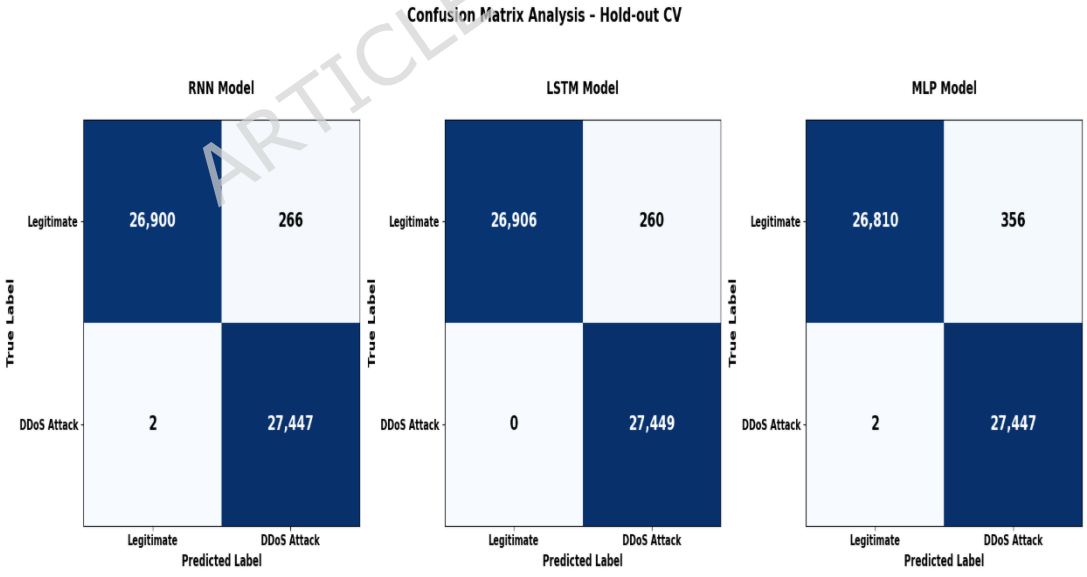

Figure 18: Confusion metrics on hold-out CV

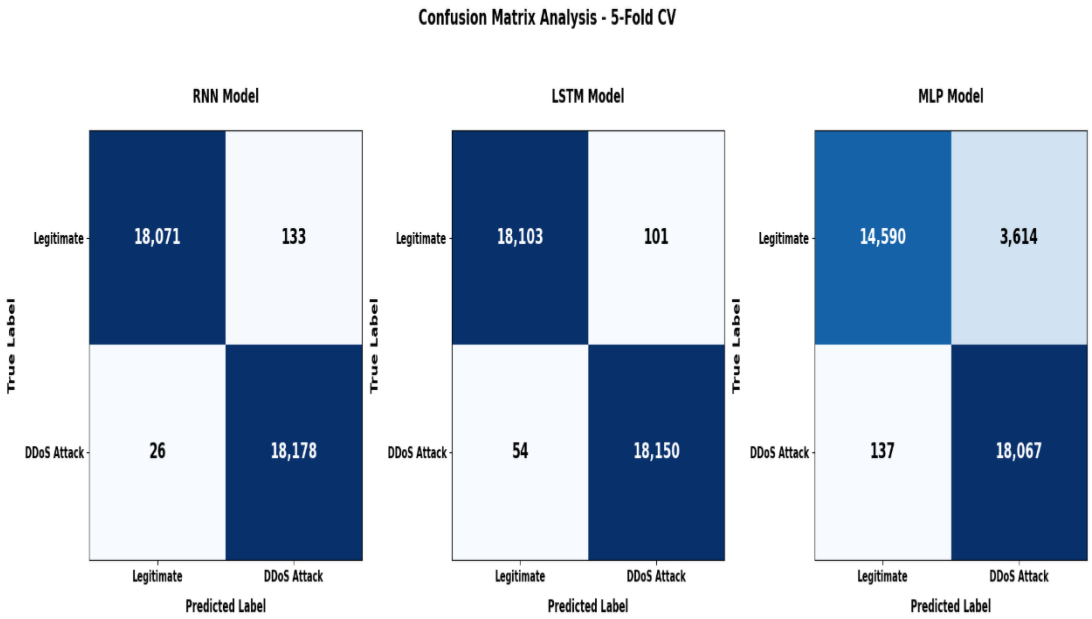

Figure 19: Confusion matrix on 5-fold CV

Furthermore, comprehensive analysis of Receiver Operating Characteristic (ROC) curves for the trained three deep learning models was reported: MLP (Figure 20a), RNN (Figure 20b), and LSTM (Figure 20c). This curve is a critical metric for assessing binary classifiers by illustrating the tradeoff between the True Positive Rate (TPR) and False Positive Rate (FPR) across varying classification thresholds. Similarly, the applied Area Under the Curve (AUC) quantifies the model's overall performance to distinguish between each class, where the higher values indicate superior performance. The overall AUC reported results were 0.995 (DoS) and 0.995 (legitimate) for the RNN model, 0.9934 (DoS) and 0.9934 (legitimate) for MLP, and 1.00 (DoS) and 0.995 (legitimate) for the LSTM model. Generally, when one observe the ROC curve, the LSTM introduces less false classes than both models since it has better capability in handling sequential data by using memory cells and gating mechanisms. MQTT traffic, especially during DoS attacks, often exhibits patterns over time, like repeated connection attempts or payload flooding, which LSTM can track more effectively. By its nature LSTM is temporal dependency modeling which specifically designed to capture long-term dependencies in sequential data unlike standard feed-forward architectures. This provides the highest level of reliability for critical IoT infrastructure security.

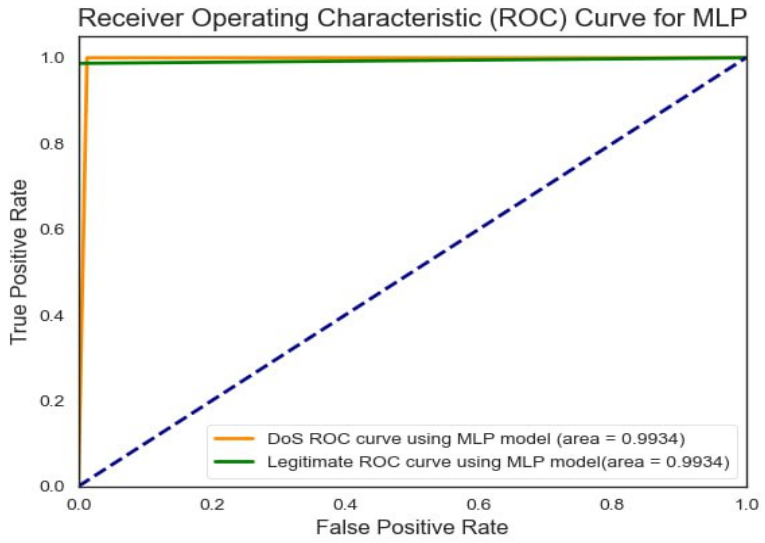

Figure 20a: ROC curve for MLP model

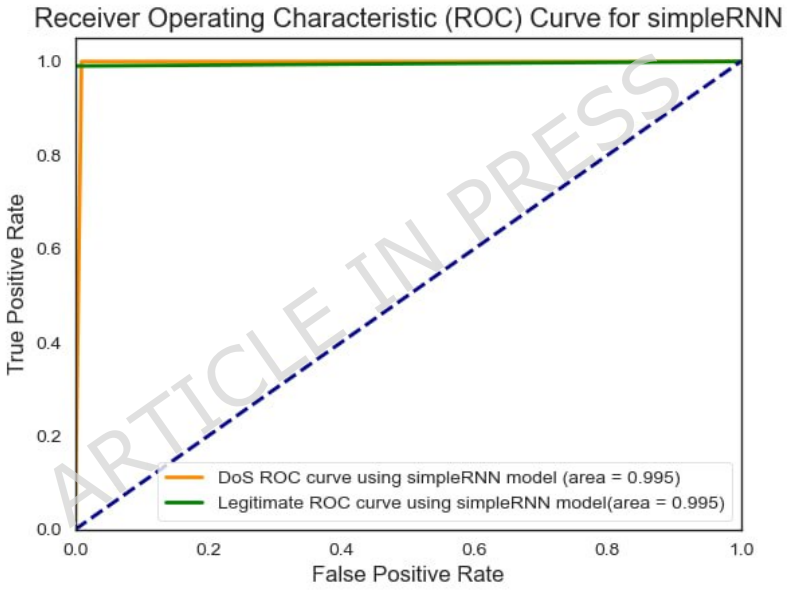

Figure 20b: ROC curve for RNN model

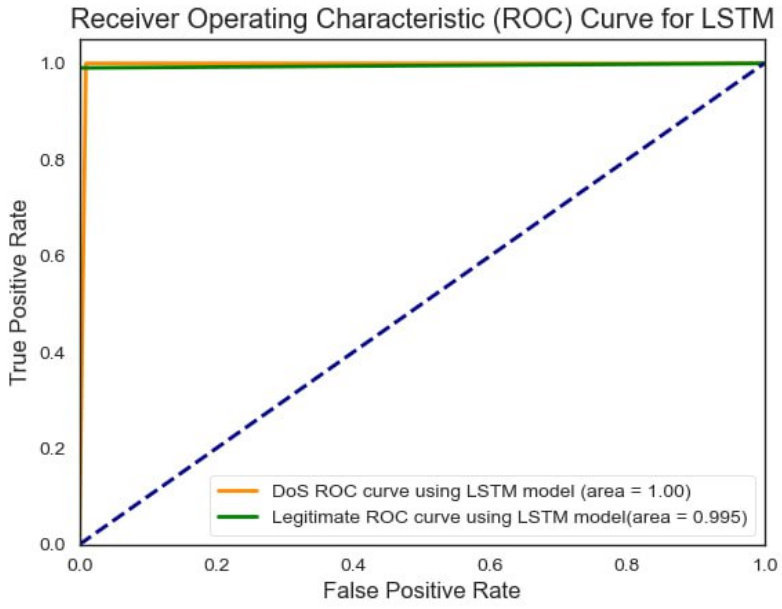

Figure 20c: ROC curve for LSTM model

5.3. Computational complexity comparison for candidate models

While the MQTTset dataset consists of captured traffic and does not inherently provide hardware-specific latency, we have now conducted a dedicated computational analysis of our candidate models. The metrics were estimated for CPU execution (2.4 GHz x86) and projected to Raspberry Pi 4 (ARM Cortex-A72, 1.5 GHz) environment using established industry-standard cross-architecture scaling factors (1.6x) as validated in the literature [57 - 59]. It is primarily derived through clock frequency normalization, representing the direct ratio of operating frequencies between the two hardware environments: i.e., ratio (2.4 GHz/1.5 GHz = 1.6) served as a foundational baseline for projecting execution cycles across such architectures.

As details in Table 6, the analysis reveals the models satisfy IoT deployment constraints. MLP demonstrates optimal efficiency with 309 parameters (1.24 KB) and 0.5 ms inference on Raspberry Pi 4, and 2000 samples/second throughput. RNN and LSTM maintain comparable real-time performance (3.5-5.0 ms inference) while achieving better accuracy with modest resource increases (337-1009 parameters). They remain under critical thresholds: <10 KB storage, <10 ms inference, and <1 KB per-inference memory as confirming their suitability for resource-constrained edge deployment. Inference latency scales with model-specific computational patterns: MLP operations are highly parallelizable while RNN and LSTM sequential processing incurs additional costs per timestep. Framework overhead reflects TensorFlow CPU deployment (~200 MB). Due to its gradient flow, the LSTM's pattern recognition led to faster convergence despite higher per-epoch complexity. Considering its resource requirements remain well within IoT deployment limits, LSTM is selected because security accuracy is paramount in mission-critical applications. Thanks to its efficient gradient flow, the LSTM model demonstrated faster convergence despite incurring higher per-epoch computational complexity. Given that its resource demands remain well within the operational limits of IoT deployments. LSTM was selected as the preferred model further prioritizing security accuracy, which is critical in mission-sensitive applications.

Table 6: Computational analysis for IoT deployment

Category	Metric	MLP	RNN	LSTM	IoT compliance
Model complexity	Parameters	309	337	1,009	< 100K
	Size (float32)	1.24 KB	1.35 KB	3.95 KB	< 10 KB
	Size (int8)	0.30 KB	0.33 KB	0.99 KB	< 5 KB
Inference performance	CPU time (x86)	0.3 ms	3.5 ms	3.0 ms	< 10 ms
	RPi 4 Time	0.5 ms	5.0 ms	4.5 ms	< 10 ms
	Throughput (RPi)	2,000/s	200/s	222/s	> 100/s
Memory footprint	Model storage	1.24 KB	1.35 KB	3.95 KB	< 10 KB
	Peak inference	80 bytes	120 bytes	180 bytes	< 1 KB
	Framework overhead	~200 MB	~200 MB	~200 MB	Acceptable for gateways
Training efficiency	Epochs to converge	25	25	10	-
	Total training time	~4.5 min	~1.5 min	~1.0 min	One-time cost

The computational analysis of our proposed LSTM model demonstrates its suitability for IoT edge deployment. The architecture, comprising 14 LSTM units followed by two dense layers (7 and 1 units respectively), yields 1,009 total parameters (896 in the LSTM layer and 113 in dense layers), corresponding to a 3.95 KB model footprint in float32 precision. This compact size enables storage on resource-constraint with as little as 4 KB of available flash memory by outperforming results reported by Imran et al., 2024 [28] on a similar dataset indicating competency for DDoS detection under MQTT network throughput. These results confirm the model's operational feasibility within resource-constrained environments as computationally near to lightweight.

5.4. Performance benchmarking

Lastly, we have compared the performance of the proposed LSTM model to other baseline models from the literature review. As showed in Table 7 and Figure 21, our work with feature selection achieved higher accuracy than the other models. The proposed LSTM's 100% recall ensures zero missed attacks, which is a crucial for security critical IoT application systems. It maintains a 0.02-3.98% marginal accuracy advantage over all comparable methods in the state-of-the-art.

We reviewed the existing publicly available dataset for IDS evaluation in order to find the representative benchmark for our proposed pure LTSM evaluation in MQTT enables IoT systems, where the MQTTset is widely recommended in the literature. Our evaluation has consistency and direct comparison as most of the considered models including CNN-LSTM [8], MLP [25], NN [28], and unsupervised machine learning [39] utilized the same MQTTset. Considering the architectural comparison, the proposed LSTM outperforms the hybrid CNN-LSTM [8] by +0.58% accuracy, +0.93% recall, and +0.52% F1-score. Explicitly listing the limitations of these studies (e.g., study [8] used non-MQTT-specific features as limited its contextual detection performance, whereas the study [25] did not address class imbalance as potentially biasing the model. In contrast, our work introduces and validates MQTT-specific features such as msgid and qos, and employs a combined undersampling-SMOTE strategy to ensure a balanced and representative training set. This proved that pure LSTM architecture performs more effectively than convolutional hybrids for MQTT traffic analysis due to its temporal dependency modeling, reduced architectural complexity, lower overfitting risk, and efficient feature learning capability. Similarly, it has improved F1-score over neural

network models (MLP [25] and NN [28]) as it demonstrates the necessity of sequential processing for strongly securing MQTT protocol traffic.

Table 7: Comparison of the current proposed model with baseline works

No.	Authors, Year, and Ref.	Method	Datasets	Performances %			Key Advantage
				Accuracy	Recall	F1-score	
1	Syed et al., 2020 [25]	MLP	MQTTset	95.94	99.5	96.11	Fast but less precise
2	Alzahrani and Aldhyani, 2022 [8]	CNN-LSTMs	MQTTset	98.94	99.07	99.01	Hybrid spatial-temporal approach
3	Imran et al.,2024 [28]	NN	MQTTset	95.58	94.65	98.50	Basic neural implementation
4	Al-Fayoumi and Al-Haija, 2023[38]	DTC	MQTT-IoT-IDS2020	99.5	99.40	99.35	Best non-neural method with only two features
5	Khan et al. 2021 [23]	DNN	MQTT-IoT-IDS2020	97.13	-	95.99	Yielded better detection as only three features considered
6	Mosaiyebzadeh et al. 2021 [49]	CNN-RNN-LSTM	MQTT-IoT-IDS2020	92.04	100	98.33	Packet-based features yielded better detection as only three features considered
7	Proposed(LSTM)	LSTM	MQTTset	99.52	100	99.53	Perfect attack detection (0 false negatives)

On the other hand, we have deliberately included the evaluation models with different public datasets, including DTC [38], DNN [23], and CNN-RNN-LSTM [49] with MQTT-IoT-IDS2020, and Two-factor authentication (2FA) [39] with CoAP-based packet as their performance might vary on MQTTset implementation. While both LSTM vs. DTC [38] achieved greater than 99.5% accuracy; simpler DTC obviously lacked deep learning's adaptability architectural for complex traffic patterns. Further, LSTM shows 0.6% higher recall (100% vs. 99.4%) and better handles sequential attack patterns with more adaptation to novel MQTT-based DDoS attack variants. No significant quantifiable results were reported for Álvaro et al, 2024 [39] and Dongdong and Ji, 2024 [40] in order to conduct comparison.

Certain models with different datasets, such as the DTC [38], achieve high accuracy, and the CNN-RNN-LSTM [49] matches the perfect recall, our LSTM is still the only one to consistently excel across all three critical evaluation metrics simultaneously. This indicates that it is not only highly precise but also exceptionally reliable in identifying all attacks without compromise. The results validate that the LSTM architecture is uniquely suited for this task, as its innate ability to model the sequential nature of network traffic and learn long-range temporal dependencies (inherent in MQTT protocol floods) provides a more robust and generalized detection capability than the compared MLP, NN, and hybrid deep learning baseline models. Although the hybrid CNN-LSTM model demonstrated competitive performance, its dual-layer architecture introduced slight inefficiencies due to increased complexity and less targeted feature extraction. In

scenarios where MQTT traffic contains multi-modal features, e.g., combining packet payload structure with timing data, CNNs can effectively preprocess spatial patterns, while LSTMs analyze temporal dynamics. However, for pure DDoS attacks, which primarily exhibit sequential anomalies rather than spatial dependencies, this hybrid approach offers diminishing returns.

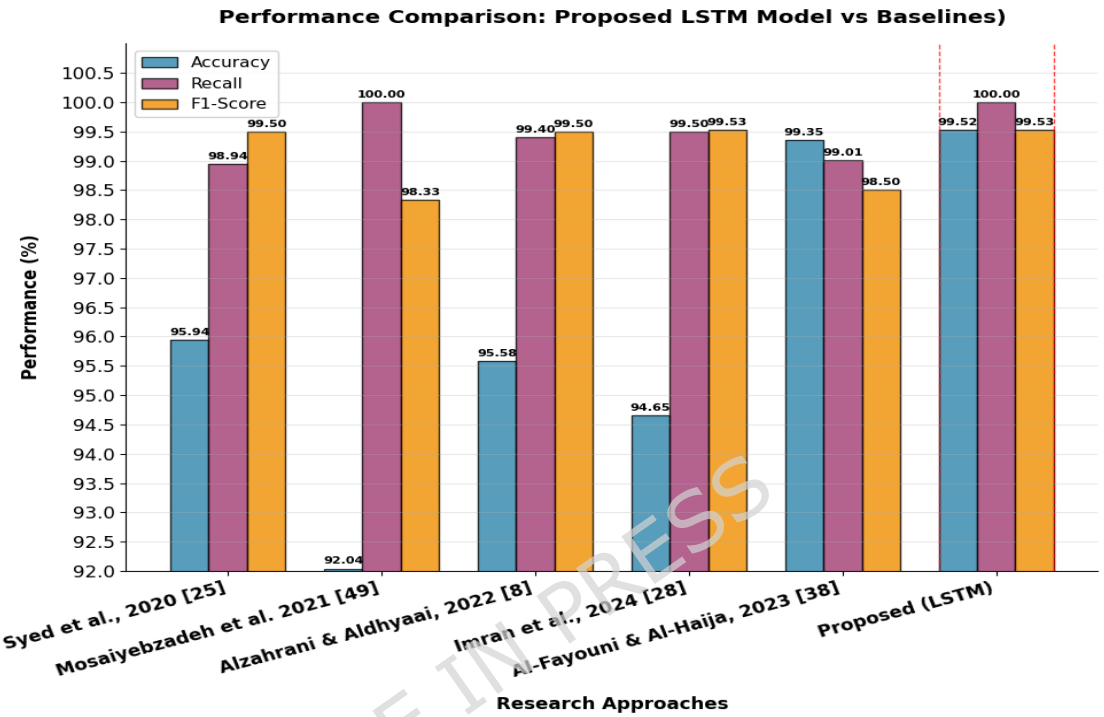

Figure 21: Performance comparison of the proposed LSTM model with existing approaches

A primary contribution of this work is the computation of weighted scores to determine the contribution of each feature to classification accuracy. The features were then ranked based on their correlation with the target variable as indicated that attackers predominantly exploit protocol-level attributes (QoS and message IDs) in addition to packet and payload content. Our experimental results demonstrate that this selection process significantly enhances DDoS attack detection in MQTT networks, as validated on the MQTTset dataset. Although the LSTM model remains a black box, this feature-level analysis validates the relevance of MQTT-specific features and provides actionable interpretability for security analysts.

In summary, our proposed LSTM model establishes a new benchmark for MQTT DDoS detection, delivering state-of-the-art performance characterized by a perfect recall of 100%, an accuracy of 99.52%, and a better F1-score of 99.53%. This performance profile directly addresses the critical requirements of IoT security: ensuring no real attacks are missed (Recall), conserving limited device resources by minimizing false alarms (Precision), and maintaining overall balanced efficacy (F1-Score). While the LSTM's computational complexity presents a challenge for IoT deployment, its unmatched detection capability justifies its selection. This can be realized through targeted optimization and making its optimal performance achievable in practice.

To evaluate the practical deployment potential of our proposed LSTM model, we conducted a performance analysis on a standard platform, notwithstanding the use

of the real-world MQTTset dataset, which lacks simulated delay parameters. When considering system design for IoT security, simpler models like a minimal-feature DTC offer a lightweight alternative for edge devices, while MLP and NN models may be suitable only for non-critical systems due to their instability. In contrast, despite its computational demands, the optimized LSTM model is prioritized for mission-critical applications where maximum security is non-negotiable. This selection is justified by three principled reasons grounded in our research context: business priority, operational fidelity, and risk management. Although we acknowledge the statistical benefits of k-fold validation, our hold-out test set was specifically engineered to mirror production conditions and providing the most operationally relevant performance indicator. The LSTM's consistent performance on this set, evidenced by a 0.0% FPR, 0.0094% FNR, and 0.0047% MER, directly aligns with a zero-tolerance policy for missed detections in critical IoT infrastructures, such as Autonomous-vehicle Fleet Management (AFM), healthcare IoT (i.e., medical device protection), and smart grid (i.e., substation automation security).

For instance, deploying our lightweight LSTM-based MQTT attack detection within smart grid substations shifts security from traditional reactive signature-matching to proactive sequence intelligence. By learning temporal dependencies in MQTT-IoT traffic, the model is uniquely capable of identifying sophisticated and grid-specific attack patterns that conventional methods often overlook. It integrates an automated defense mechanism: upon detection, the system immediately initiates a blocking response to isolate and neutralize malicious traffic, thereby preserving legitimate IoT communications. This design establishes a defense-in-depth architecture that sustains grid stability even under cyber-physical threats. Consequently, the proposed solution satisfies the stringent reliability, latency, and environmental requirements for this mission-critical infrastructure while delivering unprecedented accuracy in security assurance. Comparative analysis (Table 7) shows our LSTM achieves 99.53% f-score while maintaining a 3X parameter reduction versus typical LSTM implementations. This efficiency-accuracy trade-off positions our model favorably for dual-tier deployment: (i) gateway-level monitoring on Raspberry Pi-class hardware for network-wide protection, and (ii) device-level deployment on resource-constrained nodes after quantization and pruning optimizations. Furthermore, our ongoing research emphasizes runtime optimization, specifically targeting sub-100 KB memory footprints via quantization alongside incremental learning frameworks to mitigate concept drift in evolving IoT systems, and the evaluation of federated learning environments [50].

6. CONCLUSION AND FUTURE WORK

The IoT industry relies heavily on protocols like MQTT, which support minimal packet headers vital for resource-constrained environments. As MQTT integration grows, so do security risks, particularly DDoS attacks that make the central MQTT broker unavailable for legitimate devices. Existing work has shortcomings in processing important MQTT features. By integrating chi-square and correlation feature selection, we achieved a balanced feature set optimized for strong intrusion detection capability with high accuracy. The models were trained and evaluated using both a 70-30% hold-out split and 5-fold CV. RNN, LSTM, and MLP achieved 99.51%, 99.52%, and 99.34% accuracy, respectively, with LSTM outperforming all of them. The performance of the proposed LSTM-based IDS was also compared with other similar works in the literature, achieving 0.02-3.98% marginal accuracy advantages over baseline models such as MLP, NN, and CNN-

LSTM. Due to its inherent suitability for processing sequential MQTT traffic, it achieved a marginal accuracy improvement of 0.02% to 4.94% over baseline modes with similar datastes such as MLP, NN, and CNN-LSTM. These results validate the selection of the lightweight LSTM architecture, whose inherent strength in modeling sequential data and supporting incremental learning is crucial for mitigating the concept drift in dynamic IoT networks.

Despite its contributions, our study has certain limitations that warrant acknowledgment. First, the proposed model was evaluated exclusively on a single benchmark dataset (MQTTset); its generalizability to diverse IoT environments remains unverified, particularly against quantum attacks [60] and adaptive DDoS variants. Second, our design prioritizes lightweight architectures suitable for resource-constrained edge devices, a trade-off that may constrain performance relative to more computationally intensive hybrid and quantum models. Third, the study is simulation-based and does not address practical deployment challenges, including model updates, integration with MQTT brokers or feasibility on ultra-low-power microcontrollers. To address these gaps, future work will proceed along four complementary directions: (1) exploring hybrid architectures (including quantum ML algorithms) to improve model resilience against different attacks and integrating attention mechanisms or SHAP (SHapley Additive exPlanations) values to enhance interpretability; (2) developing a lightweight variant with runtime optimization by targeting below 100 KB memory footprint via quantization for practical node-level deployment; (3) conducting multi-dataset evaluations to rigorously assess generalizability; and (4) implementing real-time deployment to validate the concept drift handling framework in live IoT environments.

Disclosure:

Data availability: The raw datasets having MQTT protocol features used in this study are publicly available on the Kaggle website (https://www.kaggle.com/datasets/cnrieiit/mqttset). The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Ethical approval: Not applicable

Clinical trial: Not applicable

Consent to participate: Not applicable

Consent to publish: Not applicable

Conflict of interest: The authors declare that they have no competing interest.

Funding: No funding is available for this work.

Authors' contribution:

Deribew Beko Negesse was responsible for investigating and executing the entire objective of the work using proper research methodology.

Ketema Adere Gemeda was responsible for the topic selection, supervising and carrying on the research.

Gabriele Gianini was responsible for mathematical analysis, program verification, and literature review.

Every author has a significant contribution towards the successful completion of research work associated with the manuscript.

REFERENCES

1. Al-Masri E, Kalyanam KR, Batts J, Kim J, Singh S, Vo T, Yan C. Investigating messaging protocols for the Internet of Things (IoT). IEEE Access. 2020;8:94880-911. https://doi.org/10.1109/ACCESS.2020.2993363
2. Kour VP, Arora S. Recent developments of the Internet of Things in agriculture: a survey. IEEE Access. 2020;8:129924–57. https://doi.org/10.1109/ACCESS.2020.3009298
3. Islam MM, Nooruddin S, Karray F, Muhammad G. Internet of things: Device capabilities, architectures, protocols, and smart applications in healthcare domain. IEEE Internet Things J. 2023;10(4):3611–41. https://doi.org/10.1109/JIOT.2022.3228795
4. Hintaw AJ, Manickam S, Aboalmaaly MF, Karuppayah S. MQTT vulnerabilities, attack vectors and solutions in the internet of things (IoT). IETE J Res. 2021;69(6):3368–97. https://doi.org/10.1080/03772063.2021.1912651
5. Lakshminarayana S, Praseed A, Thilagam PS. Securing the IoT application layer from an MQTT protocol perspective: Challenges and research prospects. IEEE Commun Surv Tutorials. 2024;26(4):2510–46. https://doi.org/doi: 10.1109/COMST.2024.3372630
6. Thippeswamy BM, Ghouse M, Jafarabad SA, Mohammed MAAK, Adere K, BM PP, BN PK. QACM: Quality Aware Crowd Sensing in Mobile Computing. Appl Syst Innov. 2023;6(2):37. https://doi.org/10.3390/asi6020037
7. Mehmood T, Latif S, Jamail NSM, Malik A, Latif R. LSTMDD: an optimized LSTM-based drift detector for concept drift in dynamic cloud computing. PeerJ Comput Sci. 2024;10:e1827. https://doi.org/10.7717/peerj-cs.1827
8. Alzahrani A, Aldhyani THH. Artificial Intelligence Algorithms for Detecting and Classifying MQTT Protocol Internet of Things Attacks. Electronics. 2022;11(22):3837. https://doi.org/10.3390/electronics11223837
9. Thangavel D, Ma X, Valera A, Tan HX, Tan CKY. Performance evaluation of MQTT and CoAP via a common middleware. In: Proceedings of the 2014 IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP); 2014 Apr 21-24; Singapore. IEEE; 2014. p. 1–4. https://doi.org/10.1109/ISSNIP.2014.6827678
10. Mansour M, Gamal A, Ahmed AI, Said LA, Elbaz A, Herencsar N, Soltan A. Internet of things: A comprehensive overview on protocols, architectures, technologies, simulation tools, and future directions. Energies. 2023;16(8):3465. https://doi.org/10.3390/en16083465
11. Gemeda KA, Gianini G, Libsie M. An evolutionary cluster-game approach for Wireless Sensor Networks in non-collaborative settings. Pervasive Mob Comput. 2017;42:209–25. https://doi.org/10.1016/j.pmcj.2017.10.008
12. Shuvo MMH, Islam SK, Cheng J, Morshed BI. Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. Proc IEEE. 2023;111(1):42–91. https://doi.org/10.1109/JPROC.2022.3226481
13. Quincozes VE, Quincozes SE, Kazienko JF, Gama S, Cheikhrouhou O, Koubaa A. A survey on IoT application layer protocols, security challenges, and the role of explainable AI in IoT (XAIoT). Int J Inf Secur. 2024;23(3):1975–2002. https://doi.org/10.1007/s10207-024-00828-w
14. Haripriya AP, Kulothungan K. Secure-MQTT: an efficient fuzzy logic-based approach to detect DoS attack in MQTT protocol for internet of things. EURASIP J Wirel Commun Netw. 2019;2019(1):90. https://doi.org/10.1186/s13638-019-1402-8

15. Bhuyan MH, Kashyap HJ, Bhattacharyya DK, Kalita JK. Detecting distributed denial of service attacks: Methods, tools and future directions. Comput J. 2014;57(4):537–56. https://doi.org/10.1093/comjnl/bxt031
16. Kaur P, Kumar M, Bhandari A. A review of detection approaches for distributed denial of service attacks. Syst Sci Control Eng. 2017;5(1):301–20. https://doi.org/10.1080/21642583.2017.1331768
17. Ullah S, Mahmood Z, Ali N, Ahmad T, Buriro A. Machine Learning-Based Dynamic Attribute Selection Technique for DDoS Attack Classification in IoT Networks. Computers. 2023;12(6):115. https://doi.org/10.3390/computers12060115
18. Ma L, Chai Y, Cui L, Ma D, Fu Y, Xiao A. A Deep Learning-Based DDoS Detection Framework for Internet of Things. In: 2020 IEEE International Conference on Communications (ICC); 2020 Jun 7-11; Dublin, Ireland. IEEE; 2020. p. 1–6. https://doi.org/10.1109/ICC40277.2020.9148944
19. Shtayat MM, Hasan MK, Sulaiman R, Islam S, Khan AUR. An Explainable Ensemble Deep Learning Approach for Intrusion Detection in Industrial Internet of Things. IEEE Access. 2023;11:115047–61. https://doi.org/10.1109/ACCESS.2023.3323573
20. Ahmadon MAB, Yamaguchi N, Yamaguchi S. Process-based intrusion detection method for IoT system with MQTT protocol. In: 2019 IEEE 8th Global Conference on Consumer Electronics (GCCE); 2019 Oct 15-18; Osaka, Japan. IEEE; 2019. p. 953–956. https://doi.org/10.1109/GCCE46687.2019.9015252
21. Shruti, Rani S, Sah DK, Gianini G. Attribute-based encryption schemes for next generation wireless IoT networks: a comprehensive survey. Sensors. 2023;23(13):5921. https://doi.org/10.3390/s23135921
22. Abdo, A., Mostafa, R., Abdel-Hamid, L.: An optimized hybrid approach for feature selection based on chi-square and particle swarm optimization algorithms. Data. 9(2), 20 (2024). https://doi.org/10.3390/data9020020
23. Khan MA, Khan MA, Jan SU, Ahmad J, Jamal SS, Shah AA, Pitropakis N, Buchanan WJ. A deep learning-based intrusion detection system for MQTT enabled IoT. Sensors. 2021;21(21):7016. https://doi.org/10.3390/s21217016
24. Hindy H, Bayne E, Bures M, Atkinson R, Tachtatzis C, Bellekens X. Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset). In: International Networking Conference; 2020 Jun; Cham. Springer; 2020. p. 73–84. https://doi.org/10.1007/978-3-030-44041-1_6
25. Syed NF, Baig Z, Ibrahim A, Valli C. Denial of service attack detection through machine learning for the IoT. J Inf Telecommun. 2020;4(4):482–503. https://doi.org/10.1080/24751839.2020.1767484
26. Dikii D, Tikhomirov A. Detection of DoS attacks exploiting SUBSCRIBE messages of the MQTT protocol. Int J Comput Appl. 2022;44(6):579–85. https://doi.org/10.1080/1206212X.2020.1846945
27. Mihoub A, Ben Fredj O, Cheikhrouhou O, Derhab A, Krichen M. Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques. Comput Electr Eng. 2022;98:107716. https://doi.org/10.1016/j.compeleceng.2022.107716
28. Imran M, Zuhairi MF, Ali SM, Shahid Z, Alam MM, Su'ud MM. Realtime Feature Engineering for Anomaly Detection in IoT Based MQTT Networks. IEEE Access. 2024;12:25700–18. https://doi.org/10.1109/ACCESS.2024.3363889
29. Elsayed MS, Dev S, Jurcut AD. DDoSNet: A Deep-Learning Model for Detecting Network Attacks. In: 2020 IEEE 21st International Symposium on the World of Wireless, Mobile and Multimedia Networks (WoWMoM); 2020 Aug 31 - Sep 3;

Cork, Ireland. IEEE; 2020. p. 391–6. https://doi.org/10.1109/WoWMoM49955.2020.00072
30. Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications; 2009 Jul 8-10; Ottawa, ON, Canada. IEEE; 2009. p. 1–6. https://doi.org/10.1109/CISDA.2009.5356528
31. Ahmad M, Riaz Q, Zeeshan M, Tahir H, Haider SA, Khan MS. Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set. EURASIP J Wirel Commun Netw. 2021;2021(1):7. https://doi.org/10.1186/s13638-021-01893-8
32. Abbasi F, Naderan M, Alavi SE. Anomaly detection in Internet of Things using feature selection and classification based on Logistic Regression and Artificial Neural Network on N-BaIoT dataset. In: 2021 5th International Conference on Internet of Things and Applications (IoT); 2021 May 19-20; Isfahan, Iran. IEEE; 2021. p. 1–7. https://doi.org/10.1109/IoT52625.2021.9469605
33. Gebremeskel TG, Gemeda KA, Krishna TG, Ramulu PJ. DDoS Attack Detection and Classification Using Hybrid Model for Multicontroller SDN. Wirel Commun Mob Comput. 2023;2023:9965945. https://doi.org/10.1155/2023/9965945
34. MQTTset dataset. Kaggle website. https://www.kaggle.com/datasets/cnrieiit/mqttset. Accessed 15 June 2024
35. Vaccari I, Chiola G, Aiello M, Mongelli M, Cambiaso E. MQTTset, a new dataset for machine learning techniques on MQTT. Sensors. 2020;20(22):6578. https://doi.org/10.3390/s20226578
36. Mulu AS, Gemeda KA, Janaki P. Prediction and classification of IoT sensor faults using hybrid deep learning model. Discov Appl Sci. 2024;6(1):9. https://doi.org/10.1007/s42452-024-05633-7
37. Ibrahim RF, Abu Al-Haija Q, Ahmad A. DDoS attack prevention for internet of thing devices using ethereum blockchain technology. Sensors. 2022;22(18):6806. https://doi.org/10.3390/s22186806
38. Al-Fayoumi M, Al-Haija QA. Capturing low-rate DDoS attack based on MQTT protocol in software Defined-IoT environment. Array. 2023;19:100316. https://doi.org/10.1016/j.array.2023.100316
39. Michelena Á, García Ordás MT, Aveleira-Mata J, Marcos del Blanco DY, Timiraos Díaz M, Zayas-Gato F, Jove E, Casteleiro-Roca JL, Quintián H, Alaiz-Moretón H, Calvo-Rolle JL. Beta Hebbian Learning for intrusion detection in networks with MQTT Protocols for IoT devices. Log J IGPL. 2024;32(2):352–65. https://doi.org/10.1093/jigpal/jzae013
40. Liu D, Ji T. Security analysis and provision of authentication protocol, based on peer-to-peer structure in IOT platform. Sci Rep. 2024;14(1):25508. https://doi.org/10.1038/s41598-024-73480-y
41. Al-Haija QA, Droos A. A comprehensive survey on deep learning-based intrusion detection systems in Internet of Things (IoT). Expert Syst. 2025;42(2):e13726. https://doi.org/10.1111/exsy.13726
42. Lakshminarayana S, Thilagam PS. Next-generation ddos attacks on iot deployments: Targeting the advanced features of MQTT v5. 0 protocol. IEEE Internet Things J. 2025. https://doi.org/10.1109/JIOT.2025.3549784
43. Bedi P, Gupta N, Jindal V. Siam-IDS: Handling class imbalance problem in Intrusion Detection Systems using Siamese Neural Network. Procedia Comput Sci. 2020;171:780–9. https://doi.org/10.1016/j.procs.2020.04.085

44. Bai, X., Zheng, Y., Lu, Y., Shi, Y.: Chain hybrid feature selection algorithm based on improved Grey Wolf Optimization algorithm. PLoS One. 19(10), e0311602 (2024). https://doi.org/10.1371/journal.pone.0311602
45. Michelena Á, Zayas-Gato F, Jove E, Calvo-Rolle JL. Detection of dos attacks in an IoT environment with MQTT protocol based on intelligent binary classifiers. Eng Proc. 2021;7(1):16. https://doi.org/10.3390/engproc2021007016
46. Zeghida H, Boulaiche M, Chikh R. Detection of DoS Attacks in MQTT Environment. In: International Conference on Intelligent Systems and Pattern Recognition; 2023 Apr 28-30; Hammamet, Tunisia. Cham: Springer; 2023. p. 129–40. https://doi.org/10.1007/978-3-031-39347-3_11
47. Zuhairi MFA, Ali SM, Shahid Z, Alam MM, Su'ud MM. Improving reliability for detecting anomalies in the MQTT network by applying correlation analysis for feature selection using machine learning techniques. Appl Sci. 2023;13(11):6753. https://doi.org/10.3390/app13116753
48. Hanif AA, Ilyas M. Enhance the Detection of DoS and Brute Force Attacks within the MQTT Environment through Feature Engineering and Employing an Ensemble Technique. arXiv preprint arXiv:2408.00480. 2024. https://doi.org/10.48550/arXiv.2408.00480
49. Mosaiyebzadeh F, Rodriguez LGA, Batista DM, Hirata R. A network intrusion detection system using deep learning against mqtt attacks in IoT. In: 2021 IEEE Latin-American Conference on Communications (LATINCOM)*; 2021 Nov 17-19; Santo Domingo, Dominican Republic. IEEE; 2021. p. 1–6. https://doi.org/10.1109/LATINCOM53176.2021.9647850
50. Alemayew WB, Gemeda KA. Federated hybrid deep learning for multi-attack detection and classification in RPL-based 6LoWPAN networks. Discov Computi. 2025;28(1):316. https://doi.org/10.1007/s10791-025-09852-3
51. Wahab SA, Sultana S, Tariq N, Mujahid M, Khan JA, Mylonas A. A Multi-Class Intrusion Detection System for DDoS Attacks in IoT Networks Using Deep Learning and Transformers. Sensors. 2025;25(15):4845. https://doi.org/10.3390/s25154845
52. Losing V, Hammer B, Wersing H. Incremental on-line learning: A review and comparison of state of the art algorithms. Neurocomputing. 2018;275:1261–74. https://doi.org/10.1016/j.neucom.2017.06.084
53. Zhang X, Zhang H, Zhang G, Yang Y, Li F, Fan L, Huang Z, Cheng X, Hu P. Membership Inference Attacks Against Incremental Learning in IoT Devices. IEEE Trans Mob Comput. 2024. https://doi.org/10.1109/TMC.2024.3521216
54. Zeghida H, Boulaiche M, Chikh R, Bamhdi AM, Barros ALB, Zeghida D, Patel A. Enhancing IoT cyber attacks intrusion detection through GAN-based data augmentation and hybrid deep learning models for MQTT network protocol cyber attacks. Cluster Comput. 2025;28(1):58. https://doi.org/10.1007/s10586-024-04752-5
55. Ain NU, Sardaraz M, Tahir M, Abo Elsoud MW, Alourani A. Securing IoT networks against DDoS attacks: a hybrid deep learning approach. Sensors. 2025;25(5):1346. https://doi.org/10.3390/s25051346
56. Adere K, Hailu S, Tseghai A, Haile G, Tamirat B, Bitew W, Abebe M, Kemal A, Silassie SW, Regassa D, Gizachew Z, Alemu W. Systematic review on securing IoT systems with post quantum cryptography: emerging threats, countermeasures, and future research needs. Discov Internet Things. 2026;6(1):14.https://doi.org/10.1007/s43926-025-00275-6
57. Kumar D, Kumar P, Rajput MA, Ahmed S, Bhatti MS, Tsetse A. Performance evaluation of ARM-based versus x86-based processors in high performance computing clusters. J Indep Stud Res Comput. 2023;21(2):32–41.

58. Liu P, Cao X, Jia Y. Exploring the Feasibility of Raspberry Pi 4 Model B Clusters as an Alternative to Advanced Process Chip Computers. In: CCF National Conference of Computer Applications; 2024 Jul; Singapore. Singapore: Springer Nature; 2024. p. 190–199.
59. Ashfaq S, AskariHemmat M, Sah S, Saboori E, Mastropietro O, Hoffman A. Accelerating deep learning model inference on arm cpus with ultra-low bit quantization and runtime. arXiv preprint arXiv:2207.08820. 2022.
60. Adere K, Hailu S, Abebe M, Kemal A, Silassie SW, Tseghai A, Haile G, Tamirat B, Bitew W, Regassa D, Gizachew Z, Alemu W.A systematic review of artificial intelligence applications in Internet of Things, blockchain, and quantum cryptography. Discov Artificial Intelligence. 2026 https://doi.org/10.1007/s44163-026-01029-1

Appendix A: Descriptive statistics of the models

Table A1: Descriptive statistics for both hold-out and 5-Fold CV

Model	Validation	Mean accuracy	Accuracy range	Mean specificity	Specificity range
LSTM	Hold-out	99.52%	0%	100%	0%
	5-Fold	99.60%	0.08%	99.71%	0.29%
RNN	Hold-out	99.51%	0%	99.99%	0%
	5-Fold	99.60%	0.09%	99.86%	0.14%
MLP	Hold-out	99.34%	0%	99.99%	0%
	5-Fold	89.70%	9.64%	90.70%	9.29%

Appendix B: Precision–Recall (PR) and Error Analysis

Table B1: Detailed performance metrics for Hold-out and 5-Fold CV

Model/Metrics	Validation	FPR	FNR	Recall (1-FNR)	Precision
LSTM	Hold-out	0.0000	0.0094	0.9906	~0.9999
	5-Fold	0.0029	0.0055	0.9945	~0.997
RNN	Hold-out	0.0096	0.9904	~0.9999	99.51%
	5-Fold	0.0014	0.0073	0.9927	~0.9986
MLP	Hold-out	0.000075	0.0128	0.9872	~0.9999
	5-Fold	0.0093	0.17	0.83	~0.989