IoT

MDPI

Article

Advancing XSS Detection in IoT over 5G: A Cutting-Edge Artificial Neural Network Approach

Rabee Alqura'n [1], Mahmoud AlJamal [1], Issa Al-Aiash [1], Ayoub Alsarhan [1], Bashar Khassawneh [2], Mohammad Aljaidi [3,*] and Rakan Alanazi [4]

[1] Department of Information Technology, Faculty of Prince Al-Hussien bin Abdullah, Hashemite University, Zarqa 13133, Jordan; alquran.rabee@gmail.com (R.A.); mahmood.yj.98@gmail.com (M.A.); issaalayyash76@gmail.com (I.A.-A.); ayoubm@hu.edu.jo (A.A.)
[2] Department of Computer Science, Faculty of Computer Science and Informatics, Amman Arab University, Amman 11953, Jordan; b.khassawneh@aau.edu.jo
[3] Department of Computer Science, Faculty of Information Technology, Zarqa University, Zarqa 13110, Jordan
[4] Department of Information Technology, Faculty of Computing and Information Technology, Northern Border University, Rafha 91911, Saudi Arabia; rakan.nalenezi@nbu.edu.sa
* Correspondence: mjaidi@zu.edu.jo

Abstract: The rapid expansion of the Internet of Things (IoT) and the advancement of 5G technology require strong cybersecurity measures within IoT frameworks. Traditional security methods are insufficient due to the wide variety and large number of IoT devices and their limited computational capabilities. With 5G enabling faster data transmission, security risks have increased, making effective protective measures essential. Cross-Site Scripting (XSS) attacks present a significant threat to IoT security. In response, we have developed a new approach using Artificial Neural Networks (ANNs) to identify and prevent XSS breaches in IoT systems over 5G networks. We significantly improved our model's predictive performance by using filter and wrapper feature selection methods. We validated our approach using two datasets, NF-ToN-IoT-v2 and Edge-IIoTset, ensuring its strength and adaptability across different IoT environments. For the NF-ToN-IoT-v2 dataset with filter feature selection, our Bilayered Neural Network (2 × 10) achieved the highest accuracy of 99.84%. For the Edge-IIoTset dataset with filtered feature selection, the Trilayered Neural Network (3 × 10) achieved the best accuracy of 99.79%. We used ANOVA tests to address the sensitivity of neural network performance to initial conditions, confirming statistically significant improvements in detection accuracy. The ANOVA results validated the enhancements across different feature selection methods, demonstrating the consistency and reliability of our approach. Our method demonstrates outstanding accuracy and robustness, highlighting its potential as a reliable solution for enhancing IoT security in the era of 5G networks.

Keywords: Internet of Things (IoT); 5G networks; XSS attacks; Artificial Neural Networks (ANNs); cybersecurity

Citation: Alqura'n, R.; AlJamal, M.; Al-Aiash, I.; Alsarhan, A.; Khassawneh, B.; Aljaidi, M.; Alanazi, R. Advancing XSS Detection in IoT over 5G: A Cutting-Edge Artificial Neural Network Approach. *IoT* **2024**, *5*, 478–508. https://doi.org/10.3390/iot5030022

Academic Editor: Amiya Nayak

Received: 22 April 2024
Revised: 4 July 2024
Accepted: 19 July 2024
Published: 25 July 2024

1. Introduction

The rapid evolution of the Internet of Things (IoT) and the advent of 5G technology have profoundly transformed various sectors, including e-learning, e-health, and intelligent industrial manufacturing. While these advancements have integrated smart devices into our daily lives, they have also introduced numerous security challenges. The interconnectivity of devices in IoT ecosystems creates vulnerabilities that cyberattackers can exploit, threatening the integrity and security of these systems [1]. This vulnerability is particularly dangerous in IoT devices operating over 5G networks, which enhance connectivity and capacity while expanding the spectrum of frequencies available for mobile networks [2].

As 5G networks continue to improve, advancements in wireless communication technology are made. However, with new technology, new security issues arise. Developing

robust security guidelines and approaches is critical, mainly as more individuals and systems rely on 5G networks; this entails keeping data safe, safeguarding essential services and systems, and protecting user privacy. By addressing security problems head-on, we can collaboratively develop a secure and robust 5G network. The increased connectivity and bandwidth of 5G networks facilitate more connecting mobile devices simultaneously [3].

The faster speeds and shorter delays of 5G technology raise additional concerns about user privacy and data security. The increased amount of data generated and shared creates risks of unauthorized access to private or sensitive information [4]. Network elements such as base stations and edge servers are potential failure points that criminals might exploit to gain unauthorized access to user data. The interconnected nature of devices and systems in 5G networks heightens these risks [5]. Furthermore, the vast amount of data disseminated by IoT devices using 5G poses concerns about data security, including their resale, user profiling, and intelligence collection [6,7].

One of the security breaches that may threaten IoT devices is Cross-Site Scripting (XSS) attacks, which can be particularly harmful to web-based networks connected to IoT devices [8]. These attacks allow a hacker to exploit XSS vulnerabilities in web applications, altering IoT devices' behavior or gaining access to the system to acquire sensitive information. Severe consequences of XSS attacks on IoT include unauthorized use of devices, data breaches, privacy breaches, and potential physical harm or safety threats [9]. The high-speed distribution of data over 5G networks amplifies the threat posed by XSS attacks, making it critical to develop a comprehensive and proactive security approach [10].

XSS is a type of security vulnerability commonly found in web applications that allow attackers to inject malicious scripts into web pages viewed by other users [11]. These scripts can execute in the context of the user's browser, potentially leading to unauthorized actions, data theft, session hijacking, and more [12]. XSS attacks come in three main types: Stored XSS, Reflected XSS, and DOM-based XSS. Stored XSS involves permanently storing malicious scripts in a target server, such as in a database, comment field, or forum post, served to users' browsers upon request [13]. Reflected XSS occurs when malicious scripts are reflected off a web server in immediate responses, such as error messages or search results, executing as soon as the user receives them [14]. DOM-based XSS happens when client-side scripts in the web application modify the DOM environment in unsafe ways, enabling the execution of an attacker's script [15].

Integrating IoT devices with 5G networks creates new vectors for XSS attacks due to increased connectivity, faster data transmission speeds, and device interdependencies [16]. Attackers can inject malicious scripts into IoT devices that host web interfaces, such as smart home devices with web dashboards. These malicious scripts can be injected into device logs, configurations, or user interfaces. In a 5G network, where IoT devices frequently communicate, a script injected into one device could propagate to other interconnected devices [17]. The high-speed data transmission capabilities of 5G networks mean that the execution of malicious scripts can happen more quickly and on a larger scale, leading to widespread disruption if one device is compromised.

Web-based management interfaces for IoT devices are another common target for XSS attacks. Attackers can inject scripts into fields that administrators or users interact with, like login pages, settings, or logs, causing the malicious script to execute when accessed [18]. Social engineering and phishing are also tactics used by attackers to trick users into clicking on links that contain malicious scripts, exploiting vulnerabilities in the web interfaces of IoT devices [19]. Additionally, as 5G enhances mobile device capabilities, mobile apps that control IoT devices can be targeted by injecting scripts into web views within these apps, compromising the managed IoT devices [20].

Mitigating XSS attacks in IoT over 5G involves several strategies. Input validation and sanitization ensure that all input received by IoT devices and web interfaces is free of malicious scripts [21]. Implementing a Content Security Policy (CSP) restricts the sources from which scripts can be executed, enhancing security through regular security updates for IoT device firmware and software patch known vulnerabilities. Secure coding practices during

development can prevent the introduction of XSS vulnerabilities. Additionally, educating users about the dangers of phishing and social engineering attacks is crucial in reducing the risk of XSS attacks. By employing these measures, the security of IoT devices operating over 5G networks can be significantly improved, mitigating the potential impact of XSS attacks [22].

Contribution

In this research, we achieved an important contribution in the IoT field over 5G networks, so we can summarize them as the following:

1. **Novel ANN Application for XSS Detection:** We introduced a novel ANN approach to detect XSS attacks in IoT systems over 5G, significantly improving detection accuracy and efficiency compared with traditional methods.
2. **Comprehensive Dataset Utilization:** We employed NF-ToN-IoT-v2 and Edge-IIoTset datasets to validate the model's effectiveness and reliability across diverse IoT environments, ensuring generalizability and robustness.
3. **Enhanced Feature Selection:** We utilized both filter (mutual information (MI)) and wrapper (recursive feature elimination (RFE)) feature selection methods to optimize predictive performance, reducing computational complexity while maintaining high accuracy.
4. **Statistical Validation via ANOVA Test:** We applied an ANOVA test to confirm significant improvements in detection accuracy, addressing performance variability due to initial conditions and ensuring robustness and consistency.
5. **High Detection Accuracy:** We achieved remarkable detection accuracies with BLNN and TLNN models, reaching up to 99.84% using BLNN on the NF-ToN-IoT-v2 dataset and 99.79% using TLNN on the Edge-IIoTset dataset, demonstrating the potential for real-time intrusion detection in IoT systems over 5G networks.

The rest of the paper is structured as follows: Section 2 provides related works; Section 3 presents the proposed methodology to detect XSS attacks; Section 4 presents the results and performance evaluation of the ANN detection approach; Section 5 presents the results of the ANOVA test; Section 6 discusses the effectiveness and efficiency of the proposed approach over the related works; and finally, Section 7 concludes the paper and suggests future research directions.

2. Related Work

The widespread use of IoT systems and the deployment of 5G networks have significantly changed how we interact with our environments. However, this new level of connectivity has also opened up many security risks, with XSS attacks posing a significant threat to the security of IoT systems. This section overviews literature reviews on using Machine Learning (ML) and Deep Learning (DL) to secure 5G IoT networks, focusing on studies that address XSS attacks on IoT systems.

Duan et al. [23] addressed the intrusion detection problem in IoT systems, particularly relevant to smart cities. The researchers proposed a novel approach supported by dynamic line graph neural networks and semi-supervised learning. They tested their model on six datasets, including NF-ToN-IoT-V2, and achieved the highest detection accuracy of 95.70% for XSS attacks.

Gaber et al. [24] suggested an injection attack detection system for IoT, proposing an Intrusion Detection System (IDS). They investigated two feature selection approaches, constant removal and recursive feature elimination, with three machine learning classifiers: Support Vector Machine (SVMs), Random Forest, and Decision Tree. Using the AWID dataset (version AWID-CLS-R, created by Constantinos Kolias et al., University of the Aegean, Samos, Greece), the Decision Tree classifier outperformed others with a 99% injection attack detection rate by applying only eight selected features. This research highlights the importance of injection attack detection for the security of smart cities, where numerous threats are anticipated due to their development. Awad et al. [25] emphasized the rapid increase in cyberattacks on IoT networks and devices, highlighting the significance of

ML in Network Intrusion Detection Systems (NIDSs). They noted that the prediction time in anomaly-based NIDSs is directly proportional to the number of factors used by the ML model. Their proposed model achieved a detection accuracy of 98% for XSS attacks using just 13 features, demonstrating the effectiveness of the feature importance model.

Yigit et al. [26] conducted a study on a digital twin-empowered smart attack detection system for 6G Edge of Things (EoT) networks using the ToN-IoT datasets. They employed an online learning algorithm with AutoFS and AutoCM for dynamic and adaptive feature selection and classification. Their system achieved a sensitivity metric of 98.04% for XSS attack detection, proving its efficiency in detecting and preventing these attacks due to innovative feature selection and machine learning techniques. Sarhan et al. [27] explored XSS attack detection within IoT environments, integrating their work with the NF-ToN-IoT-V2 dataset. They utilized a machine-learning-based model to identify XSS injection attacks prevalent in such networks. Their model showcased robustness with an accuracy of 96.83% in detecting XSS attacks.

Awad et al. [28] conducted a study focused on enhancing IIoT security using ML and DL techniques for intrusion detection. The primary objective was to detect and mitigate 14 distinct types of cyberattacks targeting IIoT and IoT protocols. Their methodology involved using the Edge-IIoTset dataset. They implemented various ML algorithms, including k-nearest neighbors (K-NNs), Decision Trees (DTs), and neural networks (NNs). The experiments were conducted using the KNIME platform, with preprocessing steps that included data cleaning, missing value, and normalization to improve classification performance. Their results revealed that the K-NN algorithm achieved an accuracy of 54.37%, while the DT algorithm achieved 85.48% accuracy in detecting XSS attacks. Their study is relevant as it focuses on the effectiveness of ML and DL in securing IoT environments, aligning with our goal of using ANN for XSS attack detection over 5G networks. However, its lower accuracy for specific attacks like XSS indicates a gap in comprehensive threat detection.

Ahmed and Askar [29] developed EdgeGuard, a framework utilizing machine learning for proactive intrusion detection on edge networks. The main aim was to identify and counteract various cyber threats targeting edge and IoT environments. Their approach used convolutional neural networks (CNNs) with residual connections to effectively identify complex patterns in network traffic data. The experiments, conducted using the Edge-IIoTset dataset, demonstrated that their method achieved 77% accuracy in detecting XSS attacks. This research is significant as it showcases the effectiveness of ML techniques in enhancing the security of IoT and edge environments, aligning with our objective of using ANN for detecting XSS attacks over 5G networks.

Ferrag et al. [30] introduced SecurityBERT, a model designed to be both lightweight and privacy-preserving, utilizing the BERT architecture to detect cyber threats in IoT devices. The research focused on enhancing threat detection accuracy while keeping computational demands low, thus making the model ideal for use in environments with limited resources. The methodology incorporated Privacy-Preserving Fixed-Length Encoding (PPFLE) and the Byte-Level Byte-Pair Encoder (BBPE) Tokenizer to effectively process network traffic data. Testing on the Edge-IIoTset dataset showed that SecurityBERT achieved an overall accuracy of 98.2% in detecting fourteen types of attacks, and it achieved 76.22% accuracy specifically for XSS attack detection.

The literature review highlights the significance of Artificial Intelligence (AI) approaches in securing 5G IoT networks from XSS attacks. Most studies noted the increased vulnerability caused by the adoption of IoT systems and the societal transformation facilitated by 5G networks. They emphasized the need for feature selection approaches to enhance the detection rate of intrusion detection systems, affirming the necessity of improving security in AI optimization. Thus, these efforts are justified to enhance the security and preservation of IoT in smart cities despite increasing insecurity.

3. Proposed Methodology

The rapid development of 5G technology and IoT has significantly transformed various aspects of modern life. These advancements have provided mobile networks with wider bandwidths, faster connections, and improved performance. However, they have also introduced a new range of security threats. Among these, XSS attacks are particularly impactful on data confidentiality, exploiting vulnerabilities in network components and web applications and jeopardizing user privacy and data security. The primary objective of this research is to develop a robust deep learning method for detecting XSS attacks on 5G-enabled IoT devices. This approach is crucial for preventing security breaches and ensuring the overall security of IoT systems. Figure 1 illustrates the proposed methodology for identifying XSS attacks using the NF-ToN-IoT-v2 and Edge-IIoTset datasets.

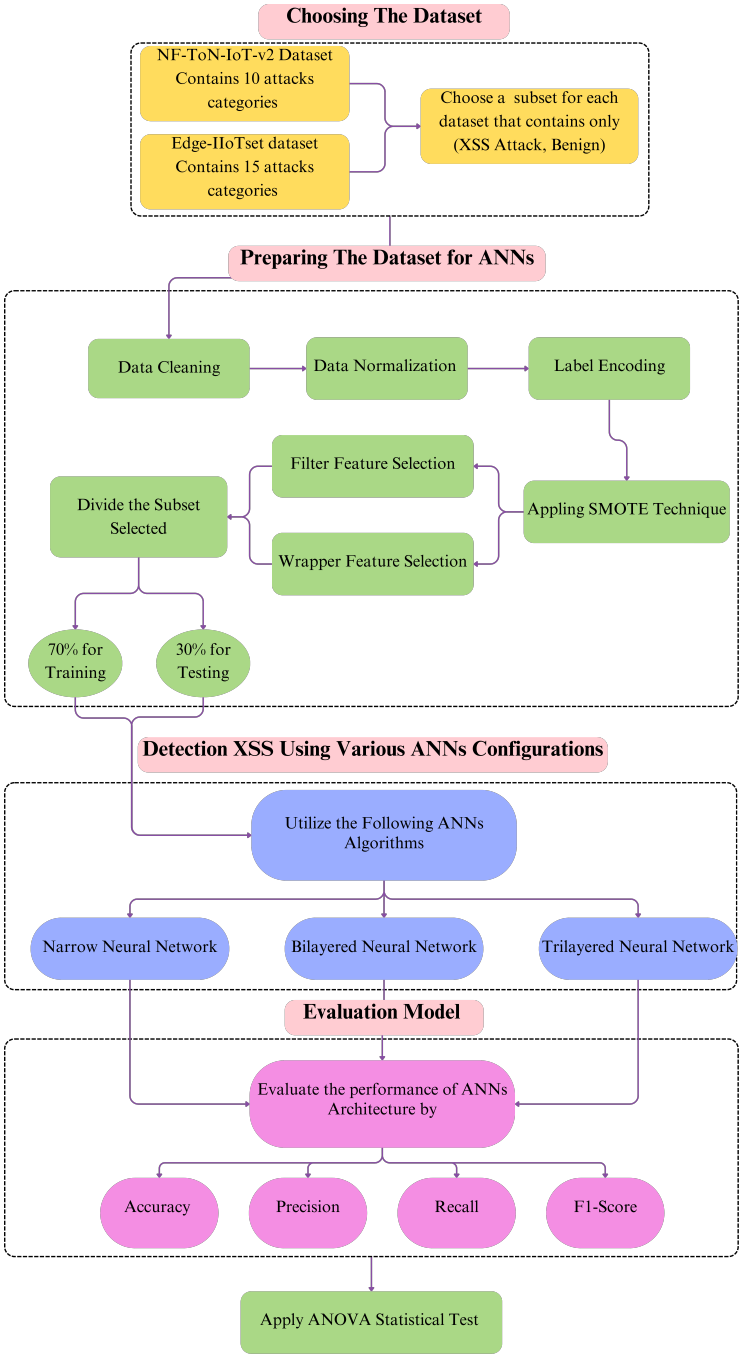


Figure 1. Proposed Approach to Detection XSS Attack.

The methodology begins with selecting subsets from the NF-ToN-IoT-v2 and Edge-IIoTset datasets that focus on "XSS" and "benign" categories. The preprocessing steps include data cleaning, normalization, and label encoding. The Synthetic Minority Over-sampling Technique (SMOTE) is applied to address the issue of imbalanced data. Next, both filter and wrapper feature selection methods are used to identify the most valuable features for XSS detection by ANN architectures. The dataset is divided into subsets: 70% for training and 30% for testing. Various ANN classifiers, including Narrow Neural Network, Bilayered Neural Network, and Trilayered Neural Network, are then employed to detect XSS attacks. The performance of these models is evaluated using metrics such as accuracy, precision, recall, and F1-score. Additionally, the ANOVA statistical test is applied to validate the results. This approach demonstrates its effectiveness in enhancing IoT system security and mitigating potential risks by achieving high accuracy in identifying XSS attacks.

3.1. Dataset

In this work, we adopted two datasets to evaluate the performance of our model as follows:

- **NF-ToN-IoT-v2 Dataset**: All the NetFlow records in the ToN-IoT dataset are generated from publicly available packet captures (pcaps), resulting in the development of NFSIoT, a NetFlow-based IoT network dataset. The NF-ToN-IoT-v2 dataset comprises a total of 16,940,496 data flows. Of these, 6,099,469 (36.01%) are benign samples, and 10,841,027 (63.99%) are attack samples [27]. The sampling distribution is well balanced, as shown in Table 1.

Table 1. Overview of the NF-ToN-IoT-v2 dataset categories, detailing the diversity and scale of cybersecurity threats and benign instances recorded.

Type	Number of Instances	Definition
Benign	6,099,469	Standard secure data packets.
Backdoor	16,809	A technique for breaking into computers that can be accessed from a distance by using specific built-in client apps.
DoS	712,609	An attempt to bring down a system by overloading it.
DDoS	2,026,234	Same as denial-of-service, but with several attack vectors.
Injection	684,465	Attacks that try to alter the way code runs by giving it insecure inputs; "code injections" and "SQL injections" are two of the most popular types.
MITM	7723	In the man in the middle attack, a malicious user is inserted in between the victim and the host with whom they are corresponding.
Password	1,153,323	Uses brute force attacks or a sniffer to get the passwords of users.
Ransomware	3425	The victim's data are encrypted by malicious people, who only unlock them after demanding a ransom.
Scanning	3,781,419	This group, which comprises several techniques for obtaining information about hosts and networks, is also referred to as probing.
XSS	2,455,020	The victim's machines are infected with malicious scripts that change or steal data.

Table 2 provides a comprehensive overview of the features included in the NF-ToN-IoT-v2 dataset, which is utilized for detecting XSS attacks in IoT environments over 5G networks. Each feature is described with its corresponding data type and classification as categorical or numerical. Key features include network-related attributes such as source and destination IP addresses, port numbers, protocol identifiers, and traffic metrics like byte and packet counts. Additional features capture specific behaviors

and network traffic characteristics, such as TCP flags, flow durations, TTL values, packet lengths, and retransmission statistics. The dataset also includes metadata about DNS queries and FTP command responses, which are critical for identifying anomalies indicative of security threats. Combining these diverse features enables a thorough analysis and accurate classification of network traffic, facilitating the detection of potential XSS vulnerabilities and enhancing the overall security of IoT systems in high-speed 5G networks.

Table 2. NF-ToN-IoT-v2 dataset features and descriptions.

Feature Name	Type	Description
IPV4_SRC_ADDR	Categorical	Source IP address in IPv4 format
L4_SRC_PORT	Numerical	Source port number at the transport layer
IPV4_DST_ADDR	Categorical	Destination IP address in IPv4 format
L4_DST_PORT	Numerical	Destination port number at the transport layer
PROTOCOL	Categorical	Protocol used for the communication (e.g., TCP, UDP)
L7_PROTO	Categorical	Application layer protocol identifier
IN_BYTES	Numerical	Number of bytes received by the source
IN_PKTS	Numerical	Number of packets received by the source
OUT_BYTES	Numerical	Number of bytes sent from the source
OUT_PKTS	Numerical	Number of packets sent from the source
TCP_FLAGS	Numerical	Flags set in the TCP header
CLIENT_TCP_FLAGS	Categorical	TCP flags set by the client
SERVER_TCP_FLAGS	Categorical	TCP flags set by the server
FLOW_DURATION_MILLISECONDS	Numerical	Duration of the flow in milliseconds
DURATION_IN	Numerical	Duration of incoming traffic
DURATION_OUT	Numerical	Duration of outgoing traffic
MIN_TTL	Numerical	Minimum time-to-live value observed
MAX_TTL	Numerical	Maximum time-to-live value observed
LONGEST_FLOW_PKT	Numerical	Size of the largest packet in the flow
SHORTEST_FLOW_PKT	Numerical	Size of the smallest packet in the flow
MIN_IP_PKT_LEN	Numerical	Minimum length of IP packets
MAX_IP_PKT_LEN	Numerical	Maximum length of IP packets
SRC_TO_DST_SECOND_BYTES	Numerical	Bytes sent from the source to the destination per second
DST_TO_SRC_SECOND_BYTES	Numerical	Bytes sent from the destination to the source per second
RETRANSMITTED_IN_BYTES	Numerical	Number of bytes retransmitted from the source
RETRANSMITTED_IN_PKTS	Numerical	Number of packets retransmitted from the source
RETRANSMITTED_OUT_BYTES	Numerical	Number of bytes retransmitted to the source
RETRANSMITTED_OUT_PKTS	Numerical	Number of packets retransmitted to the source
SRC_TO_DST_AVG_THROUGHPUT	Numerical	Average throughput from the source to the destination
DST_TO_SRC_AVG_THROUGHPUT	Numerical	Average throughput from the destination to the source
NUM_PKTS_UP_TO_128_BYTES	Numerical	Number of packets up to 128 bytes
NUM_PKTS_128_TO_256_BYTES	Numerical	Number of packets between 128 and 256 bytes
NUM_PKTS_256_TO_512_BYTES	Numerical	Number of packets between 256 and 512 bytes
NUM_PKTS_512_TO_1024_BYTES	Numerical	Number of packets between 512 and 1024 bytes
NUM_PKTS_1024_TO_1514_BYTES	Numerical	Number of packets between 1024 and 1514 bytes
TCP_WIN_MAX_IN	Numerical	Maximum TCP window size for incoming traffic
TCP_WIN_MAX_OUT	Numerical	Maximum TCP window size for outgoing traffic
ICMP_TYPE	Categorical	ICMP message type
ICMP_IPV4_TYPE	Categorical	ICMP type for IPv4
DNS_QUERY_ID	Categorical	DNS query identifier
DNS_QUERY_TYPE	Categorical	Type of DNS query
DNS_TTL_ANSWER	Numerical	Time-to-live for DNS answers
FTP_COMMAND_RET_CODE	Categorical	Return code for FTP commands
Attack	Categorical	Classification label (e.g., Benign, Attack)

- **Edge-IIoTset Dataset**: The Edge-IIoTset dataset is a comprehensive cybersecurity dataset for IoT and Industrial Internet of Things (IIoT) environments, designed to support developing and evaluating intrusion detection systems. This dataset contains network traffic data collected from various IoT devices under normal and attack conditions. The dataset comprises 157,800 records featuring diverse types of cyberattacks and benign instances [31], as shown in Table 3.

Table 3. Overview of the Edge-IIoTset dataset categories, detailing the diversity and scale of cybersecurity threats and benign instances recorded.

Type	Number of Instances	Definition
Backdoor	10,195	Unauthorized access to a system through a backdoor vulnerability.
DDoS_HTTP	10,561	Distributed denial-of-service attacks over HTTP.
DDoS_ICMP	14,090	Distributed denial-of-service attacks using ICMP packets.
DDoS_TCP	10,247	Distributed denial-of-service attacks using TCP packets.
DDoS_UDP	14,498	Distributed denial-of-service attacks using UDP packets.
Fingerprinting	1001	Techniques used to gather information about systems.
MITM	1214	The man in the middle attack, where a malicious user is inserted between the victim and the host with whom they are corresponding.
Normal	24,301	Standard secure data packets.
Password	9989	Uses brute force attacks or a sniffer to obtain the passwords of users.
Port_Scanning	10,071	Techniques for obtaining information about hosts and networks, also referred to as probing.
Ransomware	10,925	The victim's data are encrypted by malicious people, who only unlock them after demanding a ransom.
SQL_injection	10,311	Attacks that try to alter the way a code runs by giving it insecure inputs, specifically targeting SQL databases.
Uploading	10,269	Unauthorized uploading of data to a server.
Vulnerability_scanner	10,076	Tools and techniques used to scan for security vulnerabilities.
XSS	10,052	The victim's machines are infected with malicious scripts that change or steal data.

Table 4 provides a comprehensive overview of the features included in the Edge-IIoTset dataset, which is utilized for detecting various cyberattacks in IoT and Industrial Internet of Things IIoT environments. Each feature is described with its corresponding data type, either categorical or numerical. Key features include network-related attributes such as source and destination IP addresses, port numbers, protocol identifiers, and traffic metrics like byte and packet counts. Additional features, such as TCP flags, flow durations, and retransmission statistics, capture specific behaviors and network traffic characteristics. The dataset also includes metadata about HTTP requests, DNS queries, and other protocol-specific details critical for identifying anomalies indicative of security threats. Combining these diverse features enables a thorough analysis and accurate classification of network traffic, facilitating the detection of potential cyber vulnerabilities and enhancing the overall security of IoT systems.

Table 4. Edge-IIoTset dataset features and descriptions.

Feature Name	Type	Description
frame.time	Categorical	Timestamp of the frame capture
ip.src_host	Categorical	Source IP address
ip.dst_host	Categorical	Destination IP address
arp.dst.proto_ipv4	Categorical	Destination protocol address in ARP packets
arp.opcode	Numerical	ARP operation code
arp.hw.size	Numerical	ARP hardware size
arp.src.proto_ipv4	Categorical	Source protocol address in ARP packets
icmp.checksum	Numerical	Checksum value for ICMP packets
icmp.seq_le	Numerical	Sequence number in ICMP packets
icmp.transmit_timestamp	Numerical	Transmit timestamp in ICMP packets
http.file_data	Categorical	Data field in HTTP packets
http.content_length	Numerical	Content length in HTTP packets
http.request.uri.query	Categorical	Query string in HTTP request URI
http.request.method	Categorical	HTTP request method
http.referer	Categorical	HTTP referer header
http.request.full_uri	Categorical	Full URI in HTTP request
http.request.version	Categorical	HTTP request version
http.response	Numerical	HTTP response status code
http.tls_port	Numerical	Port used for TLS connections
tcp.ack	Numerical	TCP acknowledgment number
tcp.ack_raw	Numerical	Raw TCP acknowledgment number
tcp.checksum	Numerical	Checksum value in TCP packets
tcp.connection.fin	Numerical	FIN flag in TCP connections
tcp.connection.rst	Numerical	RST flag in TCP connections
tcp.connection.syn	Numerical	SYN flag in TCP connections
tcp.connection.synack	Numerical	SYN-ACK flag in TCP connections
tcp.dstport	Numerical	Destination port number in TCP connections
tcp.flags	Numerical	Flags set in TCP packets
tcp.len	Numerical	Length of TCP segment
tcp.srcport	Numerical	Source port number in TCP connections
tcp.stream	Numerical	TCP stream index
tcp.window_size_value	Numerical	TCP window size value
udp.length	Numerical	Length of UDP segment
udp.port	Numerical	Port number in UDP packets
udp.stream	Numerical	UDP stream index
mqtt.hdrflags	Numerical	MQTT header flags
mqtt.len	Numerical	Length of MQTT segment
mqtt.msg	Numerical	MQTT message type
mqtt.proto_len	Numerical	MQTT protocol length
mqtt.protoname	Categorical	MQTT protocol name
mqtt.topic	Categorical	MQTT topic name
mqtt.topic_len	Numerical	Length of MQTT topic
mqtt.ver	Numerical	MQTT protocol version
mbtcp.len	Numerical	Length of Modbus TCP segment
mbtcp.trans_id	Numerical	Modbus TCP transaction identifier
mbtcp.unit_id	Numerical	Modbus TCP unit identifier
Attack_type	Categorical	Type of attack (Benign, MITM, DoS, DDoS, XSS, Scanning, Ransomware, SQL)

3.2. Data Preprocessing

Preprocessing is essential in transforming unprocessed input into usable data, involving extensive cleaning to remove errors and superfluous information. Our method modifies the raw NF-ToN-IoT-v2 dataset, which includes 44 attributes and 53,464 entries, by correcting anomalies with substitute values and addressing missing entries using the maximum values of attributes. To facilitate the training of ANNs, we convert categorical labels into numeric codes, classifying network traffic as "Benign (0)" or "Attack (1)", and use the SMOTE technique to balance the dataset. We also apply feature selection techniques

to decrease the size of the dataset and simplify the computational demands, enhancing the effectiveness and quality of the analysis. These steps in the data preparation process are crucial for ensuring that the ANN model is trained efficiently and effectively utilizing MATLAB software (version R2023a, MathWorks, Natick, MA, USA).

- **Data Cleaning:** The first step involved identifying and eliminating all cases with missing (NaN) or infinite (Inf) values. They may be absent from the dataset due to measurement errors or data corruption. Once such cases were cleaned, the dataset became more uniform and accurate for further analysis and model training [32].
- **Data Normalization:** The data have been normalized to improve the performance of the training and the ANN model. In addition, normalizing is critical when using a dataset in which each feature has significant numerical disparities. Normalizing all features to the range between 0 and 1 helps ensure the classification's accuracy, but it also aids in minimizing the training time and potential error since the less exponentially prominent features would not govern the learning process [33].
- **Label Encoding:** Label encoding is a fundamental preprocessing step in the field of ANN, particularly useful when working with categorical data. This technique transforms categorical variables into a numerical format, making them compatible with ANN algorithms that require numerical input. In this study, we employed label encoding to convert categorical features into numeric labels, facilitating the training and evaluation of our predictive model [34]. The adopted label encoding process can be summarized using Algorithm 1.

Algorithm 1: Label encoding of categorical features.

Input: Dataset D with categorical features C.
Output: Dataset D' with categorical features transformed into numeric labels.
Initialize an instance of LabelEncoder: label_encoder ← new LabelEncoder(). // Step 1: Initialize Label Encoder
Define a list of categorical columns: `categorical_columns` ← list of categorical features in D. // Step 2: Define Categorical Columns
for each column c in `categorical_columns`
    Fit the LabelEncoder to the column c and transform the values:
    transformed_values ← label_encoder.fit_transform($D[c]$). // Step 3: Fit and Transform
    Replace the original column values in D with the transformed numeric labels:
    $D[c]$ ← transformed_values. // Replace Values

3.3. Filter Feature Selection Method

We utilized mutual information (MI) to evaluate and select features. This method operates independently of any ANN model, focusing instead on the intrinsic properties of the data. The fundamental concept behind filter methods is to score the relevance of features based on statistical measures, precisely the MI in this case, which quantifies the amount of information one variable provides about another. The MI between a feature and the target variable is a non-negative value that measures their dependency [35]. It is calculated as the following Equation (1):

$$MI(X;Y) = \sum_{x \in X, y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \quad (1)$$

where X and Y are the feature and target variable, respectively; $p(x,y)$ is the joint probability distribution function of X and Y; and $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y. A higher MI score indicates a greater relevance of the

feature to the target variable, suggesting that the feature shares more information with the target.

The process began with the encoded dataset S, during which the features X and the target variable y were isolated. Following this, for each feature f in X, we computed the mutual information score between f and y using the `mutual_info_classif` function. We then selected the top ten features with the highest mutual information (MI) scores to ensure an optimal balance between retaining highly informative features and minimizing model complexity. This approach was strategically chosen based on empirical evidence, suggesting that selecting features with the highest MI scores significantly enhances the model's predictive accuracy while reducing the risk of overfitting. Algorithm 2 illustrates the details of this method. Figure 2 presents the MI scores for all features in the NF-ToN-IoT-V2 dataset, and Figure 3 presents the selected features with the highest MI values. Figure 4 presents the MI scores for all features in the Edge-IIoTset dataset, and Figure 5 presents the selected features with the highest MI values.

Algorithm 2: Filter feature selection based on mutual information (MI).

Input: Dataset S with features X and target class labels y, and optionally a threshold t for feature selection.

Output: Ranked list of features S^* based on their mutual information scores.

$S^* \leftarrow \emptyset$ // Initialize the ranked list of features

for each feature f in X

  Compute mutual information score between feature f and y using `mutual_info_classif` and store the score. // Score calculation for f

Rank X based on computed mutual information scores to form S^*. // Feature ranking

if threshold t is specified

  $S^* \leftarrow$ features from S^* with scores $\geq t$. // Threshold filtering

else

  Select the top k features from S^* for final inclusion. // Top k selection

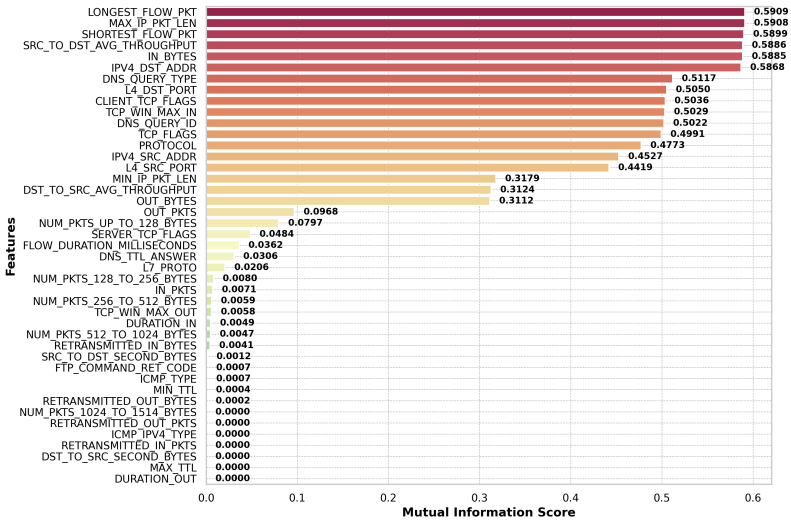


Figure 2. MI scores for all features in the NF-ToN-IoT-V2 dataset.

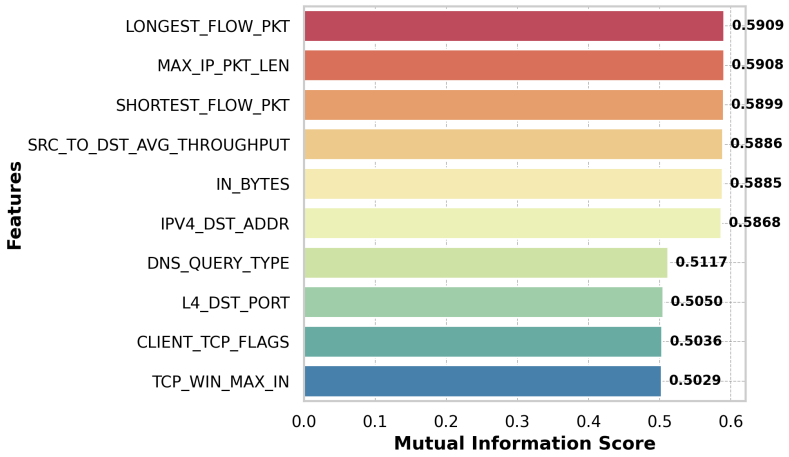


Figure 3. Selected features with the highest MI values in the NF-ToN-IoT-V2 dataset.

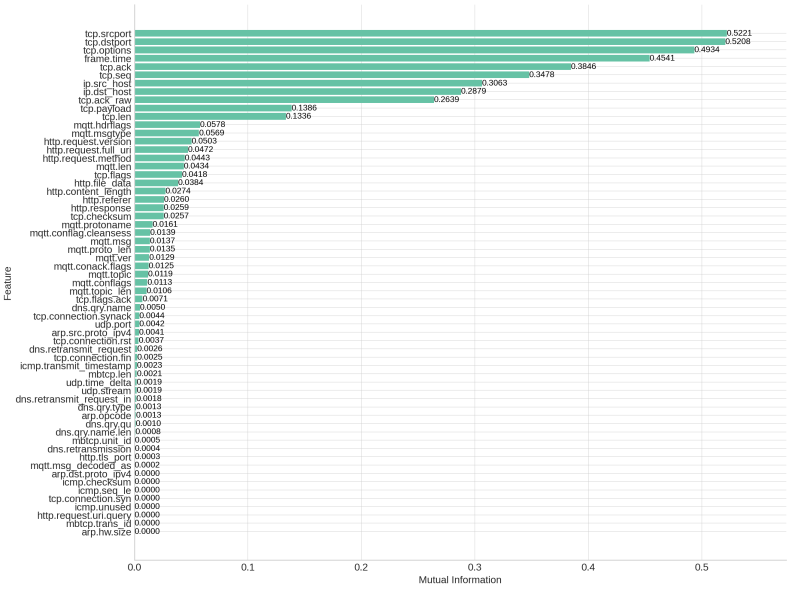


Figure 4. MI scores for all features in the Edge-IIoTset dataset.

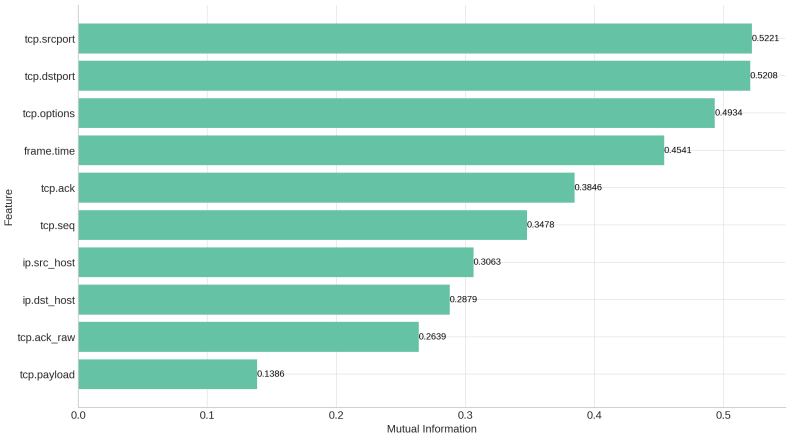


Figure 5. Selected features with the highest MI values in the Edge-IIoTset dataset.

3.4. Wrapper Feature Selection Method

We employed the recursive feature elimination (RFE) method for feature selection. Unlike filter methods, RFE operates with a predictive model, iteratively refining the feature subset to enhance model performance [36]. The core principle of wrapper methods is to

evaluate feature subsets based on the performance of a chosen model, optimizing for the most predictive combination of features. RFE begins by training an estimator on the entire set of features and computing the importance of each feature. The least important features are then recursively pruned from the current set of features. Specifically, RFE ranks the features based on their importance and recursively removes the least important feature, refitting the model on the remaining features in each iteration until the desired number of features is reached [37].

In this study, we utilized a Linear Regression model as the estimator within the RFE algorithm, leveraging its robustness and capability to handle complex datasets. The process started with the encoded dataset S, isolating the features X and the target variable y. We then applied the RFE method with the Random Forest classifier to iteratively rank and select the top ten most important features. Equation (2) clarifies how it works as follows:

$$RFE(X; y) = \arg \min_{|X'|=k} \sum_{i=1}^{n} L(y_i, f(X_i')) \tag{2}$$

where X' represents the selected subset of features, k is the number of desired features, L is the loss function, and f is the predictive model. By minimizing the loss function, RFE identifies the feature subset that contributes most significantly to the model's predictive accuracy.

This approach ensures that the selected features not only retain high predictive power but also maintain the interpretability and relevance of the model. Algorithm 3 details the RFE process. Figures 6 and 7 illustrate the feature importances and the selected features in the NF-ToN-IoT-V2 dataset, respectively, while Figures 8 and 9 present the corresponding results for the Edge-IIoTset dataset.

Algorithm 3: Wrapper feature selection based on recursive feature elimination (RFE).

Input: Dataset S with features X and target class labels y, the number of features to select k, and optionally the step size s for feature elimination.
Output: Ranked list of features S^* based on their importance.

$S^* \leftarrow X$ // Initialize the set of features
while $|S^*| > k$ **do**
  Train the model f using features in S^*. // Model training
  Compute feature importances $\beta = [\beta_1, \beta_2, \ldots, \beta_m]$ using the trained model. // Importance computation
  Rank features based on the importance scores β. // Feature ranking
  Identify the least important feature f_j with the lowest β_j. // Identify least important feature
  if s is specified
    Remove the s least important features from S^*. // Remove s features
  else
    Remove the single least important feature f_j from S^*. // Remove one feature
  Evaluate model performance $L(y_i, f(X_i'))$ with the remaining features. // Performance evaluation
  Update feature importances β based on model performance. // Importance update
Rank S^* based on the final computed importances β. // Final feature ranking
Select the top k features from S^* for final inclusion. // Top k selection

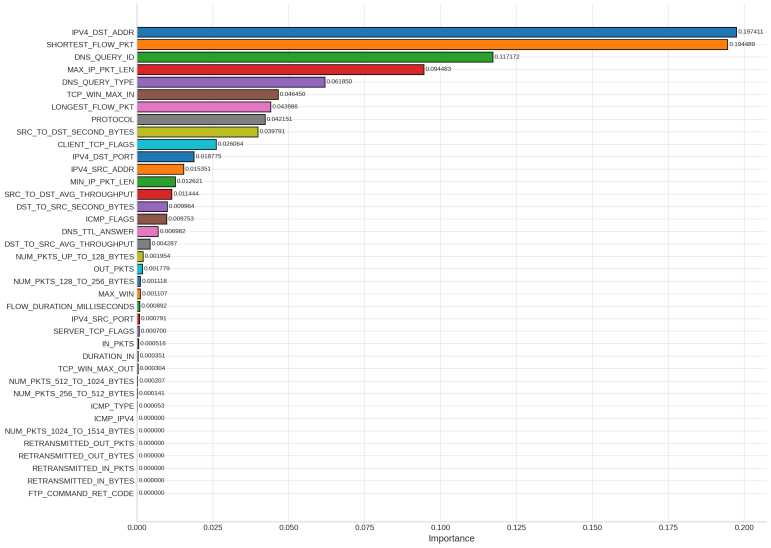


Figure 6. Feature importance scores for all features in the NF-ToN-IoT-V2 dataset.

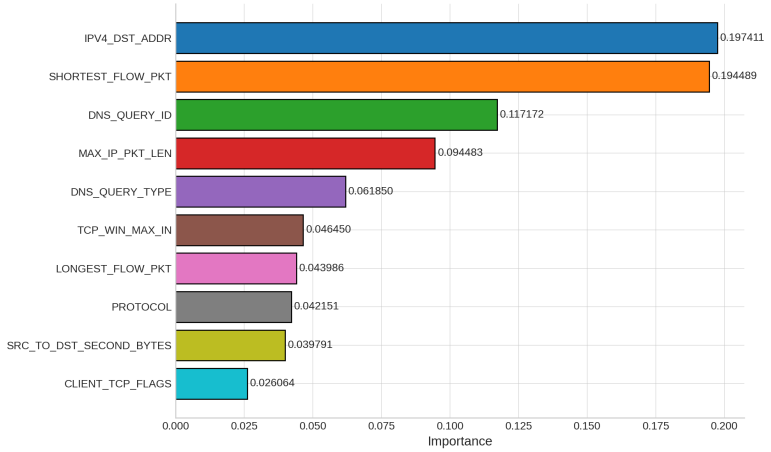


Figure 7. Selected features with the highest feature importance values in the NF-ToN-IoT-V2 dataset.

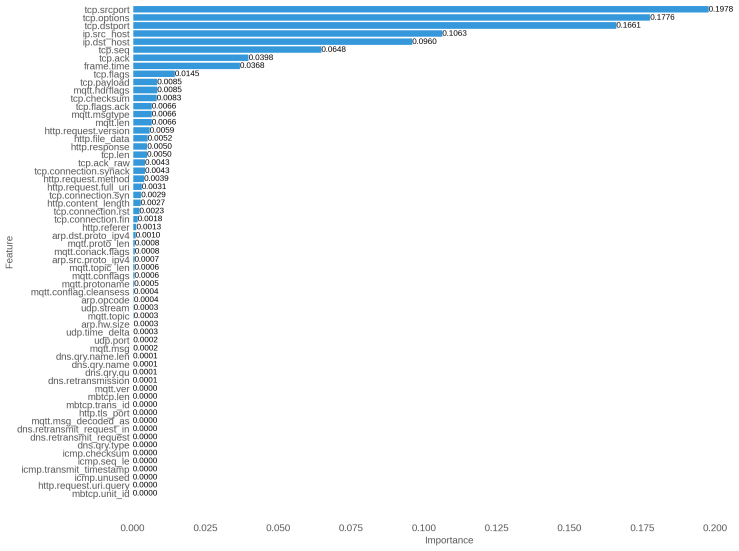


Figure 8. Feature importance scores for all features in the Edge-IIoTset dataset.

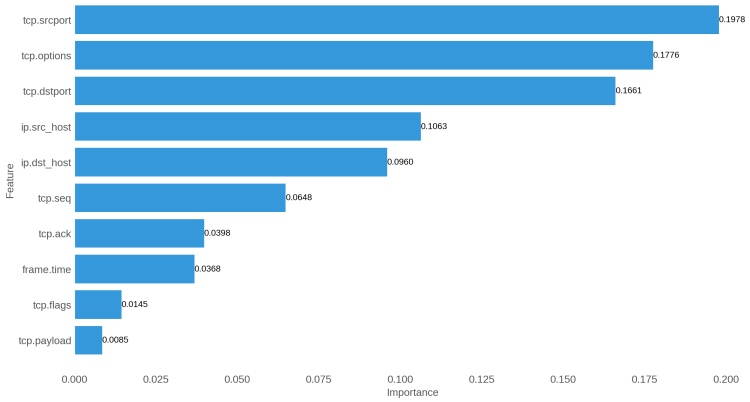


Figure 9. Selected features with the highest feature importance values in the Edge-IIoTset dataset.

3.5. Classification Methods

In addressing the challenge of XSS attack detection, neural network architectures were adopted, ranging from the simplicity of Narrow Neural Networks to the complexity encapsulated within Trilayered Neural Networks. This endeavor is propelled by MATLAB's software, allowing for an in-depth comparative analysis of architectures varying in layers and neuron counts to capture and model our dataset's intricate dynamics accurately.

3.5.1. Narrow Neural Network

The Narrow Neural Network, with its single hidden layer, exemplifies model efficiency and swift training capabilities at the foundational level, making it particularly suitable for less complex predictive tasks. This algorithm initializes with random weights and biases, progressing through cycles of forward propagation, where data transformations through linear and non-linear operations culminate in output predictions. Backpropagation follows, adjusting weights and biases to minimize error, a process mathematically expressed as Equation (3); Algorithm 4 illustrates the procedures of this model as follows [38]:

$$f(x) = \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2) \tag{3}$$

where σ denotes the activation function; W_1, W_2 the weight matrices; and b_1, b_2 the bias vectors, illuminating the path from inputs x to the network's output.

Algorithm 4: Training procedure for Narrow Neural Network (NNN).

Input: Input features X, Target labels Y, Learning rate η, Number of epochs
Output: Trained NNN model with optimized weights and biases

$S^* \leftarrow \emptyset$ // Initialize the ranked list of features
Initialize weights W_1, W_2 and biases b_1, b_2 randomly
for each epoch
  foreach sample (x, y) in X, Y **do**
    // Forward propagation
    Compute hidden layer: $H = \sigma(W_1 \cdot x + b_1)$
    Compute output: $\hat{y} = \sigma(W_2 \cdot H + b_2)$
    // Compute loss and gradient
    Calculate loss and its gradient ∇L
    // Backpropagation
    Update W_2, b_2 using gradients
    Compute gradients for W_1, b_1
    Update W_1, b_1 using gradients
return Optimized weights W_1, W_2 and biases b_1, b_2

3.5.2. Bilayered Neural Network

Evolving complexity, the Bilayered Neural Network integrates an additional hidden layer, enabling the model to capture more nuanced patterns. This architecture's ability to abstract complex relationships makes it apt for tasks with evolving data patterns, such as image and speech recognition. The BLNN extends the operational framework of the NNN, incorporating an extra layer into both the forward and backward propagation phases [39], thereby enhancing the model's depth and capability, as is apparent in Equation (4); Algorithm 5 presents the details of this model as follows:

$$f(x) = \sigma(W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2) + b_3) \tag{4}$$

Here, W_3 and b_3 extend the model to accommodate another layer of computation, enriching the network's capacity to process and learn from the input data.

Algorithm 5: Training procedure for Bilayered Neural Network (BLNN).

Input: Input features X, Target labels Y, Learning rate η, Number of epochs
Output: Trained BLNN model with optimized weights and biases
Initialize weights W_1, W_2, W_3 and biases b_1, b_2, b_3 randomly
for each epoch
  foreach sample (x, y) in X, Y **do**
    // Forward propagation
    Compute first hidden layer: $H_1 = \sigma(W_1 \cdot x + b_1)$
    Compute second hidden layer: $H_2 = \sigma(W_2 \cdot H_1 + b_2)$
    Compute output: $\hat{y} = \sigma(W_3 \cdot H_2 + b_3)$
    // Compute loss and gradient
    Calculate loss and its gradient ∇L
    // Backpropagation
    Update W_3, b_3 using gradients
    Compute and update gradients for W_2, b_2
    Compute and update gradients for W_1, b_1
return Optimized weights and biases

3.5.3. Trilayered Neural Network

The Trilayered Neural Network, with its three hidden layers, represents the zenith of complexity in our exploration. This architecture's deep structure is adept at modeling high-level abstractions, making it ideal for tackling the most intricate tasks in ANN, including natural language processing and advanced time series forecasting. The TLNN algorithm meticulously orchestrates forward and backward propagations across three layers, refining the network's parameters for optimal performance [40]. The mathematical representation captures this complexity in Equation (5); Algorithm 6 illustrates the procedures of this model as follows:

$$f(x) = \sigma(W_4 \cdot \sigma(W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2) + b_3) + b_4) \tag{5}$$

where W_4 and b_4 are added to accommodate the third layer, highlighting the intricate computations that enable the TLNN to perform its sophisticated analyses.

Algorithm 6: Training procedure for Trilayered Neural Network (TLNN).

Input: Input features X, Target labels Y, Learning rate η, Number of epochs
Output: Trained TLNN model with optimized weights and biases
Initialize weights W_1, W_2, W_3, W_4 and biases b_1, b_2, b_3, b_4 randomly
for each epoch
  foreach sample (x, y) in X, Y **do**
    // Forward propagation
    Compute first hidden layer: $H_1 = \sigma(W_1 \cdot x + b_1)$
    Compute second hidden layer: $H_2 = \sigma(W_2 \cdot H_1 + b_2)$
    Compute third hidden layer: $H_3 = \sigma(W_3 \cdot H_2 + b_3)$
    Compute output: $\hat{y} = \sigma(W_4 \cdot H_3 + b_4)$
    // Compute loss and gradient
    Calculate loss and its gradient ∇L
    // Backpropagation
    Sequentially update $W_4, b_4; W_3, b_3; W_2, b_2; W_1, b_1$ using calculated gradients
return Optimized weights W_1, W_2, W_3, W_4 and biases b_1, b_2, b_3, b_4

3.6. Evaluation Metrics

The original set was divided into two 80% for training and the remaining for testing. Performance evaluation measures were selected based on the concepts introduced above in the confusion matrix: true positive (TP), false positive (FP), false negative (FN), and true negative (TN). FP corresponds to false alarms, and FN corresponds to misses. The evaluation process uses accuracy, precision, recall, and F1-score measures.

- **Accuracy:** Accuracy measures the overall correctness of model predictions by comparing correct predictions with the total predictions. While useful, it may not be the best indicator for imbalanced datasets, where it can misleadingly appear high if the model predominantly predicts the majority class accurately but fails with the minority class [41].

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6}$$

- **Precision:** Precision is a metric that evaluates the accuracy of a model's positive predictions. It is calculated by dividing the number of true positives by the total predicted positives, which include both true and false positives. This measure is crucial in contexts where avoiding false positives is important. It helps assess how well a model identifies only relevant instances as positive [42].

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

- **Recall:** Sensitivity or True Positive Rate: Recall is calculated as the ratio of actual positive instances that are predicted as positives. In other words, it shows how many positive instances the model correctly identified without missing anyone. It is calculated as the ratio of true positive and the sum of true positive and false negative [43].

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

- **F1-score:** The harmonic mean of precision and recall, F1-score is the balanced measure considering precision and recall together and mainly used to find an optimal balance between precision and recall. It is calculated by the following [44]:

$$F1 - score = \frac{2 \cdot Recall \cdot precision}{Recall + precision} \tag{9}$$

These metrics are crucial for evaluating the performance of a classification model comprehensively. Accuracy provides a general indication of the model's effectiveness. Meanwhile, precision, recall, and the F1-score offer detailed insights into the model's ability to manage false positives and false negatives and the balance between precision and recall. The choice of which metric(s) to use depends on the specific problem being addressed and the desired outcomes of the model evaluation.

3.7. ANOVA Test for Performance Variability Analysis

To analyze the variability in neural network performance due to different initial conditions, we employed the Analysis of Variance (ANOVA) test. ANOVA is a statistical method used to determine whether there are any statistically significant differences between the means of two or more independent groups [45]. In this study, ANOVA was applied to compare neural network performance metrics (accuracy) across multiple trials with varying initial conditions.

The ANOVA test partitions the total variability in the data into components attributable to different sources of variation. In a one-way ANOVA, the primary components are the variability between groups and the variability within groups. The between-groups variability measures the variation due to differences between the groups, whereas the within-groups variability measures the variation within each group.

The total sum of squares (SS_{total}) is the sum of the between-groups sum of squares ($SS_{between}$) and the within-groups sum of squares (SS_{within}). The sum of squares for between groups ($SS_{between}$) is calculated as follows:

$$SS_{between} = \sum_{i=1}^{k} n_i(\bar{X}_i - \bar{X})^2 \tag{10}$$

where k is the number of groups, n_i is the number of observations in group i, $\bar{X}_i$ is the mean of group i, and $\bar{X}$ is the overall mean of all observations.

The sum of squares for within groups (SS_{within}) is calculated as follows:

$$SS_{within} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_i)^2 \tag{11}$$

where X_{ij} is the j-th observation in group i.

The mean squares for between groups ($MS_{between}$) and within groups (MS_{within}) are obtained by dividing the corresponding sum of squares by their degrees of freedom (df). The mean square for between groups ($MS_{between}$) is calculated as follows:

$$MS_{between} = \frac{SS_{between}}{k-1} \tag{12}$$

and the mean square for within groups (MS_{within}) is calculated as follows:

$$MS_{within} = \frac{SS_{within}}{N-k} \tag{13}$$

where N is the total number of observations across all groups, and k is the number of groups.

The F-statistic is then calculated as the ratio of the between-groups mean square to the within-groups mean square, as follows:

$$F = \frac{MS_{between}}{MS_{within}} \tag{14}$$

This F-statistic follows an F-distribution with $k-1$ and $N-k$ degrees of freedom. The p-value associated with the F-statistic is used to determine whether the observed differences between group means are statistically significant.

We applied this ANOVA test to analyze the variability in neural network performance, ensuring that the evaluation accounts for differences due to initial conditions. Algorithm 7 used the ANOVA test to analyze the variability in neural network performance; this approach provides a statistically robust comparison of performance metrics. The results of the ANOVA test are discussed in Section 4.

Algorithm 7: ANOVA test for neural network performance.

Input: Number of trials per group (T), Number of groups (G)
Output: F-statistic, p-value
Initialize arrays to store performance metrics for each group
for each group g from 1 to G
    for each trial t from 1 to T
        Set random seed
        Train neural network
        Record performance metric (accuracy) for group g
Compute overall mean of all observations ($\bar{X}$)
Compute $\text{SS}_{\text{between}} = 0$, $\text{SS}_{\text{within}} = 0$
for each group i from 1 to G
    Compute group mean ($\bar{X}_i$)
    $\text{SS}_{\text{between}} += n_i(\bar{X}_i - \bar{X})^2$
    for each observation j in group i
        $\text{SS}_{\text{within}} += (X_{ij} - \bar{X}_i)^2$
Compute $\text{MS}_{\text{between}} = \frac{\text{SS}_{\text{between}}}{G-1}$
Compute $\text{MS}_{\text{within}} = \frac{\text{SS}_{\text{within}}}{N-G}$
Compute $F = \frac{\text{MS}_{\text{between}}}{\text{MS}_{\text{within}}}$
Determine p-value from F-distribution
return F-statistic and p-value

4. Results

The performance metrics presented in Table 5 provide a comprehensive evaluation of various neural network architectures during the training and testing stages across four datasets: NF-ToN-IoT-V2 Filtered Dataset, NF-ToN-IoT-V2 Wrapper Dataset, Edge-IIoTset Filtered Dataset, and Edge-IIoTset Wrapper Dataset.

For the NF-ToN-IoT-V2 Filtered Dataset, the Bilayered Neural Network architectures (2 × 10 and 2 × 25) exhibit superior performance, achieving remarkably high metrics across both training and testing stages. Specifically, the Bilayered Neural Network 2 × 10 achieves a training accuracy of 99.93% and a testing accuracy of 99.84%, with precision values of 99.90% (train) and 99.78% (test), recall values of 99.92% (train) and 99.82% (test), and F1-scores of 99.91% (train) and 99.80% (test). The Bilayered Neural Network 2 × 25 closely follows with a training accuracy of 99.91% and a testing accuracy of 99.88%, precision values of 99.88% (train) and 99.84% (test), recall values of 99.89% (train) and 99.86% (test), and F1-scores of 99.89% (train) and 99.85% (test). These results highlight their robust capability in generalizing from training data to accurately detect XSS attacks with minimal false positives and false negatives.

In comparison, the Medium Neural Network 1 × 25 demonstrates strong performance, particularly in the testing stage, with an accuracy of 98.99%, a precision value of 98.68%, a recall value of 98.88%, and an F1-score of 98.78%. This surpasses the Wide Neural Network 1 × 100, which records a slightly lower testing accuracy of 98.79%, a precision value of 98.43%, a recall value of 98.66%, and an F1-score of 98.54%. This indicates the effectiveness of the Medium Neural Network in balancing model complexity and generalization ability,

making it a preferable choice over the Wide Neural Network for this dataset. Conversely, the Narrow Neural Network 1 × 10 shows a lower performance, with a training accuracy of 98.77%, a testing accuracy of 98.74%, precision values of 98.40% (train) and 98.36% (test), recall values of 98.64% (train) and 98.60% (test), and F1-scores of 98.52% (train) and 98.48% (test). These results highlight the challenges simpler architectures face in accurately detecting XSS attacks, underlining the importance of a little increasing model complexity for better performance.

Table 5. Performance metrics of different neural network architectures during training and testing stages.

Algorithm	Layers	Accuracy		Precision		Recall		F1-Score	
		Train Stage	Test Stage	Train Stage	Test Stage	Train Stage	Test Stage	Train Stage	Test Stage
NF-ToN-IoT-V2 Filtered Dataset									
Narrow Neural Network	1 × 10	0.9877	0.9874	0.984	0.9836	0.9864	0.986	0.9852	0.9848
Medium Neural Network	1 × 25	0.9894	0.9899	0.9862	0.9868	0.9882	0.9888	0.9872	0.9878
Wide Neural Network	1 × 100	0.9881	0.9879	0.9845	0.9843	0.9868	0.9866	0.9857	0.9854
Bilayered Neural Network	**2 × 10**	**0.9993**	**0.9984**	**0.999**	**0.9978**	**0.9992**	**0.9982**	**0.9991**	**0.998**
Bilayered Neural Network	2 × 25	0.9991	0.9988	0.9988	0.9984	0.9989	0.9986	0.9989	0.9985
Trilayered Neural Network	3 × 10	0.9913	0.991	0.9886	0.9883	0.9903	0.99	0.9895	0.9891
Trilayered Neural Network	3 × 25	0.991	0.9899	0.9883	0.9868	0.99	0.9888	0.9891	0.9878
NF-ToN-IoT-V2 Wrapper Dataset									
Narrow Neural Network	1 × 10	0.9747	0.981	0.9674	0.9754	0.972	0.979	0.9697	0.9772
Medium Neural Network	1 × 25	0.987	0.9853	0.9831	0.9809	0.9856	0.9837	0.9843	0.9823
Wide Neural Network	1 × 100	0.9877	0.9877	0.984	0.984	0.9864	0.9864	0.9852	0.9852
Bilayered Neural Network	2 × 10	0.9946	0.9939	0.9929	0.992	0.994	0.9932	0.9934	0.9926
Bilayered Neural Network	**2 × 25**	**0.9988**	**0.9983**	**0.9984**	**0.9978**	**0.9986**	**0.9981**	**0.9985**	**0.9979**
Trilayered Neural Network	3 × 10	0.9976	0.9976	0.9968	0.9968	0.9973	0.9973	0.9971	0.9971
Trilayered Neural Network	3 × 25	0.9972	0.9971	0.9963	0.9961	0.9968	0.9967	0.9966	0.9964
Edge-IIoTset Filtered Dataset									
Narrow Neural Network	1 × 10	0.9664	0.9659	0.957	0.9564	0.9629	0.9625	0.9599	0.9594
Medium Neural Network	1 × 25	0.9744	0.9745	0.967	0.9672	0.9717	0.9718	0.9693	0.9694
Wide Neural Network	1 × 100	0.9796	0.9779	0.9736	0.9661	0.9774	0.9708	0.9755	0.9684
Bilayered Neural Network	2 × 10	0.9815	0.9813	0.9761	0.9692	0.9795	0.9737	0.9778	0.9714
Bilayered Neural Network	2 × 25	0.9932	0.99	0.9911	0.9884	0.9924	0.9901	0.9917	0.9892
Trilayered Neural Network	**3 × 10**	**0.9985**	**0.9979**	**0.998**	**0.9972**	**0.9983**	**0.9976**	**0.9981**	**0.9974**
Trilayered Neural Network	3 × 25	0.997	0.9964	0.9961	0.9953	0.9966	0.996	0.9963	0.9956
Edge-IIoTset Wrapper Dataset									
Narrow Neural Network	1 × 10	0.9867	0.9859	0.9827	0.9817	0.9852	0.9844	0.984	0.9831
Medium Neural Network	1 × 25	0.99	0.9911	0.987	0.9884	0.9889	0.9901	0.9879	0.9892
Wide Neural Network	1 × 100	0.99	0.9893	0.987	0.9861	0.9889	0.9882	0.9879	0.9871
Bilayered Neural Network	**2 × 10**	**0.9968**	**0.9971**	**0.9958**	**0.9962**	**0.9964**	**0.9967**	**0.9961**	**0.9964**
Bilayered Neural Network	2 × 25	0.9949	0.9953	0.9933	0.9939	0.9943	0.9948	0.9938	0.9943
Trilayered Neural Network	3 × 10	0.9953	0.995	0.9938	0.9934	0.9948	0.9944	0.9943	0.9939
Trilayered Neural Network	3 × 25	0.9934	0.993	0.9914	0.9909	0.9926	0.9922	0.992	0.9915

For the NF-ToN-IoT-V2 Wrapper Dataset, the Trilayered Neural Network configurations (3 × 10 and 3 × 25) show consistently robust performance. The 3 × 10 network achieves a training accuracy of 99.76% and a testing accuracy of 99.76%, precision values of 99.68% (train) and 99.68% (test), recall values of 99.73% (train) and 99.73% (test), and F1-scores of 99.71% (train) and 99.71% (test). The 3 × 25 network follows closely with a training accuracy of 99.72% and a testing accuracy of 99.71%, precision values of 99.63% (train) and 99.61% (test), recall values of 99.68% (train) and 99.67% (test), and F1-scores of 99.66% (train) and 99.64% (test). These configurations outperform the Bilayered Neural Network 2 × 25, which, although also performing exceptionally well, records slightly lower metrics, particularly in precision (99.84% train, 99.78% test) and recall (99.86% train, 99.81% test).

In the case of the Edge-IIoTset dataset, the Trilayered Neural Network 3 × 10 emerges as the top performer with a test stage accuracy of 99.79%, showcasing its outstanding generalization capabilities. Specifically, it records a training accuracy of 99.85%, precision values of 99.80% (train) and 99.72% (test), recall values of 99.83% (train) and 99.76% (test), and F1-scores of 99.81% (train) and 99.74% (test). The Bilayered Neural Network 2 × 25 also achieves high performance, particularly in the filtered dataset variant, recording a training accuracy of 99.32% and a test stage accuracy of 99.00%, precision values of 99.11% (train) and 98.84% (test), recall values of 99.24% (train) and 99.01% (test), and F1-scores of 99.17% (train) and 98.92% (test). These results indicate that while both configurations are highly effective, the Trilayered Neural Network 3 × 10 holds a slight edge in terms of overall performance.

Across all datasets, the Bilayered Neural Network architectures consistently achieve the highest accuracy and robustness with relatively lower complexity compared with the trilayered networks. This is particularly evident in the NF-ToN-IoT-V2 Filtered Dataset, NF-ToN-IoT-V2 Wrapper Dataset, and Edge-IIoTset Dataset Wrapper Dataset, where the Bilayered Neural Network 2 × 10 and 2 × 25 configurations demonstrate superior performance metrics with high accuracy, precision, recall, and F1-scores, underscoring their effectiveness and efficiency. These findings suggest that the bilayered architectures provide a balanced and highly effective solution for accurately detecting XSS attacks in IoT systems over 5G networks, achieving high performance with less complexity.

The confusion matrices presented in Figure 10 provide a comprehensive view of the performance of various neural network configurations during both the training and testing stages. Starting with the Narrow Neural Network 1 × 10 (a, b), the model performs well on the majority class (Benign), with 26,152 true positives and 10,813 true negatives during training, but there are 276 false positives and 184 false negatives. In the test stage, it shows 11,204 true positives and 4633 true negatives, with 121 false positives and 81 false negatives. Moving on to the Medium Neural Network 1 × 25 (c, d), there is a decrease in false negatives compared with the narrow network, with 26,197 true positives and 10,831 true negatives during training and 238 false positives and 159 false negatives. In the test stage, it achieves 11,233 true positives and 4644 true negatives, with 97 false positives and 65 false negatives. This indicates that a larger number of neurons in the single hidden layer has improved the model's ability to identify XSS attacks. However, false positives have increased slightly, which may be acceptable if reducing false negatives is a priority. The Wide Neural Network 1 × 100 (e, f) sees a further reduction in both false positives and false negatives, with 26,163 true positives and 10,817 true negatives during training, and 267 false positives and 178 false negatives. In the test stage, it shows 11,210 true positives and 4635 true negatives, with 116 false positives and 78 false negatives. The performance of the Medium Neural Network 1 × 25 is better between all single-layer models, highlighting its high ability to generalize from training to testing data and effectively balance the trade-off between sensitivity and specificity.

In the case of multi-layer networks, the Bilayered Neural Network achieved the highest performance between all models, starting with the 2 × 10 configuration (g, h) demonstrating exceptional accuracy during both the training and test stages, with 26,459 true positives and 10,940 true negatives during training and 16 false positives and 10 false negatives. In the test stage, it achieves 11,329 true positives and 4684 true negatives, with 16 false positives and 10 false negatives, effectively balancing in detection XSS attacks over IoT environments. Moving to a more complex 2 × 25 configuration (i, j), the Bilayered Neural Networks have a little reduction in performance with 26,454 true positives and 10,937 true negatives and very low errors of 20 false positives and 14 false negatives during training. In the test stage, it maintains this strong performance with 11,334 true positives and 4686 true negatives and minimal errors of 11 false positives and 8 false negatives, indicating excellent generalization and robust detection capability. The Trilayered Neural Networks, while still highly effective, show slightly higher error rates compared with their bilayered counterparts. The 3 × 25 configuration (m, n) records 26,239 true positives and 10,849 true negatives during training, with 202 false positives and 135 false negatives, and in the test stage, it achieves 11,233 true positives and 4644 true negatives, with 97 false positives and 65 false negatives. The 3 × 10 configuration (k, l) shows 26,247 true positives and 10,852 true negatives during training, with 196 false positives and 130 false negatives, and in the test stage, it records 11,246 true positives and 4649 true negatives, with 86 false positives and 58 false negatives. Across all stages and configurations, it is evident that as the network width and depth increase, they do not necessarily increase the model's performance in the detection process, as we observed in the experimental results of the NF-ToN-IoT-V2 Filtered Dataset, where bilayered models with a simple configuration of 2 × 10 obtained better results than their more complex counterparts. Therefore, the issue of complexity must be taken into consideration when adopting attack detection models.

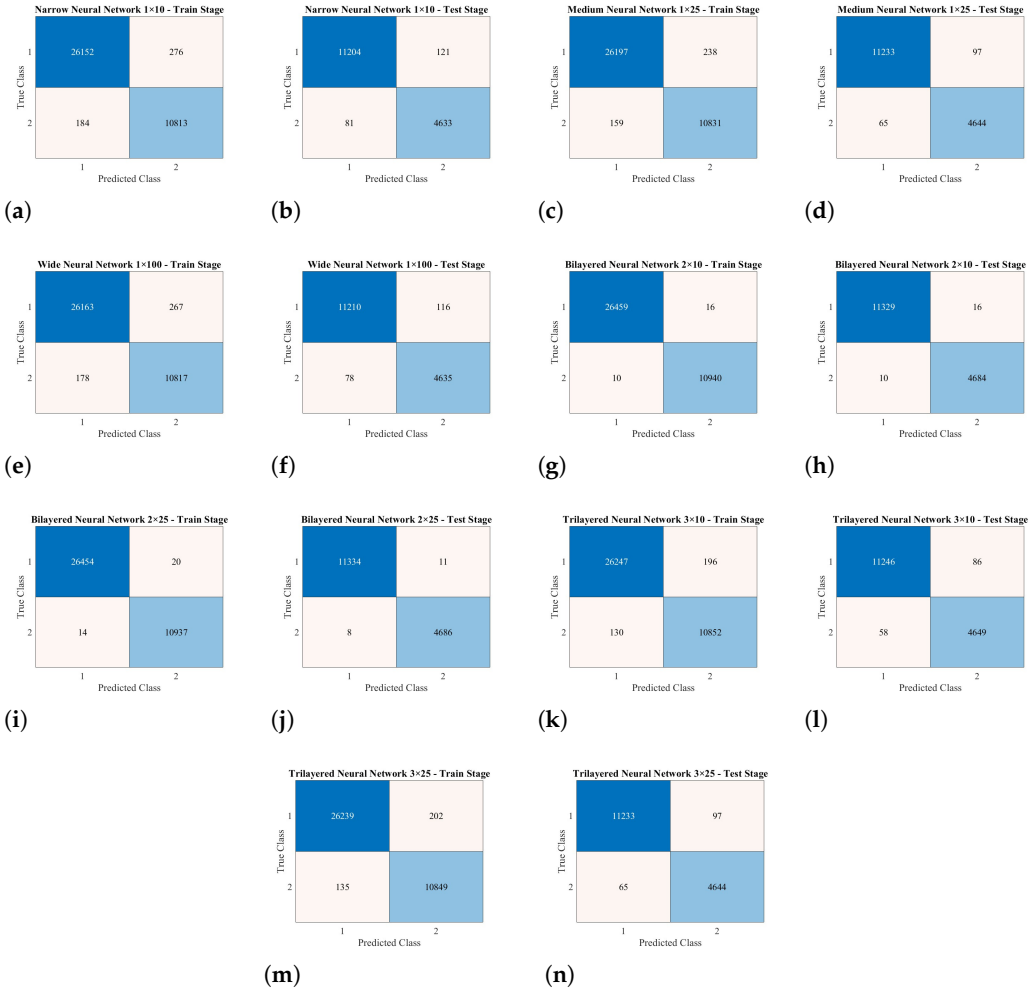


Figure 10. Confusion matrices for NF-ToN-IoT-V2 Filtered Dataset. (**a**) Narrow Neural Network 1 × 10-Train, (**b**) Narrow Neural Network 1 × 10-Test, (**c**) Medium Neural Network 1 × 25-Train, (**d**) Medium Neural Network 1 × 25-Test, (**e**) Wide Neural Network 1 × 100-Train, (**f**) Wide Neural Network 1 × 100-Test, (**g**) Bilayered Neural Network 2 × 10-Train, (**h**) Bilayered Neural Network 2 × 10-Test, (**i**) Bilayered Neural Network 2 × 25-Train, (**j**) Bilayered Neural Network 2 × 25-Test, (**k**) Trilayered Neural Network 3 × 10-Train, (**l**) Trilayered Neural Network 3 × 10-Test, (**m**) Trilayered Neural Network 3 × 25-Train, and (**n**) Trilayered Neural Network 3 × 25-Test.

The confusion matrices in Figure 11 show that the Narrow Neural Network 1 × 10 (a, b) achieves 25,808 true positives and 10,670 true negatives during training, but has 568 false positives and 379 false negatives; in testing, it has 11,132 true positives, 4602 true negatives, 183 false positives, and 122 false negatives. The Medium Neural Network 1 × 25 (c, d) improves with 26,133 true positives and 10,805 true negatives during training and 11,180 true positives and 4623 true negatives in testing, with fewer false positives and negatives than the narrow network. The Wide Neural Network 1 × 100 (e, f) shows 26,152 true positives and 10,813 true negatives during training and 11,208 true positives and 4634 true negatives in testing, making the Medium Neural Network 1 × 25 the most balanced single-layer model. For multi-layer networks, the Bilayered Neural Network 2 × 10 (g, h) achieves 26,335 true positives and 10,888 true negatives in training and 11,278 true positives and 4663 true negatives in testing, with low error rates. The 2 × 25 configuration (i, j) presents the best performance with 26,446 true positives and 10,934 true negatives in training and 11,328 true positives and 4684 true negatives in testing. The Trilayered Neural Networks show slightly higher error rates: the 3 × 25 configuration (k, l) has 26,403 true positives and 10,917 true negatives in training and 11,314 true pos-

itives and 4678 true negatives in testing, while the 3 × 10 configuration (m, n) achieves 26,414 true positives and 10,921 true negatives in training and 11,321 true positives and 4680 true negatives in testing. The Bilayered Neural Network 2 × 10 offers the best balance of simplicity and performance, achieving superior results in detecting XSS attacks.

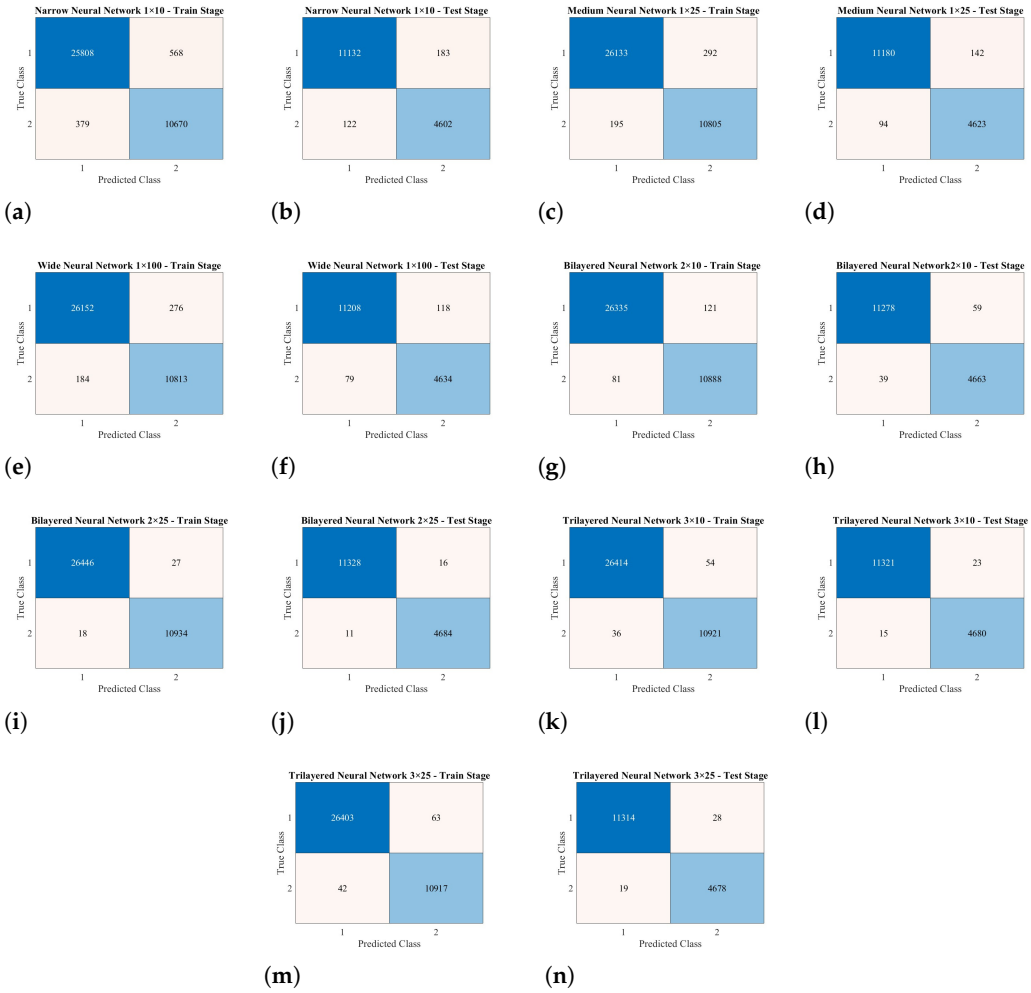


Figure 11. Confusion matrices for NF-ToN-IoT-V2 Wrapper Dataset. (**a**) Narrow Neural Network 1 × 10-Train, (**b**) Narrow Neural Network 1 × 10-Test, (**c**) Medium Neural Network 1 × 25-Train, (**d**) Medium Neural Network 1 × 25-Test, (**e**) Wide Neural Network 1 × 100-Train, (**f**) Wide Neural Network 1 × 100-Test, (**g**) Bilayered Neural Network 2 × 10-Train, (**h**) Bilayered Neural Network 2 × 10-Test, (**i**) Bilayered Neural Network 2 × 25-Train, (**j**) Bilayered Neural Network 2 × 25-Test, (**k**) Trilayered Neural Network 3 × 10-Train, (**l**) Trilayered Neural Network 3 × 10-Test, (**m**) Trilayered Neural Network 3 × 25-Train, and (**n**) Trilayered Neural Network 3 × 25-Test.

The confusion matrices presented in Figure 12 provide insights into the performance of various neural network configurations on the Edge-IIoTset Filtered Dataset. The Narrow Neural Network 1 × 10 (a, b) shows relatively higher error rates, with 16,442 true positives and 6798 true negatives during training and 7043 true positives and 2912 true negatives during testing. The Medium Neural Network 1 × 25 (c, d) improves upon this, recording 16,578 true positives and 6854 true negatives during training and 7105 true positives and 2938 true negatives during testing. The Wide Neural Network 1 × 100 (e, f) further enhances accuracy, with 16,666 true positives and 6891 true negatives during training and 7099 true positives and 2935 true negatives during testing. The Bilayered Neural Network 2 × 10 (g, h) demonstrates exceptional performance, achieving 16,699 true positives and 6904 true negatives during training and 7117 true positives and 2943 true negatives during

testing. The 2 × 25 configuration (i, j) shows even better results, with 16,898 true positives and 6,986 true negatives during training and 7226 true positives and 2988 true negatives during testing. The Trilayered Neural Networks 3 × 10 (k, l) and 3 × 25 (m, n) perform well but show slightly higher error rates compared with their bilayered counterparts. Overall, the Bilayered Neural Networks, particularly the 2 × 25 configuration, achieve the highest accuracy with lower complexity, emphasizing the importance of considering model complexity in neural network design for security applications.

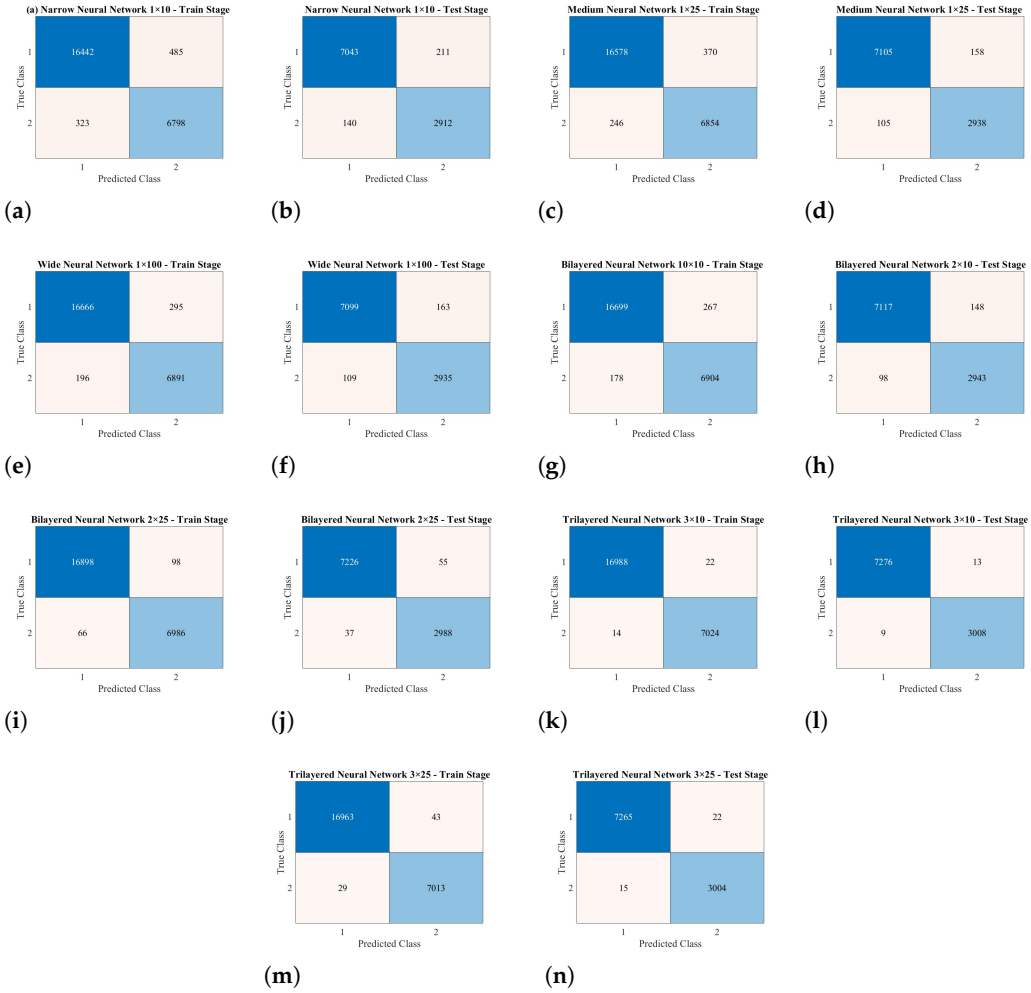
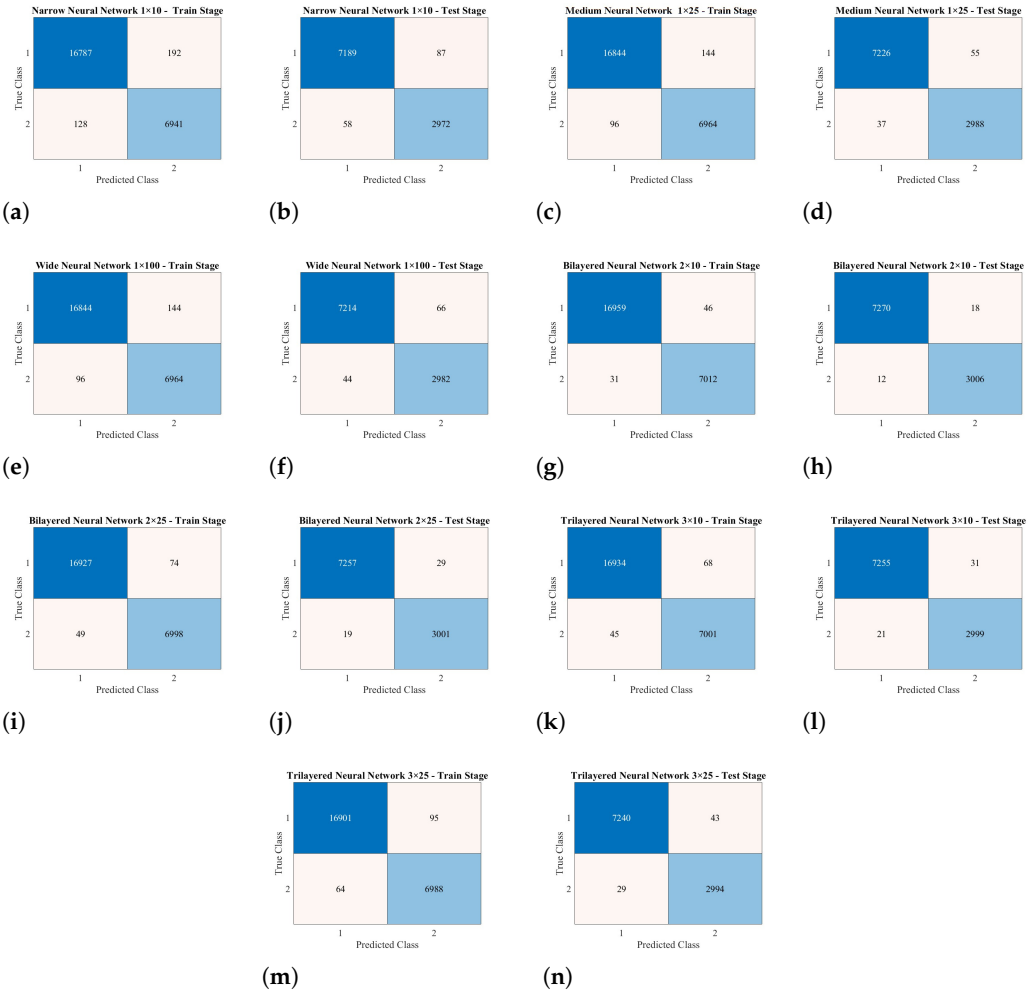


Figure 12. Confusion matrices for Edge-IIoTset Filtered Dataset. (**a**) Narrow Neural Network 1 × 10-Train, (**b**) Narrow Neural Network 1 × 10-Test, (**c**) Medium Neural Network 1 × 25-Train, (**d**) Medium Neural Network 1 × 25-Test, (**e**) Wide Neural Network 1 × 100-Train, (**f**) Wide Neural Network 1 × 100-Test, (**g**) Bilayered Neural Network 2 × 10-Train, (**h**) Bilayered Neural Network 2 × 10-Test, (**i**) Bilayered Neural Network 2 × 25-Train, (**j**) Bilayered Neural Network 2 × 25-Test, (**k**) Trilayered Neural Network 3 × 10-Train, (**l**) Trilayered Neural Network 3 × 10-Test, (**m**) Trilayered Neural Network 3 × 25-Train, and (**n**) Trilayered Neural Network 3 × 25-Test.

Finally, for the Edge-IIoTset Wrapper Dataset, Figure 13 shows the confusion matrices during the training stage. The Narrow Neural Network (1 × 10) (a) achieved 16,787 true positives and 6941 true negatives, while the Wide Neural Network (1 × 100) (b) slightly outperformed with 16,844 true positives and 6964 true negatives. The Medium Neural Network (1 × 25) (c) had similar results to the wide network. In the test stage, the Wide Neural Network (d) again showed strong performance with 7214 true positives and 2982 true negatives. The Bilayered Neural Network with a 2 × 10 configuration (e) during training achieved 16,959 true positives and 7012 true negatives and 7270 true positives

and 3006 true negatives in the test stage (f), indicating excellent detection capabilities. The 2 × 25 configuration (g) also performed well with 16,927 true positives and 6998 true negatives during training and 7257 true positives and 3001 true negatives during testing (h). The Trilayered Neural Network configurations, both 3 × 10 (i) and 3 × 25 (j), demonstrated robust performance, with the 3 × 10 configuration achieving 16,934 true positives and 7001 true negatives in training and 7255 true positives and 2999 true negatives in testing (k). The 3 × 25 configuration achieved 16,901 true positives and 6988 true negatives in training and 7240 true positives and 2994 true negatives in testing (l). The results indicate that while the complexity of the networks increases, the performance remains high, with the bilayered and trilayered networks showing particularly strong results in both the training and testing phases.



(**a**) (**b**) (**c**) (**d**)

(**e**) (**f**) (**g**) (**h**)

(**i**) (**j**) (**k**) (**l**)

(**m**) (**n**)

Figure 13. Confusion matrices for Edge-IIoTset Wrapper Dataset. (**a**) Narrow Neural Network 1 × 10-Train, (**b**) Narrow Neural Network 1 × 10-Test, (**c**) Medium Neural Network 1 × 25-Train, (**d**) Medium Neural Network 1 × 25-Test, (**e**) Wide Neural Network 1 × 100-Train, (**f**) Wide Neural Network 1 × 100-Test, (**g**) Bilayered Neural Network 2 × 10-Train, (**h**) Bilayered Neural Network 2 × 10-Test, (**i**) Bilayered Neural Network 2 × 25-Train, (**j**) Bilayered Neural Network 2 × 25-Test, (**k**) Trilayered Neural Network 3 × 10-Train, (**l**) Trilayered Neural Network 3 × 10-Test, (**m**) Trilayered Neural Network 3 × 25-Train, and (**n**) Trilayered Neural Network 3 × 25-Test.

5. Results of the ANOVA Test for Wrapper and Filtered NF-ToN-IoT-V2 and Edge-IIoTset Datasets

To investigate the performance of the DL models over filtered and wrapper NF-ToN-IoT-V2 and Edge-IIoTset datasets, we applied ANOVA tests to determine whether

statistically noteworthy accuracy differences exist in those seven models. Those are explained using the neural network example with the weight of the initial condition (also to point out that deep networks have many hyperparameters). Each model was trained ten times with random initial weights to make it robust and put this kind of variability into perspective. The results of these ANOVA tests are presented in Table 6.

Table 6. ANOVA test results for filtered and wrapper datasets for NF-ToN-IoT-V2 and Edge-IIoTset datasets.

NF-ToN-IoT-V2 Datasets						
Feature Selection	**Source**	**SS**	**df**	**MS**	**F**	**Prob > F**
Filtered	Groups	0.0750	6	0.0125	0.1381	0.9907
	Error	5.7026	63	0.0905	-	-
	Total	5.7776	69	-	-	-
Wrapper	Groups	0.0930	6	0.0155	0.1835	0.9804
	Error	5.3221	63	0.0845	-	-
	Total	5.4151	69	-	-	-
Edge-IIoTset Dataset						
Feature Selection	**Source**	**SS**	**df**	**MS**	**F**	**Prob > F**
Filtered	Groups	0.1077	6	0.0179	0.2020	0.9750
	Error	5.5953	63	0.0888	-	-
	Total	5.7030	69	-	-	-
Wrapper	Groups	0.1457	6	0.0243	0.2432	0.9603
	Error	6.2878	63	0.0998	-	-
	Total	6.4334	69	-	-	-

The ANOVA test results indicate that, for the NF-ToN-IoT-V2 and Edge-IIoTset datasets, the differences in accuracy between the seven neural network models are not statistically significant. For the filtered NF-ToN-IoT-V2 dataset, the between-groups sum of squares (samples) is 0.0750, while the within-groups sum of squares (error) is 5.7026. The sum of squares is 5.7776, with between-groups and within-groups degrees of freedom of 6 and 63, respectively. The mean square between the two groups is 0.0125, and the mean square within the groups is 0.0905, resulting in an F-statistic of 0.1381 and a p-value of 0.9907. Likewise, the NF-ToN-IoT-V2 pool dataset has a p-value of 0.9804, with the sum of squares between groups at 0.0930 and within groups at 5.3221. For the Edge-IIoTset dataset, the filtered dataset displays a p-value of 0.9750 with a between-groups sum of squares of 0.1077 and a within-groups sum of squares of 5.5953, while the pooled dataset has a p-value of 0.9603, with a between-groups sum of squares of 0.1457 and a within-groups sum of squares of 0.1457 and a within-groups sum of squares of 0.9603. At 6.2878, these high p-values indicate that any observed differences in model accuracy are likely the result of random chance rather than actual differences in model performance.

The box plots in Figure 14 visually support the ANOVA results, showing similar distributions of accuracy values across the models with overlapping interquartile ranges (IQRs) and medians. In the filtered NF-ToN-IoT-V2 dataset (Figure 14a), the models exhibit varying ranges of accuracy values, with some models like Model 1 and Model 3 showing higher variability and wider ranges, while Model 6 demonstrates more consistent performance with narrower ranges. This pattern is similarly observed in the wrapper NF-ToN-IoT-V2 dataset (Figure 14b). For the Edge-IIoTset dataset, both the filtered (Figure 14c) and wrapper (Figure 14d) plots indicate that while some models exhibit wider ranges of accuracy values, the overall distributions are quite similar across the models. However, these differences in variance do not translate into significant differences in overall performance, as the variance analysis (ANOVA) results indicate. The consistency across models and datasets suggests

that the training process is stable, providing a reasonable guarantee of accuracy no matter which model is chosen.

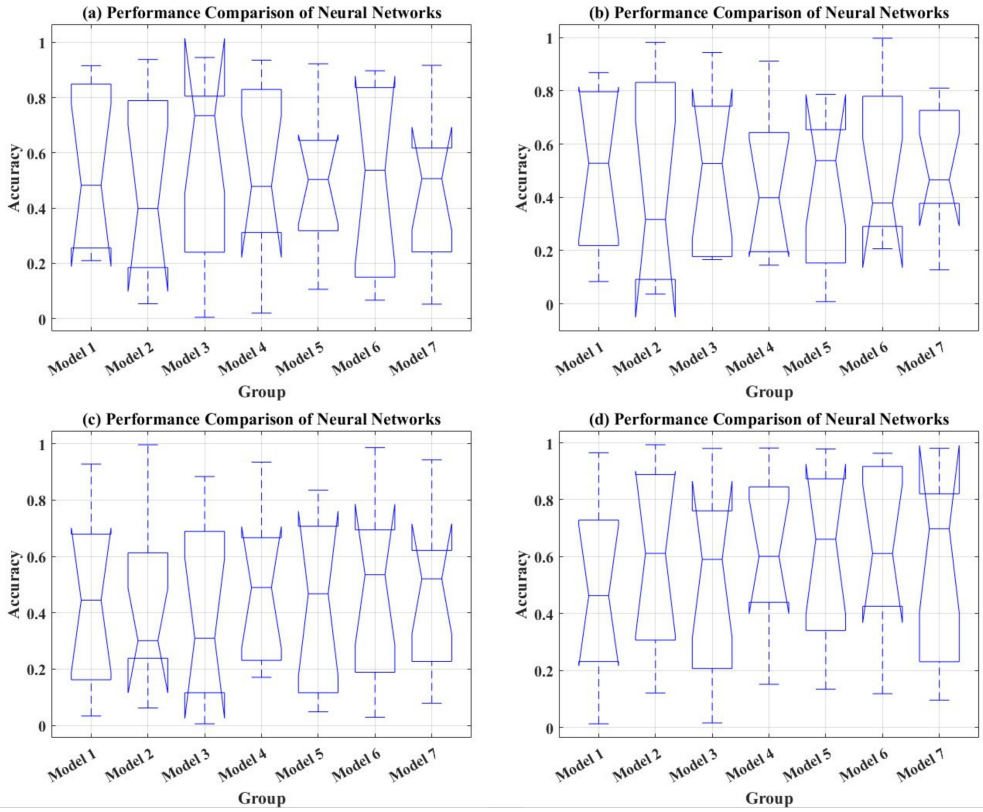


Figure 14. Performance comparison of neural networks' notched box plot: (**a**) filter NF-ToN-IoT-V2, (**b**) wrapper NF-ToN-IoT-V2, (**c**) filter Edge-IIoTset, and (**d**) wrapper Edge-IIoTset.

The results suggest that all seven models may be used interchangeably without fear of significant change in performance; this demonstrates the consistency of the neural network training process and details the robustness of the model across the datasets and feature selection techniques.

6. Comparison with Related Works

In this section, we compare our proposed XSS attack detection method with existing studies in the field, focusing on the datasets, feature selection methods, and performance outcomes. This comparison highlights the advancements and improvements offered by our approach, as shown in Table 7.

Our work advances the current state of the art by achieving a commendable high performance in detection accuracy. For the NF-ToN-IoT-V2 dataset, our Bilayered Neural Network achieved an impressive 99.84% accuracy, outperforming that of Duan et al. (2022) [23], who attained 95.70% using DLGNN, and that of Awad et al. (2022) [25], who reported 98% with a Random Forest model. Similarly, Yigit et al. (2023) [26] achieved 98.04% using an LSTM-AE model, and Sarhan et al. (2022) [27] reported 96.83% using Extra Trees. Our approach's use of advanced neural network architectures and a combination of filter and wrapper feature selection techniques clearly contributes to this superior performance. These results highlight the effectiveness of our method in utilizing comprehensive feature selection and sophisticated neural network structures to achieve higher accuracy in detecting XSS attacks.

Table 7. Evaluating our XSS injection cyberattack detection approach with related works.

Reference	Utilized Dataset	Method for Selecting and Extracting Features	Best Performance Achieved
[23]	NF-ToN-IoT-V2	-	95.70% using DLGNN.
[24]	AWID	Recursive feature elimination, constant removal	99% using Decision Trees, Random Forest, SVM
[25]	NF-ToN-IoT-V2	feature importance model	98% using RF
[26]	NF-ToN-IoT-V2	AutoFS and AutoCM	98.04% using LSTM-AE
[27]	NF-ToN-IoT-V2	-	96.83% using Extra Tree
[28]	Edge-IIoTset	-	85.48% using Decision Trees
[29]	Edge-IIoTset	-	77% using CNN
[30]	Edge-IIoTset	-	76.22% using SecurityBERT classifier
Proposed Method	**NF-ToN-IoT-V2, Edge-IIoTset**	**Filter and wrapper feature selection**	**99.84% for NF-ToN-IoT-V2 using Bilayered Neural Network and 99.79% for Edge-IIoTset using Trilayered Neural Network**

For the Edge-IIoTset dataset, our Trilayered Neural Network achieved a remarkable 99.79% accuracy, a substantial improvement over previous works. Awad et al. (2024) [28] achieved 85.48% using Decision Trees, Ahmed et al. (2024) [29] reported 77% with a CNN, and Ferrag et al. (2024) [30] achieved 76.22% using SecurityBERT. These comparisons underscore the robustness and efficacy of our method, particularly in handling complex datasets. The significant increase in detection accuracy, as shown in Figure 15, demonstrates the advantage of our approach in leveraging deep learning techniques to handle the intricate nature of IoT data. Integrating sophisticated neural network models and comprehensive feature selection methodologies sets a new benchmark for XSS attack detection in IoT systems, highlighting the potential for enhanced security in 5G networks. Our results confirm our approach's effectiveness and indicate its potential applicability in broader IoT security contexts, paving the way for future research and development in this critical area.

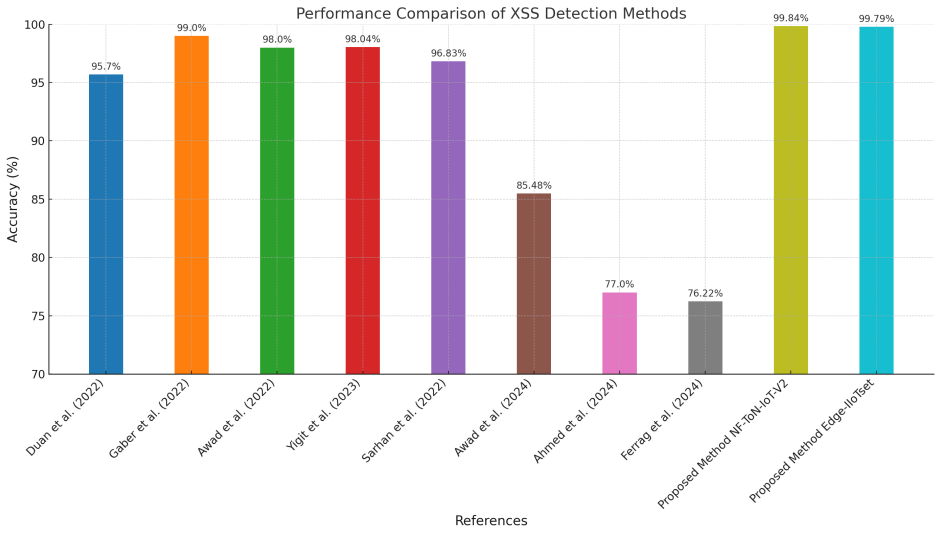


Figure 15. Comparing the detection accuracies of XSS injection attacks with previous studies. References: Duan et al. (2022) [23], Gaber et al. (2022) [24], Awad et al. (2022) [25], Yigit et al. (2023) [26], Sarhan et al. (2022) [27], Awad et al. (2024) [28], Ahmed et al. (2024) [29], Ferrag et al. (2024) [30].

7. Conclusions

Our research presents a cutting-edge solution for detecting and mitigating Cross-Site Scripting (XSS) attacks in IoT systems over 5G networks using Artificial Neural Networks (ANNs). We developed an innovative ANN-based approach that significantly outperforms

traditional methods, achieving high detection accuracies validated by extensive testing on the NF-ToN-IoT-v2 and Edge-IIoTset datasets. We optimized our models' performance by employing mutual information (MI) and recursive feature elimination (RFE) for feature selection, reducing computational demands while maintaining exceptional accuracy. Our Bilayered Neural Network (BLNN) and Trilayered Neural Network (TLNN) models reached accuracies of 99.84% and 99.79%, respectively, highlighting their superiority over existing methods. The robustness and reliability of our approach were further confirmed through ANOVA tests, which demonstrated statistically significant improvements in detection accuracy. This research sets a new standard for XSS attack detection in IoT environments, showcasing the effectiveness of sophisticated neural network architectures and comprehensive feature selection techniques. Our findings emphasize the critical role of advanced artificial intelligence models in enhancing IoT security, paving the way for future innovations in safeguarding IoT systems in the 5G era.

Author Contributions: Conceptualization, R.A. (Rabee Alqura'n), M.A. (Mahmoud AlJamal) and I.A.-A.; formal analysis, A.A., B.K. and M.A. (Mohammad Aljaidi); funding acquisition, R.A. (Rakan Alanaz); investigation, R.A. (Rabee Alqura'n) and A.A.; methodology, M.A. (Mahmoud AlJamal) and M.A. (Mohammad Aljaidi); project administration, A.A.; resources, I.A.-A. and B.K.; software, R.A. (Rabee Alqura'n) and M.A. (Mahmoud AlJamal); supervision, M.A.(Mohammad Aljaidi); writing—original draft, R.A. (Rabee Alqura'n), M.A. (Mahmoud AlJamal) and I.A.-A.; writing—review and editing, A.A., B.K., M.A. (Mohammad Aljaidi) and R.A. (Rakan Alanaz). All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by Northern Border University, Arar, Saudi Arabia, through the project number "NBU-FFR-2024-1661-04".

Data Availability Statement: Research data will be available on individual requests to the corresponding author.

Acknowledgments: The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, Saudi Arabia.

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

References

1. Gaba, P.; Raw, R.S.; Kaiwartya, O.; Aljaidi, M. B-SAFE: Blockchain-Enabled Security Architecture for Connected Vehicle Fog Environment. *Sensors* **2024**, *24*, 1515. [CrossRef] [PubMed]
2. Yadav, N.; Pande, S.; Khamparia, A.; Gupta, D. Intrusion detection system on IoT with 5G network using deep learning. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 9304689 . [CrossRef]
3. Almiani, M.; AbuGhazleh, A.; Jararweh, Y.; Razaque, A. DDoS detection in 5G-enabled IoT networks using deep Kalman backpropagation neural network. *Int. J. Mach. Learn. Cybern.* **2021**, *12*, 3337–3349. [CrossRef]
4. Andrews, J.G.; Buzzi, S.; Choi, W.; Hanly, S.V.; Lozano, A.; Soong, A.C.; Zhang, J.C. What will 5G be? *IEEE J. Sel. Areas Commun.* **2014**, *32*, 1065–1082. [CrossRef]
5. De Donno, M.; Giaretta, A.; Dragoni, N.; Bucchiarone, A.; Mazzara, M. Cyber-storms come from clouds: Security of cloud computing in the IoT era. *Future Internet* **2019**, *11*, 127. [CrossRef]
6. Mohammed, M.H.; Rasheed, A.F. A secured Architecture of Internet of Things (IoT) in the 5G age. In *New Trends in Network Cyber Security (Part 1)*; Ali, Q., Alhafid, A., Hussein, S., Al-Tayyar, H., Alabasy, M.E., Eds.; LAP LAMBERT Academic Publishing: Saarbrücken, Germany, 2021; ISBN: 9783639861488.
7. Aljaidi, M.; Alsarhan, A.; Samara, G.; AL-Khassawneh, Y.A.; Al-Gumaei, Y.A.; Aljawawdeh, H.; Alqammaz, A. A Critical Evaluation of A Recent Cybersecurity Attack on iTunes Software Updater. In Proceedings of the 2022 International Engineering Conference on Electrical, Energy, and Artificial Intelligence (EICEEAI), Zarqa, Jordan, 6–8 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.
8. Kim, J.; Park, J. Enhancing Security of Web-Based IoT Services via XSS Vulnerability Detection. *Sensors* **2023**, *23*, 9407. [CrossRef] [PubMed]
9. Chaudhary, P.; Gupta, B.B.; Chui, K.T.; Yamaguchi, S. Shielding smart home iot devices against adverse effects of xss using ai model. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 10–12 January 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–5.
10. Chaudhary, P.; Gupta, B.; Singh, A. XSS Armor: Constructing XSS defensive framework for preserving big data privacy in internet-of-things (IoT) networks. *J. Circuits Syst. Comput.* **2022**, *31*, 2250222. [CrossRef]

11. Kaur, J.; Garg, U.; Bathla, G. Detection of cross-site scripting (XSS) attacks using machine learning techniques: A review. *Artif. Intell. Rev.* **2023**, *56*, 12725–12769. [CrossRef]
12. Nair, S.S. Securing Against Advanced Cyber Threats: A Comprehensive Guide to Phishing, XSS, and SQL Injection Defense. *J. Comput. Sci. Technol. Stud.* **2024**, *6*, 76–93. [CrossRef]
13. Hannousse, A.; Yahiouche, S.; Nait-Hamoud, M.C. Twenty-two years since revealing cross-site scripting attacks: A systematic mapping and a comprehensive survey. *Comput. Sci. Rev.* **2024**, *52*, 100634. [CrossRef]
14. Tan, X.; Xu, Y.; Wu, T.; Li, B. Detection of reflected XSS vulnerabilities based on paths-attention method. *Appl. Sci.* **2023**, *13*, 7895. [CrossRef]
15. Santithanmanan, K.; Kirimasthong, K.; Boongoen, T. Machine Learning Based XSS Attacks Detection Method. In Proceedings of the UK Workshop on Computational Intelligence, Birmingham, UK, 6–8 September 2023; Springer: Berlin/Heidelberg, Germany, 2023; pp. 418–429.
16. Kholidy, H.A. Multi-layer attack graph analysis in the 5g edge network using a dynamic hexagonal fuzzy method. *Sensors* **2021**, *22*, 9. [CrossRef]
17. Anand, A.; Rani, S.; Anand, D.; Aljahdali, H.M.; Kerr, D. An efficient CNN-based deep learning model to detect malware attacks (CNN-DMA) in 5G-IoT healthcare applications. *Sensors* **2021**, *21*, 6346. [CrossRef] [PubMed]
18. Noman, H.A.; Abu-Sharkh, O.M. Code injection attacks in wireless-based Internet of Things (IoT): A comprehensive review and practical implementations. *Sensors* **2023**, *23*, 6067. [CrossRef]
19. Saini, H.K.; Poriye, M.; Goyal, N. A survey on security threats and network vulnerabilities in Internet of Things. In *Big Data Analytics in Intelligent IoT and Cyber-Physical Systems*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 297–314.
20. Kalhoro, S.; Shaikh, F.B.; Kalhoro, A.; Abbasi, J.U.R.; Ayyasamy, R.K. An Overview of Security Attacks in 5G Enabled Technologies: Applications and Use Case Scenarios. *ISeCure* **2024**, *16*, 17–35.
21. Kaushik, K.; Ouaissa, M.; Chaudhary, A. *Advanced Techniques and Applications of Cybersecurity and Forensics*; CRC Press: Boca Raton, FL, USA, 2024.
22. Bhardwaj, A.; Bharany, S.; Abulfaraj, A.W.; Ibrahim, A.O.; Nagmeldin, W. Fortifying home IoT security: A framework for comprehensive examination of vulnerabilities and intrusion detection strategies for smart cities. *Egypt. Inform. J.* **2024**, *25*, 100443. [CrossRef]
23. Duan, G.; Lv, H.; Wang, H.; Feng, G. Application of a dynamic line graph neural network for intrusion detection with semisupervised learning. *IEEE Trans. Inf. Forensics Secur.* **2022**, *18*, 699–714. [CrossRef]
24. Gaber, T.; El-Ghamry, A.; Hassanien, A.E. Injection attack detection using machine learning for smart IoT applications. *Phys. Commun.* **2022**, *52*, 101685. [CrossRef]
25. Awad, M.; Fraihat, S.; Salameh, K.; Al Redhaei, A. Examining the suitability of NetFlow features in detecting IoT network intrusions. *Sensors* **2022**, *22*, 6164. [CrossRef] [PubMed]
26. Yigit, Y.; Chrysoulas, C.; Yurdakul, G.; Maglaras, L.; Canberk, B. Digital twin-empowered smart attack detection system for 6g edge of things networks. *arXiv* **2023**, arXiv:2310.03554.
27. Sarhan, M.; Layeghy, S.; Portmann, M. Towards a standard feature set for network intrusion detection system datasets. In *Mobile Networks and Applications*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 1–14.
28. Awad, O.F.; Hazim, L.R.; Jasim, A.A.; Ata, O. Enhancing Iiot Security with Machine Learning and Deep Learning for Intrusion Detection. *Malays. J. Comput. Sci.* **2024**, *37*, 140–154.
29. Ahmed, Z.; Askar, S.S. EdgeGuard: Machine Learning for Proactive Intrusion Detection on Edge Networks. *Artif. Intell. Cybersecur.* **2024**, *1*, 37–43.
30. Ferrag, M.A.; Ndhlovu, M.; Tihanyi, N.; Cordeiro, L.C.; Debbah, M.; Lestable, T.; Thandi, N.S. Revolutionizing cyber threat detection with large language models: A privacy-preserving bert-based lightweight model for iot/iiot devices. *IEEE Access* **2024**, *12*, 23733–23750. [CrossRef]
31. Ferrag, M.A.; Friha, O.; Hamouda, D.; Maglaras, L.; Janicke, H. Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. *IEEE Access* **2022**, *10*, 40281–40306. [CrossRef]
32. Li, P.; Rao, X.; Blase, J.; Zhang, Y.; Chu, X.; Zhang, C. Cleanml: A benchmark for joint data cleaning and machine learning [experiments and analysis]. *arXiv* **2019**, arXiv:1904.09483.
33. Singh, D.; Singh, B. Investigating the impact of data normalization on classification performance. *Appl. Soft Comput.* **2020**, *97*, 105524. [CrossRef]
34. Dahouda, M.K.; Joe, I. A deep-learned embedding technique for categorical features encoding. *IEEE Access* **2021**, *9*, 114381–114391. [CrossRef]
35. AlJamal, M.; Mughaid, A.; Bani-Salameh, H.; Alzubi, S.; Abualigah, L. Optimizing risk mitigation: A simulation-based model for detecting fake IoT clients in smart city environments. *Sustain. Comput. Inform. Syst.* **2024**, 101019. [CrossRef]
36. Roy, K.; Farid, D.M. An Adaptive Feature Selection Algorithm for Student Performance Prediction. *IEEE Access* **2024**, *12*, 75577–75598. [CrossRef]
37. Awad, M.; Fraihat, S. Recursive feature elimination with cross-validation with decision tree: Feature selection method for machine learning-based intrusion detection systems. *J. Sens. Actuator Netw.* **2023**, *12*, 67. [CrossRef]
38. Bianchini, M.; Scarselli, F. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Trans. Neural Networks Learn. Syst.* **2014**, *25*, 1553–1565. [CrossRef] [PubMed]

39. Zhang, G.; Band, S.S.; Ardabili, S.; Chau, K.W.; Mosavi, A. Integration of neural network and fuzzy logic decision making compared with bilayered neural network in the simulation of daily dew point temperature. *Eng. Appl. Comput. Fluid Mech.* **2022**, *16*, 713–723. [CrossRef]
40. Khan, M.U.; Samer, S.; Alshehri, M.D.; Baloch, N.K.; Khan, H.; Hussain, F.; Kim, S.W.; Zikria, Y.B. Artificial neural network-based cardiovascular disease prediction using spectral features. *Comput. Electr. Eng.* **2022**, *101*, 108094. [CrossRef]
41. Mughaid, A.; AlJamal, M.; Issa, A.A.; AlJamal, M.; Alquran, R.; AlZu'bi, S.; Abutabanjeh, A.A. Enhancing cybersecurity in scada iot systems: A novel machine learning-based approach for man-in-the-middle attack detection. In Proceedings of the 2023 3rd Intelligent Cybersecurity Conference (ICSC), San Antonio, TX, USA, 23–25 October 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 74–79.
42. Mughaid, A.; Alqahtani, A.; AlZu'bi, S.; Obaidat, I.; Alqura'n, R.; AlJamal, M.; AL-Marayah, R. Utilizing Machine Learning Algorithms for Effectively Detection IoT DDoS Attacks. In Proceedings of the International Conference on Advances in Computing Research, Orlando, FL, USA, 8–10 May 2023; Springer: Berlin/Heidelberg, Germany, 2023; pp. 617–629.
43. Bradley, A.; Duin, R.; Paclik, P.; Landgrebe, T. Precision-recall operating characteristic (P-ROC) curves in imprecise environments. In Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), Hong Kong, China, 20–24 August 2006; IEEE: Piscataway, NJ, USA, 2006; Volume 4, pp. 123–127.
44. Elmrabit, N.; Zhou, F.; Li, F.; Zhou, H. Evaluation of machine learning algorithms for anomaly detection. In Proceedings of the 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), Dublin, Ireland, 15–19 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–8.
45. Ross, A.; Willson, V.L. One-way ANOVA. In *Basic and Advanced Statistical Tests: Writing Results Sections and Creating Tables and Figures*; Sense Publishers: Rotterdam, The Netherlands, 2017; pp. 21–24.