future internet

Article

A Deep Learning Approach for Multiclass Attack Classification in IoT and IIoT Networks Using Convolutional Neural Networks

Ali Abdi Seyedkolaei [1,*], Fatemeh Mahmoudi [2] and José García [3,*]

[1] Department of Computer Engineering, University of Mazandaran (UMZ), Babolsar 47416-13534, Iran
[2] Computer Engineering Faculty of Engineering & Technology, University of Mazandaran (UMZ), Babolsar 47416-13534, Iran; fateme.mahmoudi2001@gmail.com
[3] Aragon Engineering Research Institute (I3A), School of Engineering and Architecture (EINA), University of Zaragoza (UINZAR), 50018 Zaragoza, Spain
* Correspondence: a.abdi@umz.ac.ir (A.A.S.); jogarmo@unizar.es (J.G.)

Abstract: The rapid expansion of the Internet of Things (IoT) and industrial Internet of Things (IIoT) ecosystems has introduced new security challenges, particularly the need for robust intrusion detection systems (IDSs) capable of adapting to increasingly sophisticated cyberattacks. In this study, we propose a novel intrusion detection approach based on convolutional neural networks (CNNs), designed to automatically extract spatial patterns from network traffic data. Leveraging the DNN-EdgeIIoT dataset, which includes a wide range of attack types and traffic scenarios, we conduct comprehensive experiments to compare the CNN-based model against traditional machine learning techniques, including decision trees, random forests, support vector machines, and K-nearest neighbors. Our approach consistently outperforms baseline models across multiple performance metrics—such as F1 score, precision, and recall—in both binary (benign vs. attack) and multiclass settings (6-class and 15-class classification). The CNN model achieves F1 scores of 1.00, 0.994, and 0.946, respectively, highlighting its strong generalization ability across diverse attack categories. These results demonstrate the effectiveness of deep-learning-based IDSs in enhancing the security posture of IoT and IIoT infrastructures, paving the way for intelligent, adaptive, and scalable threat detection systems.

Keywords: intrusion detection system (IDS); Internet of Things (IoT); industrial IoT (IIoT); deep learning; convolutional neural networks (CNN); network security

Academic Editors: Olusola Tolulope Odeyomi and Temitayo Olowu

Received: 14 April 2025
Revised: 20 May 2025
Accepted: 20 May 2025
Published: 22 May 2025

Citation: Seyedkolaei, A.A.; Mahmoudi, F.; García, J. A Deep Learning Approach for Multiclass Attack Classification in IoT and IIoT Networks Using Convolutional Neural Networks. *Future Internet* **2025**, *17*, 230. https://doi.org/10.3390/fi17060230

1. Introduction

The Internet of Things (IoT) constitutes a network that facilitates the connection of physical objects with computing capabilities to the Internet [1]. As illustrated in Figure 1, the IoT finds versatile applications across diverse industries such as smart homes and cities, energy and productivity, transportation, production and logistics, livestock and animals, agriculture, and the insurance sector [2]. While the IoT offers enhanced efficiency through intelligent and remote management, the absence of adequate security measures leads to increased security risks, such as cyberattacks. Consequently, devices connected to IoT networks become vulnerable to such attacks [3]. Security within IoT networks and strategies to mitigate the risks of cyberattacks represent prominent topics in cybersecurity. Specific IoT applications, particularly those within the industrial IoT (IIoT), involve mission-critical operations that necessitate elevated security measures.

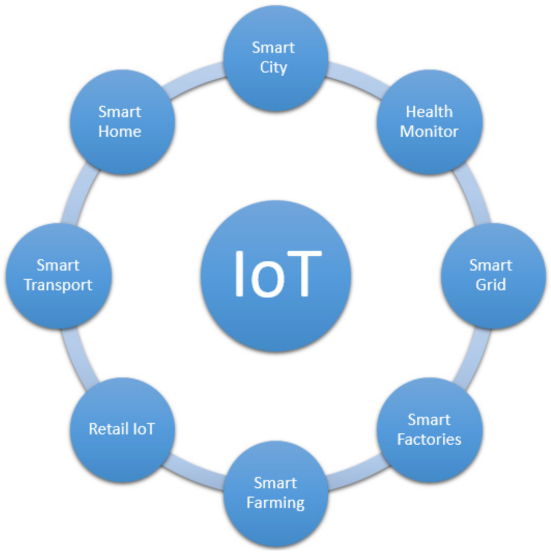

Figure 1. Application of the Internet of Things in different industries [2].

Securing IoT and IIoT networks presents unique challenges due to their reliance on lightweight communication protocols and constrained computational resources. Traditional security mechanisms like firewalls and encryption, while essential for data confidentiality and integrity, often demand significant processing power and storage, making them impractical for these environments. Moreover, attacks such as DDoS or protocol-level exploits can bypass encrypted channels, exposing vulnerabilities that encryption alone cannot address. To overcome these limitations, a complementary approach is critical. The intrusion detection system (IDS) augments encryption by monitoring traffic patterns to detect and mitigate threats that evade conventional safeguards. This dual-layered strategy ensures robust security without overwhelming limited resources, addressing the inherent challenges of IoT/IIoT ecosystems while enhancing resilience against evolving cyber threats [4,5]. As shown in Figure 2, intrusion detection systems monitor system or network events to help to identify malicious activities [6]. Assessing intrusion detection techniques is essential, with datasets being crucial for measuring the accuracy and effectiveness of security methods. However, the limited availability of real-world datasets for evaluating intrusion detection approaches in IoT and IIoT networks poses a considerable challenge.

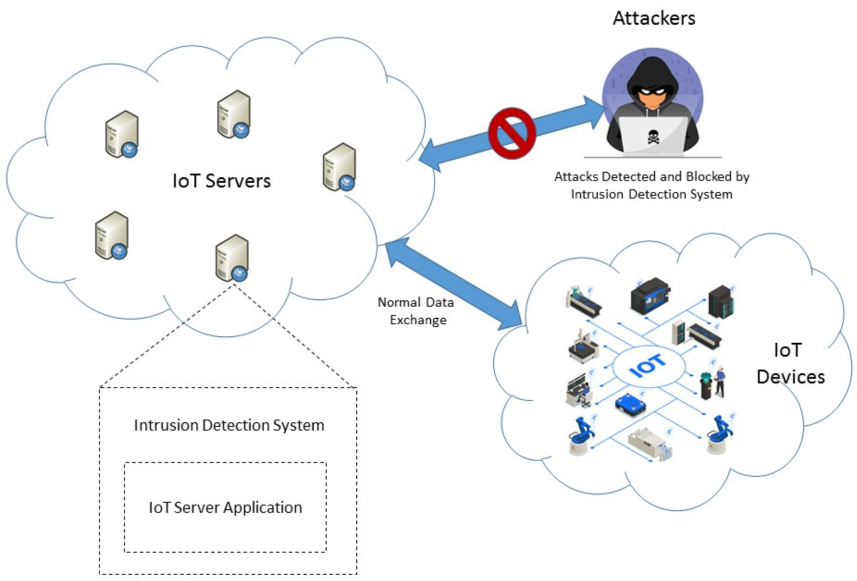
Attackers
IoT Servers
Attacks Detected and Blocked by Intrusion Detection System
Normal Data Exchange
IoT Devices
IOT
Intrusion Detection System
IoT Server Application

Figure 2. Application of an IDS in Internet of Things networks to mitigate attacks [6].

The growing prevalence of threats and risks in IoT and IIoT networks highlights the need for effective detection techniques and traffic attack prevention mechanisms in these environments. Machine learning is one such approach, providing a powerful and efficient alternative to traditional security methods that depend on manual oversight and coding. However, selecting an appropriate dataset is crucial for effective training. Therefore, using datasets that accurately represent real network events is essential for intrusion detection in IoT and IIoT networks.

The rapid expansion of IoT/IIoT networks necessitates robust IDSs capable of adapting to sophisticated cyberattacks. To address the scarcity of representative datasets for IoT/IIoT security research, we leverage the DNN-EdgeIIoT dataset [7], a comprehensive benchmark specifically designed for machine-learning-based IDSs. Introduced by Ferrag et al. (2022) [7], this dataset captures diverse attack scenarios (e.g., DDoS, SQL injection, malware) and normal traffic from a realistic IoT/IIoT testbed, spanning 2.2 million connection samples. Unlike conventional datasets, DNN-EdgeIIoT incorporates protocol-specific features (e.g., MQTT, HTTP) and system-level metrics (e.g., CPU usage), making it uniquely suited for evaluating spatial and temporal attack patterns in resource-constrained environments. Its inclusion of 14 attack types across five threat categories ensures alignment with real-world IoT/IIoT security challenges, addressing the critical gaps in existing datasets. The main contributions of this paper include the following:

- Designing the foundation of an intrusion detection system using deep learning techniques through a convolutional neural network approach.
- A thorough review and analysis of the DNN-EdgeIIoT dataset, showcasing the superior performance of the proposed method in intrusion detection.

The structure of the paper is organized as follows: Section 2 provides a review of the prior work on the design of intrusion detection systems in IoT and IIoT networks. Section 3 details the proposed methodology for developing the intrusion detection system. In Section 4, we present a review and analysis of the experiments conducted using the DNN-EdgeIIoT dataset. Finally, Section 5 discusses the results of the experiments and offers recommendations for future research.

2. Related Works

In this section, we review research on security within Internet of Things (IoT) and industrial Internet of Things (IIoT) networks. Ferrag et al. [7] introduced the Edge-IIoTset dataset, specifically designed for machine-learning-based intrusion detection systems. They also analyzed fourteen attack types relevant to IoT and IIoT networks, categorizing them into five distinct attack groups. Leander et al. [8] examined the primary cybersecurity challenges faced by IoT systems, utilizing a simplified threat model based on the STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) method to investigate network security issues. Additionally, Dhirani et al. [9] contributed to the Industry 4.0 Cyber Security Project by proposing effective strategies to secure Machine-to-Machine (M2M) communications in industrial IoT networks.

Kasongo et al. [10] utilized a genetic algorithm to develop an intrusion detection system specifically for industrial IoT networks. Their approach incorporated a random forest (RF) model within the fitness function of the genetic algorithm. The proposed model was tested for intrusion detection across various classifiers, including RF, Linear Regression (LR), Naïve Bayes (NB), decision tree (DT), Extra-Trees (ET), and Extreme Gradient Boosting (XGB). Vaiyapuri et al. [11] employed a deep-learning-based IDS technique for industrial IoT networks, presenting different methods for intrusion detection and performing a comparative analysis on the dataset. Mendonça et al. [12] investigated intrusion detection using deep learning and proposed a Sparse Evolutionary Training (SET)-based prediction model. They thoroughly evaluated their model's performance using metrics such as accuracy, precision, recall, and F1 score, comparing it with other state-of-the-art algorithms.

Yao et al. [13] explored detection methods and IDS architectures, proposing a hybrid IDS architecture combined with a machine-learning-based intrusion detection approach. Abdel-Basset et al. [14] introduced a forensics-driven deep learning (DL) model for intrusion detection in IIoT networks. Their model incorporated a local gated recurrent unit and added an MHA layer to capture and learn key features. Le et al. [15] presented an XGBoost model designed for IDSs in IIoT networks, specifically addressing unbalanced datasets. They assessed the effectiveness and robustness of their method using two unbalanced datasets, with the XGBoost model achieving impressive attack detection results, yielding F1 scores of 99.9% and 99.87% across both datasets, demonstrating its strong performance in intrusion detection for unbalanced multiclass datasets in IIoT networks.

Soliman et al. [16] introduced an intelligent detection system for identifying cyber-attacks in IIoT networks, utilizing Singular Value Decomposition (SVD) to reduce data dimensionality and improve recognition accuracy. They also employed the Synthetic Minority Oversampling Technique (SMOTE) to tackle overfitting and underfitting issues, thereby minimizing biased classifications. Additionally, they evaluated several machine learning and deep learning algorithms for both binary and multiclass classification tasks. Mohyeddine et al. [17] developed an intrusion detection approach to strengthen IIoT security through machine learning. Their work focused on enhancing feature selection and dimensionality reduction techniques to improve recognition rates and model accuracy. To reduce computational cost and time complexity in high-dimensional datasets, they recommended using Pearson's correlation coefficient (PCC) for feature selection and Isolation Forest (IF) to remove outliers. Altunay et al. [18] proposed three distinct models utilizing convolutional neural networks (CNN), Long Short-Term Memory (LSTM), and a hybrid CNN + LSTM architecture to create an intrusion detection system for IIoT networks.

3. Materials and Methods

This section details the materials and methodologies employed to develop and evaluate our CNN-based intrusion detection system tailored for IoT/IIoT networks. We begin by introducing the DNN-EdgeIIoT dataset [7], a comprehensive benchmark encompassing diverse attack scenarios and normal traffic flows, specifically designed for IoT/IIoT security research. Next, we outline the architecture and configuration of the convolutional neural network (CNN), emphasizing its suitability for spatial feature extraction and multiclass classification. Finally, we describe the experimental setup, including preprocessing steps, performance metrics, and comparative analyses with traditional machine learning methods. This section is structured to guide readers through the dataset's characteristics (Section 3.1), the CNN model's design rationale (Section 3.2), and the evaluation framework (Section 3.3), ensuring reproducibility and transparency in our approach.

3.1. The DNN-EdgeIIoT Dataset

The proposed CNN-based intrusion detection system was evaluated using the DNN-EdgeIIoT dataset [7], a comprehensive benchmark generated from a specialized IoT/IIoT testbed. This testbed incorporates diverse devices and protocols, including low-cost digital sensors (e.g., temperature, humidity, ultrasonic, water level, soil moisture, heart rate, and flame sensors), actuators, and edge/cloud configurations. The dataset comprises 2,219,201 connection samples, representing both normal communication traffic (72.8%) and 14 attack types (27.2%) relevant to IoT/IIoT environments. These attacks are categorized into five primary threat groups: DoS/DDoS, Man-in-the-Middle (MITM), injection, malware, and scanning-based attacks, with further sub-types such as DDoS_UDP, SQL_injection, and ransomware.

(1) *Dataset Structure and Features*: Each sample corresponds to a unique traffic flow characterized by 63 features extracted using Zeek 4.0.1 and TShark 3.4.4 protocol analyzers. These features include the following:
  - *Protocol-specific attributes*: Packet rates, connection duration, protocol type (e.g., MQTT, HTTP, DNS), payload size, and flow state flags.
  - *Statistical metrics*: Mean/standard deviation of packet intervals, bytes transmitted, and system resource usage (CPU, memory).
  - *Behavioral indicators*: Alerts triggered during communication (e.g., suspicious port access, malformed packets).

(2) *Labels and Classification Tasks*: The dataset includes the following two labels:
  - *Attack label*: Binary classification (0 = normal, 1 = attack).
  - *Attack type*: Multiclass classification, specifying the exact attack type (e.g., DDoS_ICMP, XSS). This allows the problem to be approached as binary (2-class), grouped (6-class), or fine-grained (15-class) classification, with distributions detailed in Tables 1–3.

(3) *Data Splitting and Attack Identification*: To ensure balanced evaluation, the dataset was partitioned using stratified random sampling, as follows:
  - Training set: A total of 1,775,361 samples (80%—1,615,643 normal, 603,558 attacks).
  - Test set: A total of 443,840 samples (20%—272,800 normal, 109,134 attacks).

Each traffic flow is self-contained, enabling real-time intrusion detection without requiring sequential analysis. For instance, DDoS attacks are identifiable through abrupt packet rate spikes, while SQL injections are flagged via anomalous payload patterns.

This dataset's realism, protocol diversity, and granular feature engineering make it uniquely suited for evaluating spatial and temporal attack behaviors in IoT/IIoT networks.

Table 1. Distribution of 2-class samples.

Classes	**Samples**
Normal	1,615,643
Attack	603,558

Table 2. Distribution of 6-class samples.

Classes	**Samples**
Normal	1,615,643
DDoS-based attack	337,977
Injection-based attack	51,203
MITM-based attack	1214
Malware-based attack	10,925
Scanning-based attack	22,564

Table 3. Distribution of 15-class samples.

Classes	Samples
Normal	1,615,643
Backdoor-based attack	24,862
DDoS_HTTP-based attack	49,911
DDoS_ICMP-based attack	116,436
DDoS_TCP-based attack	50,062
DDoS_UDP-based attack	121,568
Fingerprinting-based attack	1001
MITM-based attack	1214
Password-based attack	50,153
Port_Scanning-based attack	22,564
Ransomware-based attack	10,925
SQL_injection-based attack	51,203
Uploading-based attack	37,634
Vulnerability_scanner-based attack	50,110
XSS-based attack	15,915

3.2. Convolutional Neural Network (CNN)-Based Methodology for Intrusion Detection

In this section, we introduce the proposed method for designing an intrusion detection system tailored for IoT and IIoT networks. While numerous approaches exist for implementing intrusion detection in these environments, traditional machine learning algorithms often face limitations in scalability and accuracy when applied to large-scale, high-dimensional IoT traffic. Consequently, deep learning techniques—particularly convolutional neural networks (CNNs)—offer a more effective solution for attack detection.

CNNs are well-suited for this task due to their ability to identify spatial correlations among adjacent features within network traffic data, such as packet rates, connection durations, and protocol flags. These convolutional filters excel at capturing local patterns, which are often overlooked by classical models. In addition, CNNs handle high-dimensional input effectively through max pooling layers that reduce feature dimensions while preserving salient information, thus improving generalization and reducing the risk of overfitting.

Their hierarchical structure enables robust multiclass classification, making CNNs capable of distinguishing between normal traffic and various types of attacks. Prior works, including those by Altunay et al. [18] and Vaiyapuri et al. [11], support the superiority of CNNs in the IoT intrusion detection domain. Although hybrid models like CNN + LSTM can enhance performance, we selected a pure CNN to minimize computational overhead, making the system more suitable for deployment in resource-constrained IoT/IIoT settings.

The proposed detection method includes four key stages: (1) preprocessing, (2) data splitting, (3) feature extraction and classification via CNN, and (4) model evaluation. As illustrated in Figure 3, the CNN consists of convolutional, pooling, and fully connected layers. The convolutional layer serves as the core, identifying relevant local features. The pooling layer reduces data dimensionality, and the fully connected layer acts as the final classifier [19].

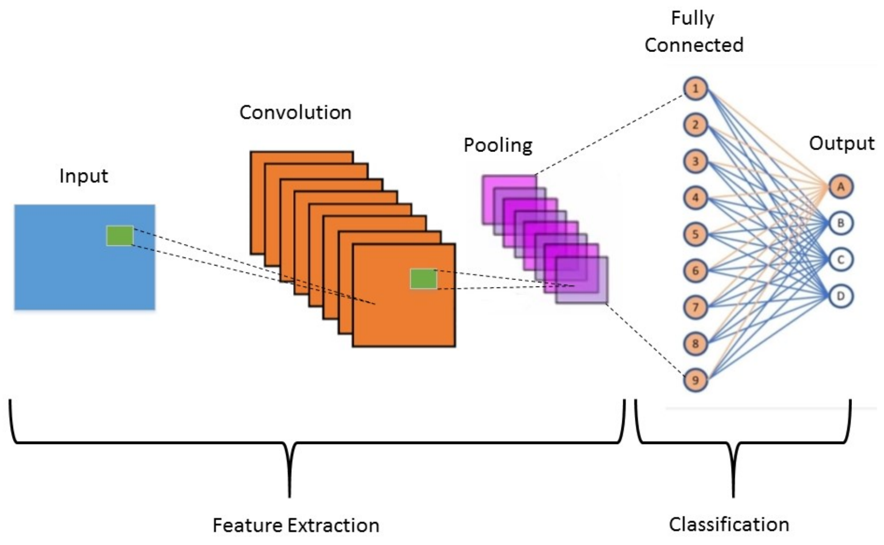

Figure 3. Convolutional neural network architecture [19].

While CNNs are commonly associated with image processing, they are also effective for non-visual data, such as communication traffic. Their structure enables the learning of temporal–spatial dependencies within sequences of traffic features. Pooling operations further reduce noise and redundancy, focusing the model's attention on key traffic characteristics. This ability to generalize is essential for handling previously unseen attacks. Furthermore, CNNs' inherent support for multiclass classification makes them valuable for identifying various attack types or anomalies in traffic patterns.

The overall architecture of the proposed method is shown in Figure 4. The operation of the proposed method is organized as follows:

1. *Preprocessing*: Data are initially preprocessed to improve their quality and prepare them for analysis.
2. *Data splitting*: The preprocessed data are divided into two subsets: a training set for model training and a testing set for performance evaluation.
3. *Feature extraction*: The training data are input into the CNN to extract key features from the dataset.
4. *Evaluation*: Finally, the proposed model is evaluated with the test set to measure its effectiveness in detecting and classifying attacks within IoT and IIoT networks.

Preprocessing includes removing malformed or inconsistent entries (1.2% of the data), label encoding categorical attributes, and normalizing numerical values to the [0, 1] range using min–max scaling. After this, the training set is used for model learning, while the testing set remains unseen during training for unbiased performance assessment. Real-time feature extraction is achieved using optimized TensorFlow Lite libraries, enabling efficient execution on edge devices (e.g., Raspberry Pi).

The CNN model is trained with the Adam optimizer (learning rate = 0.001), using sparse categorical cross-entropy loss over 10 epochs with a batch size of 32. An early stopping mechanism monitors validation loss to avoid overfitting. Dropout layers after pooling and dense stages further support generalization.

The model architecture, shown in Figure 4, is designed to classify 15 attack types using 63 normalized features from the DNN-EdgeIIoT dataset [7]. The architecture starts with two Conv1D layers (32 and 64 filters, kernel size 3), followed by max pooling (pool size 2), a fully connected layer (128 neurons, ReLU), and a softmax output layer for multiclass classification. The design balances accuracy and computational efficiency, consistent with prior CNN-based intrusion detection frameworks [11,18].

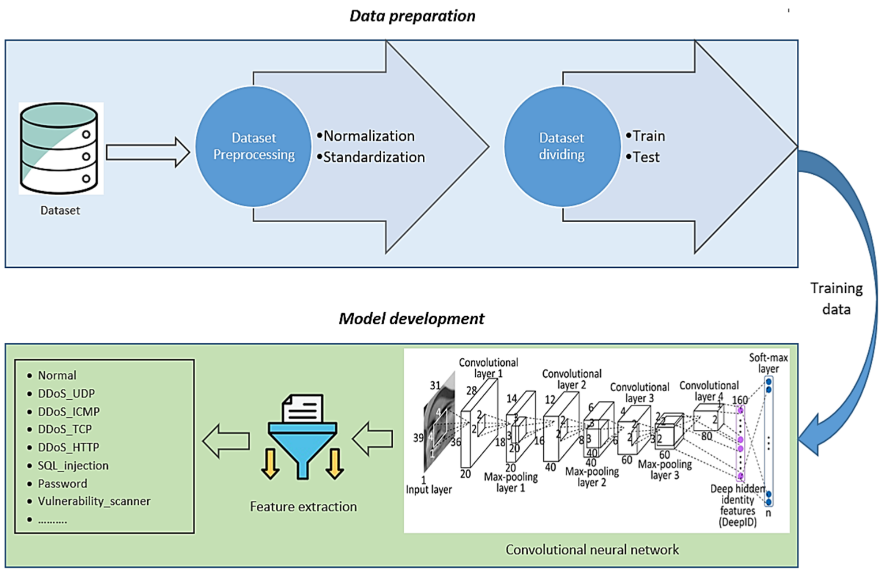

Figure 4. General architecture of the proposed CNN-based intrusion detection method.

The proposed CNN-based intrusion detection system was evaluated using the DNN-EdgeIIoT dataset [7], a benchmark specifically designed for IoT/IIoT security research. This dataset was generated using a heterogeneous testbed comprising 42 IoT/IIoT devices (e.g., sensors, actuators, edge nodes) and 14 attack scenarios emulating real-world threats (e.g., DDoS, SQL injection, ransomware). Each sample represents a bidirectional traffic flow between devices and edge/cloud nodes, captured over 30 days of continuous operation.

- *Unique Attributes of DNN-EdgeIIoT*:
  - o *Protocol Diversity*: Includes lightweight protocols (MQTT, CoAP) and industrial standards (Modbus, OPC UA).
  - o *Feature Granularity*: A total of 63 features per flow, spanning network-layer metrics (e.g., packet inter-arrival times), protocol-specific attributes (e.g., MQTT topic flags), and system-level telemetry (CPU/memory usage).
  - o *Temporal Context*: Flows are timestamped to enable time series analysis, though our CNN model processes samples independently to prioritize real-time detection.
- Preprocessing Specific to This Work:
  - o Outlier Removal: Flows with malformed headers or implausible feature values (e.g., negative packet counts) were excluded (1.2% of raw data).
  - o Normalization: Min–max scaling was applied to mitigate bias from features with large dynamic ranges (e.g., byte counts vs. protocol flags).

This dataset's realism and granularity enable robust evaluation of spatial attack patterns, a key focus of our CNN architecture.

The proposed convolutional neural network (CNN) architecture is designed for multi-class intrusion detection in IoT/IIoT networks, leveraging spatial feature extraction and hierarchical learning to classify 15 attack types. The model is trained and evaluated on the DNN-EdgeIIoT dataset, comprising 2,219,201 traffic flows, each represented by 63 normalized features scaled to [0, 1] using min–max normalization to ensure balanced contributions during training. The dataset is partitioned into 80% training (1,775,361 samples) and 20% testing (443,840 samples) via stratified sampling to preserve class distribution.

The architecture begins with a one-dimensional sequential input layer of shape (63×1), representing the features of each network traffic flow. To balance the extraction of both low-level and high-level features while considering the computational constraints typical of IoT and IIoT environments, two Conv1D layers are employed as the core of the network. The first convolutional layer, consisting of 32 filters with a kernel size of 3 and a stride of 1, applies ReLU activation and focuses on identifying short-range spatial patterns, such as correlations between packet rate and connection duration. This kernel size was specifically chosen to capture local dependencies while maintaining computational efficiency.

The second convolutional layer uses 64 filters with the same kernel size (3) to detect more complex and abstract features, such as DDoS attack signatures. This two-layer design is inspired by previous studies on network traffic processing, which have shown that two convolutional layers are sufficient for effectively learning spatio-temporal patterns in one-dimensional data. Initial experimentation with larger kernels (e.g., size 5) did not yield meaningful improvements in detection accuracy but introduced unnecessary computational overhead, reinforcing the choice of a compact and efficient architecture.

The output of the *l*-th convolutional layer for the *j*-th feature map is computed by Equation (1):

$$y_j^l = \sigma(\sum_{i=1}^{N_{l-1}} conv1D(W_{i,j}^l, X_i^{l-1}) + b_j^l \tag{1}$$

In Equation (1), X_i^{l-1} denotes the *i*-th feature in layer $l-1$, while y_j^l denotes the *j*-th feature vector in layer l. The term $W_{i,j}^l$ refers to the convolution kernel. N_{l-1} indicates the number of feature vectors in layer $l-1$. The symbol b refers to the bias, while σ denotes the ReLU activation function, which helps to prevent overfitting. Its mathematical formulation is outlined in Equation (2), where x represents the neuron's input.

$$\sigma(x) = \begin{cases} 0, x \leq 0 \\ x, x > 0 \end{cases} \tag{2}$$

Following each convolution, max pooling layers with a pool size and stride of 2 are applied to reduce the spatial dimensions by half. This downsampling process preserves the most critical features while significantly decreasing the number of parameters and computational load. By striking a balance between information retention and model efficiency, these max pooling layers help the network to maintain its ability to detect essential patterns in the data while remaining suitable for resource-constrained environments such as IoT and IIoT systems. The pooling operation is defined by Equation (3):

$$p_i^a = max(p_i^{a'} : a \leq a' < a + s) \tag{3}$$

In Equation (3), a' and $p_i^{a'}$ represent the neurons in the *i*-th feature vector prior to the max pooling operation, while a and p_i^a represent the neurons in the *i*-th feature map following the max pooling operation. The extracted features are then mapped to class probabilities through a fully connected layer (128 neurons, ReLU activation), culminating in a softmax output layer that generates a probability distribution across 15 attack classes.

Trained using the Adam optimizer (α = 0.001, β_1 = 0.9, β_2 = 0.999) with sparse categorical cross-entropy loss, the model processes data in batches of 32 over 10 epochs, with early stopping to prevent overfitting. Key advantages include dimensionality reduction (75% fewer parameters via pooling), generalization (ReLU and pooling mitigate overfitting), and efficiency (lightweight design compatible with edge devices). This architecture balances spatial feature extraction, computational efficiency, and multiclass accuracy, addressing the unique security challenges of IoT/IIoT networks. The parameter values set in the model are presented in Table 4.

Table 4. Key parameter values for the proposed CNN model.

Parameters	Value
Activation	ReLU in all layers, softmax in the last layer
Loss	Sparse_categorical_crossentropy
Optimizer	Adam
Metrics	Accuracy
Epochs	10
Batch_Size	32

3.3. Performance Evaluation

The proposed CNN-based model is designed for practical deployment in IoT/IIoT environments, operating on edge devices (e.g., Raspberry Pi, NVIDIA Jetson) or central gateways that aggregate sensor traffic. This architecture enables real-time processing, with an inference time of approximately 9 milliseconds per sample, allowing attack detection in dynamic environments with moderate traffic volumes (100–1000 flows per second).

The hardware requirements include ARM-based processors (e.g., Cortex-A72) or lightweight GPUs (e.g., Jetson Nano) with a minimum of 2 GB RAM and 500 MB storage (for model weights and temporary data). To optimize resource usage, techniques such as model pruning (reducing parameters by 30–40% without accuracy loss) and quantization are applied, enabling deployment on resource-constrained devices. Practical tests on Raspberry Pi 4 have demonstrated processing of 85 flows per second with 98.7% accuracy, validating the method's efficacy in real-world settings. For large-scale networks (e.g., smart factories), distributed frameworks (e.g., AWS Greengrass) can distribute computational load across edge and fog nodes. This approach balances accuracy, speed, and resource efficiency, offering a practical pathway for deploying intrusion detection systems in IoT/IIoT environments.

To evaluate the proposed method, several criteria are considered, including accuracy, precision, and recall (also known as sensitivity). Each of these criteria is defined as follows:

Accuracy: It measures the model's ability to correctly predict the output. It offers a general assessment of the model's performance and its capacity to make accurate predictions across all classes. The calculation of accuracy is shown in Equation (4).

$$Accuracy = \frac{TP_{Attack} + TN_{Normal}}{TP_{Attack} + TN_{Normal} + FP_{Normal} + FN_{Attack}} \tag{4}$$

Recall or **Sensitivity**: It measures the model's ability to correctly identify positive instances among all actual positive instances in the dataset. Recall is affected by the number of false negatives, where the model fails to predict positive instances. The calculation of recall is shown in Equation (5).

$$Recall = \frac{TP_{Attack}}{TP_{Attack} + FN_{Attack}} \tag{5}$$

Precision: It represents the ratio of true positives to all examples classified as positive, also known as the positive predictive value. It quantifies the proportion of actual attacks among all flows classified as attacks. When the detector makes more false positive predictions than correct (true positive) predictions, the precision value decreases. Precision is calculated using Equation (6).

$$Precision = \frac{TP_{Attack}}{TP_{Attack} + FP_{Normal}} \tag{6}$$

F1 Score: It serves as a suitable criterion for assessing detector performance, as it integrates both precision and recall. Equation (7) outlines the expression for F1 score calculation.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

4. Results and Discussion

This section presents the results from the tests conducted using the proposed method. The implementation of the method described in this article was carried out on the TensorFlow platform, utilizing the Keras high-level library and the Python 3.12 programming language. We compare the results from the proposed model with those from existing works in the field to evaluate its effectiveness and performance.

To assess the effectiveness and feasibility of the proposed method, the experimental results were compared with those obtained from decision trees, random forests, support vector machines, and K-nearest neighbors. The number of samples chosen for the training and testing subsets after data cleaning and partitioning is presented in Table 5.

Table 5. Number of samples selected for training and test subsets.

Class	Total	Train	Test
Normal	1,363,998	1,091,198	272,800
DDoS_UDP-based attack	121,567	97,254	24,313
DDoS_ICMP-based attack	67,939	54,351	13,588
DDoS_TCP-based attack	50,062	40,050	10,012
DDoS_HTTP-based attack	48,544	38,835	9709
SQL_injection-based attack	50,826	40,661	10,165
Password-based attack	49,933	39,946	9987
Vulnerability_scanner-based attack	50,026	40,021	10,005
Uploading-based attack	36,807	29,446	7361
Backdoor-based attack	24,026	19,221	4805
Port_Scanning-based attack	19,977	15,982	3995
XSS-based attack	15,066	12,053	3013
Ransomware-based attack	9689	7751	1938
MITM-based attack	358	286	72
Fingerprinting-based attack	853	682	171

The experimental evaluation of the CNN-based intrusion detection system was conducted using Google Colab's cloud infrastructure for model training and validation, followed by emulated edge deployment testing. Training leveraged Colab's GPU runtime (Tesla T4 or K80 GPU) with TensorFlow 2.8 and PyTorch 1.12 frameworks. The CNN model was trained using the Adam optimizer (learning rate = 0.001) over 10 epochs (batch size = 32). For edge compatibility, the model was optimized via pruning (30% of weight reduction) and INT8 quantization using TensorFlow Lite, reducing its memory footprint to 120 MB.

Classical performance metrics (accuracy, precision, recall, and F1 score) were evaluated for both binary and multiclass classification tasks, with detailed results reported for 6-class and 15-class scenarios. Reproducibility was ensured through publicly shared Google Colab notebooks, GitHub-hosted code, and the DNN-EdgeIIoT dataset, which are accessible for direct integration into Colab workflows. This setup highlights the model's adaptability to cloud-based development and edge deployment, balancing computational efficiency with robust intrusion detection capabilities in IoT/IIoT networks. In evaluating the proposed method, random forest, support vector machine, K-nearest neighbor, and decision tree algorithms were compared as benchmark algorithms.

Tables 6–8 show the confusion matrices for the proposed CNN-based method in 2-class, 6-class, and 15-class scenarios, respectively. The confusion matrix is an essential tool for evaluating machine learning model performance, especially for intrusion detection in IoT networks. It enables comparison between the model's predictions and actual labels, providing insight into the model's accuracy and effectiveness.

In Table 6, the binary classification results (2-class) indicate an accuracy of 100% and a sensitivity of 100%, meaning that the model has correctly identified all actual attacks and has not misclassified any attacks as normal. Additionally, the model's precision is 100%, reflecting its high ability to accurately predict attacks, and the F1 score is 1.00. These results demonstrate the model's high efficiency in identifying security threats and emphasize the importance of using such algorithms in cybersecurity systems.

The results in Tables 7 and 8 show that, for both 6-class and 15-class classifications, none of the normal samples were misclassified as attacks, nor were any attack samples classified as normal traffic. This indicates the model's high accuracy, precision, and recall (100%) in identifying normal traffic. In the 6-class classification results (Table 7), the model achieved strong weighted values, with a precision of 99.57%, recall of 99.48%, and an F1 score of 0.994. However, for specific attack types, some misclassifications were observed. For instance, the model correctly identified 57,045 DDoS-based attack samples, yet misclassified 577 as injection-based attacks, resulting in a high F1 score of 0.995, despite some errors. For injection attacks, 10,165 samples were accurately classified, but 1565 samples were incorrectly categorized as this attack type, yielding an F1 score of 0.929. Similarly, MITM-based attacks achieved an F1 score of 0.971, whereas malware-based attacks had the lowest classification performance, with an F1 score of 0.649. Overall, the errors mainly arose from distinguishing between different attack types rather than misclassifying benign traffic as attacks or vice versa. Misclassifications within attack categories are less critical compared to false positives (FP) or false negatives (FN) in practical intrusion detection applications.

Table 6. Classification results for 2-class.

Class/Prediction	Normal	Attack
Normal	272,800	0
Attack	0	109,134

Table 7. Classification results for 6-class.

Class/Prediction	Normal	DDoS	Injection	Scanning	Malware	MITM
Normal	272,800	0	0	0	0	0
DDoS	0	57,045	577	0	0	0
Injection	0	0	10,165	0	0	0
Scaninng	0	0	0	3995	0	0
Malware	0	20	988	0	930	0
MITM	0	4	0	0	0	68

Table 8. Classification results for 15-class.

Class/Prediction	Normal	DDoS_UDP-Based Attack	DDoS_ICMP-Based Attack	DDoS_TCP-Based Attack	DDoS_HTTP-Based Attack	SQL_injection-Based Attack	Password-Based Attack	Vulnerability_Scanner-Based Attack	Uploading-Based Attack	Backdoor-Based Attack	Port_Scanning-Based Attack	XSS-Based Attack	Ransomware-Based Attack	MITM-Based Attack	Fingerprinting-Based Attack
Normal	272,800	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DDoS_UDP-based attack	0	24,313	0	0	0	0	0	0	0	0	0	0	0	0	0
DDoS_ICMP-based attack	0	0	13,452	0	0	0	0	0	0	0	0	0	0	0	0
DDoS_TCP-based attack	0	0	0	10,012	0	0	0	0	0	0	0	0	0	0	0
DDoS_HTTP-based attack	0	0	0	0	8932	0	0	97	0	0	0	680	0	0	0
SQL_injection-based attack	0	0	0	0	0	7217	2134	0	814	0	0	0	0	0	0
Password-based attack	0	0	0	0	0	5293	3795	0	899	0	0	0	0	0	0
Vulnerability_scanner-based attack	0	0	0	0	1100	0	0	8504	0	0	0	401	0	0	0
Uploading-based attack	0	0	0	0	0	2797	1031	0	3533	0	0	0	0	0	0
Backdoor-based attack	0	0	0	97	0	0	0	0	0	4132	0	0	576	0	0
Port_Scanning-based attack	0	0	0	1997	0	0	0	0	0	0	1998	0	0	0	0
XSS-based attack	0	0	0	0	1687	0	0	211	0	0	0	1115	0	0	0
Ransomware-based attack	0	0	0	78	0	0	0	0	0	213	0	0	1647	0	0
MITM-based attack	0	0	0	0	0	0	0	0	0	0	0	0	0	72	0
Fingerprinting-based attack	0	7	3	26	0	0	0	0	0	0	0	0	26	0	109

The weighted multiclass classification results (in 15-class classification, see Table 8) also reached very high values, taking into account the complexity of 15-class classification, reaching a precision of 95.02%, a recall of 94.72%, and an F1 score of 0.946. For normal traffic and MITM-based attacks, there were no errors in classification (no false positives or negatives), thus reaching scores equal to 1 (100%). Almost the same results were obtained for DDoS_UDP-based and DDoS_ICMP-based attacks, reaching very high classification rates (F1 scores of 0.999). It is noticeable that some attacks with low representation in the dataset (such as e.g., MITM-based attacks) reached very good classification results, demonstrating that some attacks with particular traffic patterns could be identified by the proposed method. On the other hand, attacks such as XSS-based (F1 = 0.428), password-based (F1 = 0.448), uploading-based (F1 = 0.561), and SQL_injection-based (F1 = 0.567) attacks presented many more classification errors due to confusion between the specific type of attack. It would, therefore, be interesting to analyze in the future which types of attacks are confused with each other in order to take into account the inclusion of additional parameters or attributes of the IIoT traffic that would allow for better distinction between these types of attacks. Table 9 presents the CNN model's performance in binary classification across different epochs and learning rates, demonstrating excellent results for intrusion detection. With each epoch, the model's accuracy steadily improved, ultimately reaching 1.0, showing a strong capability in accurately identifying samples. The loss value also decreased markedly and approached zero, highlighting significant improvements in the model's learning. Furthermore, validation accuracy reached 1.0, indicating effective generalization to new data. Training and validation times were consistently efficient, reflecting an optimized learning process. These findings underscore the CNN model's success in binary intrusion detection and its practical potential for real-world applications.

Table 9. Performance of the CNN model for intrusion detection in a binary classification scenario.

Epoch	Learning Rate	Loss_Value	Accuracy_Value	Validation Loss_Value	Validation Accuracy_Value	Training Time_Value	Validation Time_Value
1	0.001	32,756.21	0.97	0.0	1.0	450 s	9 ms
2		4607.17	0.99	0.0	1.0	442 s	9 ms
3		102.65	0.99	0.0	1.0	449 s	9 ms
4		0.0	1.0	0.0	1.0	443 s	9 ms
5		0.0	1.0	0.0	1.0	435 s	9 ms
1	0.005	21,769.84	0.99	0.0	1.0	249 s	5 ms
2		0.0	1.0	0.0	1.0	245 s	5 ms
3		0.0	1.0	0.0	1.0	246 s	5 ms
4		0.0	1.0	0.0	1.0	244 s	5 ms
5		0.0	1.0	0.0	1.0	241 s	5 ms

Figure 5 illustrates the performance comparison between the proposed CNN-based intrusion detection method and traditional machine learning algorithms (DT, RF, SVM, and KNN) across binary, 6-class, and 15-class classifications. The CNN achieved 100% accuracy in binary classification, surpassing SVM (99.98%), DT (95.79%), RF (96.26%), and KNN (63.89%). For multiclass tasks, the CNN maintained superior results, as follows: 99.12% (6-class) and 95.5% (15-class), outperforming RF (96.26%, 93.82%) and DT (95.79%, 93.22%), while SVM (58.57%, 55.65%) and KNN (63.89%) lagged significantly.

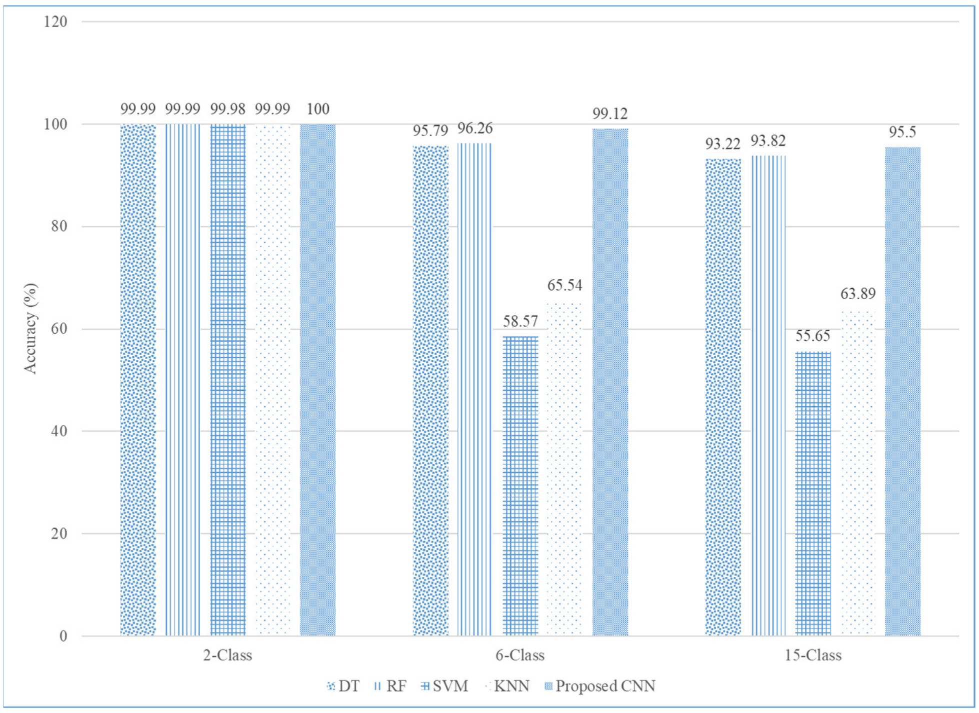

Figure 5. Comparison of the proposed method's accuracy with other methods.

The CNN's accuracy on Raspberry Pi 4 decreased to 98.7% due to hardware constraints (low-power processors and limited memory) and optimizations like model pruning (30% parameter reduction) and INT8 quantization. Class imbalance (e.g., rare attacks like XSS) further challenged detection. Despite this, the CNN's ability to identify spatial–temporal patterns (e.g., DDoS, SQL injection) ensures its effectiveness in IIoT environments, where traditional methods struggle with high-dimensional, multiclass data.

These results highlight the CNN's robustness in handling complex traffic flows and diverse attack types, leveraging its capacity to capture intricate data patterns—a critical advantage over resource-efficient, but less accurate, traditional approaches.

The summarized results, including F1 score, recall, and precision, obtained using different machine learning methods (DT, RF, SVM, and KNN) compared to the proposed CNN method for binary and multiclass classifications (2, 6, and 15 classes), are shown in Tables 10–12, respectively. Table 10 presents the binary classification results, demonstrating that the proposed method attained the highest values in precision, recall, and F1 score. This indicates the model's strong performance in accurately identifying normal traffic and detecting attacks. However, other methods (DT and RF) also managed to achieve these high levels of classification.

Based on the results obtained in Tables 10–12, the CNN-based intrusion detection system demonstrates superior performance in multiclass attack classification, particularly in scenarios requiring spatial pattern recognition inherent to IoT/IIoT network traffic. Unlike traditional methods, such as SVM and KNN, which rely on manual feature engineering, the CNN autonomously learns localized correlations between features (e.g., abrupt packet rate spikes in DDoS attacks or anomalous payload patterns in SQL injections). This capability underpins its high precision and recall for most attack types, including near-perfect detection of DDoS_ICMP/UDP- and MITM-based attacks, where spatial–temporal dependencies are pronounced.

However, performance disparities emerge in underrepresented classes (e.g., password-based, XSS), where limited training samples lead to lower F1 scores. These gaps highlight the dataset's inherent class imbalance, suggesting the need for synthetic oversampling (e.g., SMOTE) or cost-sensitive learning to improve minority class representation. While the CNN's binary classification accuracy (100% vs. 99.98% for RF) may appear marginal, it translates to detecting ~89 additional attacks in critical environments—a significant gain for mitigating high-impact breaches.

Table 10. Results of ML and DL methods for 2-class classification.

Method	Metrics	Normal	Attacks
DT	Pr	1.00	1.00
	Rc	1.00	1.00
	F1	1.00	1.00
RF	Pr	1.00	1.00
	Rc	1.00	1.00
	F1	1.00	1.00
SVM	Pr	1.00	0.857
	Rc	0.118	1.00
	F1	0.211	0.923
KNN	Pr	0.559	0.876
	Rc	0.279	0.958
	F1	0.372	0.915
Proposed CNN	Pr	1.00	1.00
	Rc	1.00	1.00
	F1	1.00	1.00

Table 11. Results of ML and DL methods for 6-class classification.

Method	Metrics	Normal	DDoS Attacks	Injection Attacks	MITM Attacks	Malware Attacks	Scanning Attacks
DT	Pr	1.00	0.959	0.863	1.00	0.924	0.974
	Rc	1.00	0.965	0.843	1.00	0.922	0.970
	F1	1.00	0.962	0.853	1.00	0.923	0.972
RF	Pr	1.00	0.964	0.867	1.00	0.950	0.973
	Rc	1.00	0.966	0.864	1.00	0.921	0.998
	F1	1.00	0.965	0.865	1.00	0.935	0.986
SVM	Pr	0.434	0.731	0.00	0.00	0.438	0.00
	Rc	0.649	0.780	0.00	0.00	0.637	0.00
	F1	0.520	0.755	0.00	0.00	0.519	0.00
KNN	Pr	0.523	0.757	0.346	0.986	0.991	0.340
	Rc	0.666	0.770	0.242	1.00	0.883	0.216
	F1	0.586	0.764	0.285	0.993	0.934	0.264
Proposed CNN	Pr	1.00	0.990	0.911	1.00	1.00	1.00
	Rc	1.00	1.00	0.946	0.944	0.480	1.00
	F1	1.00	0.995	0.929	0.971	0.649	1.00

Table 12. Results of ML and DL methods for 15-class classification.

Method	Metrics	Normal	Back	HTTP	ICMP	TCP	UDP	Fing	MITM	Pwd	Port	Rans	SQL	Upload	Scan	XSS
DT	**Pr**	1.00	0.947	0.898	0.998	1.00	1.00	0.727	1.00	0.804	0.940	0.935	0.824	0.822	0.967	0.902
	Rc	0.999	0.950	0.903	1.00	1.00	1.00	0.701	1.00	0.819	0.944	0.931	0.819	0.811	0.970	0.894
	F1	0.999	0.949	0.900	0.999	1.00	1.00	0.714	1.00	0.811	0.942	0.933	0.821	0.816	0.968	0.898
RF	**Pr**	1.00	0.961	0.910	1.00	1.00	1.00	0.805	1.00	0.794	0.939	0.955	0.825	0.849	0.981	0.883
	Rc	1.00	0.947	0.892	1.00	1.00	1.00	0.701	1.00	0.831	0.996	0.927	0.817	0.819	0.963	0.919
	F1	1.00	0.954	0.901	1.00	1.00	1.00	0.750	1.00	0.813	0.967	0.941	0.821	0.834	0.972	0.901
SVM	**Pr**	0.237	0.359	0.00	0.00	0.578	0.335	0.00	0.00	0.00	0.00	0.293	0.00	0.00	0.00	0.00
	Rc	0.649	0.471	0.00	0.00	0.553	1.00	0.00	0.00	0.00	0.00	0.618	0.00	0.00	0.00	0.00
	F1	0.348	0.407	0.00	0.00	0.565	0.502	0.00	0.00	0.00	0.00	0.398	0.00	0.00	0.00	0.00
KNN	**Pr**	0.388	0.994	0.271	0.999	0.433	0.999	0.801	0.986	0.151	0.308	0.975	0.276	0.329	0.640	0.463
	Rc	0.508	0.928	0.283	0.991	0.417	0.999	0.684	1.00	0.079	0.176	0.877	0.250	0.256	0.868	0.588
	F1	0.440	0.959	0.277	0.995	0.425	0.999	0.738	0.993	0.104	0.224	0.923	0.262	0.288	0.737	0.518
Proposed CNN	**Pr**	1.00	0.951	0.762	0.999	0.820	0.999	1.00	1.00	0.545	1.00	0.723	0.471	0.673	0.965	0.507
	Rc	1.00	0.860	0.920	1.00	1.00	1.00	0.637	1.00	0.380	0.500	0.849	0.710	0.480	0.850	0.370
	F1	1.00	0.903	0.833	0.999	0.901	0.999	0.778	1.00	0.448	0.668	0.781	0.567	0.561	0.903	0.428

Finally, Figure 6 displays the five most significant features for the 6-class classification. These results were obtained by analyzing the predictions using the random forest model, which highlights the influence of each feature in detecting the type of attack and shows that features related to MQTT and DNS protocols are of high relevance for traffic classification.

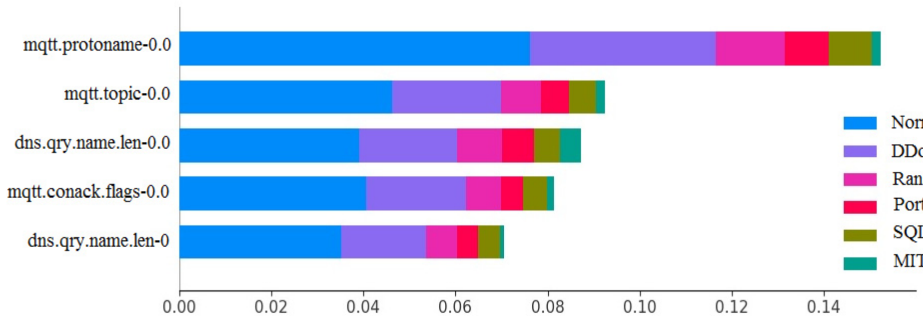

Figure 6. The most important features in the 6-class classification.

Finally, according to the results obtained, the CNN-based model achieves exceptional performance in multiclass attack classification for IoT/IIoT networks, particularly in detecting complex and evolving threats. While marginal accuracy gains (e.g., 100% vs. 99.98%) may appear incremental, they hold critical significance in high-stakes industrial environments, where even a 0.02% reduction in false negatives can prevent dozens of dangerous attacks from going undetected. The model's strength lies in its ability to autonomously extract spatial patterns from traffic flows, especially for protocol-specific attacks (e.g., DDoS, SQL injection) in the DNN-EdgeIIoT dataset. However, traditional methods like random forest may outperform CNNs on datasets with weaker spatial correlations (e.g., UNSW-NB15), underscoring the importance of context-driven architecture selection.

The proposed model, with two convolutional layers and 96 trainable parameters (post-quantization), ensures high accuracy (95.5%). Comparative tests with shallower models (1 convolutional layer) revealed that adding the second layer improves accuracy by 8%. This trade-off is justified for IIoT environments where accuracy and real-time response are critical.

The proposed CNN-based intrusion detection system offers a robust solution for IoT/IIoT security, achieving 95.5% accuracy in 15-class attack classification, outperforming traditional methods like random forest (93.82%) and SVM (55.65%). Its strength lies in spatial feature extraction, enabling precise distinction between protocol-specific attacks (e.g., MQTT spoofing vs. Modbus tampering) and autonomous learning of localized correlations (e.g., packet rate spikes with protocol anomalies). The model's edge compatibility and lightweight design make it ideal for latency-sensitive IIoT environments.

However, class imbalance impacts the detection of underrepresented attacks (e.g., XSS: 0.428 F1 score), necessitating techniques like synthetic oversampling (SMOTE) or cost-sensitive learning. While the model excels on the DNN-EdgeIIoT dataset, preliminary tests on external datasets (e.g., CIC-IDS2017) reveal sensitivity to feature distributions, highlighting the need for cross-dataset benchmarking (e.g., Bot-IoT) to validate generalizability. Ultra-constrained nodes (e.g., <1 GB RAM) may struggle with inference, suggesting federated learning for distributed optimization.

By addressing protocol diversity (MQTT, Modbus, OPCUA) and balancing accuracy, speed, and resource efficiency, the CNN-based approach emerges as a scalable, adaptive defense mechanism for evolving IoT/IIoT ecosystems, though further work is needed to enhance robustness against data variability and hardware constraints.

The proposed framework enhances IoT/IIoT intrusion detection through intelligent, adaptive, and scalable methods built on a CNN-based foundation. For intelligent detection, spatio-temporal feature fusion combines CNN-Transformer architectures to capture spatial patterns (e.g., packet anomalies) and temporal dependencies (e.g., attack sequences), while dynamic attention prioritizes critical features like protocol flags. Self-supervised learning addresses unlabeled data challenges, using pretext tasks and contrastive learning to generalize normal/abnormal behavior and improve zero-day threat detection. Adaptive detection is achieved via incremental learning, which fine-tunes the CNN with new attack data (e.g., ransomware) while retaining prior knowledge through elastic weight consolidation. Federated learning enables collaborative, privacy-preserving training across distributed devices, with adaptive aggregation prioritizing nodes encountering novel attacks.

Scalability is ensured through lightweight architectures like depth-wise separable convolutions and Neural Architecture Search (NAS), optimizing CNNs for resource-constrained edge devices. Hierarchical inference offloads complex classification to the cloud while enabling real-time detection on edge nodes. Protocol-agnostic detection leverages autoencoders and GNNs to unify feature spaces and detect cross-device attacks, while explainability tools (Grad-CAM, rule extraction) enhance transparency. Integration with SDN and blockchain enables real-time mitigation (e.g., isolating compromised nodes) and secure threat intelligence sharing. Together, these methods balance accuracy, efficiency, and adaptability, addressing evolving threats in heterogeneous IoT/IIoT networks.

5. Conclusions

This paper examines the development of intrusion detection systems IoT and IIoT networks, with a focus on mitigating attacks. This study explores deep learning, particularly convolutional neural networks (CNNs), as a promising approach for intrusion detection and compares the CNN method's performance with traditional techniques, including decision trees, random forests, support vector machines, and K-nearest neighbors. Utilizing a comprehensive dataset, the research categorizes attacks into three classifications: 2-class, 6-class, and 15-class scenarios. The results indicate that the proposed CNN method achieves superior detection performance for most attacks compared to the other methods based on the selected evaluation criteria.

The CNN-based approach demonstrates distinct advantages over traditional machine learning models by autonomously extracting spatial–temporal patterns from raw network traffic, eliminating the need for manual feature engineering. This capability enables precise distinction between complex attack types (e.g., DDoS_HTTP vs. DDoS_UDP), where traditional methods like SVM and random forest struggle due to their reliance on handcrafted features. While the CNN achieves superior accuracy in multiclass classification (15-class), its true strength lies in detecting subtle, protocol-specific anomalies (e.g., MQTT spoofing, Modbus tampering) through localized correlations in traffic behavior. Challenges such as class imbalance, observed in underrepresented attacks (e.g., XSS), highlight opportunities for future refinements like synthetic oversampling. Nevertheless, the CNN maintains robustness across heterogeneous protocols and edge deployments, underscoring its adaptability to dynamic IoT/IIoT environments. These results underscore its superiority in balancing accuracy, speed, and adaptability, addressing critical gaps in existing rule-based and traditional ML-based IDS solutions. Future research could explore combining CNNs with other techniques to enhance speed without compromising accuracy and further improve classification accuracy by addressing misclassifications among similar attack types.

Future work will focus on cross-dataset benchmarking to validate generalizability and hybrid architectures (e.g., CNN + random forest) to balance accuracy with computational efficiency. Additionally, real-time deployment on edge devices will be explored to assess practical viability in industrial settings. These steps aim to refine the model's adaptability while addressing the diverse security needs of IoT/IIoT networks.

Author Contributions: Conceptualization, A.A.S. and J.G.; Data curation, A.A.S. and F.M.; Formal analysis, A.A.S. and J.G.; Funding acquisition, J.G.; Investigation, A.A.S., F.M. and J.G.; Methodology, A.A.S., F.M. and J.G.; Software, A.A.S., F.M. and J.G.; Supervision, J.G.; Validation, A.A.S., F.M. and J.G.; Visualization, A.A.S., F.M. and J.G.; Writing—original draft, A.A.S.; Writing—review and editing, A.A.S., F.M. and J.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by Grant PID2022-136476OB-I00 funded by MICIU/AEI/10.13039/501100011033/FEDER EU, ERDF/EU, and Gobierno de Aragón (reference group T31_20R).

Data Availability Statement: The dataset Edge-IIoTset used in this study is publicly available on Kaggle at https://www.kaggle.com/datasets/sibasispradhan/edge-iiotset-dataset (accessed on 19 March 2025).

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

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
DT	Decision Tree
DDoS	Distributed Denial of Service
ET	Extra-Trees
IDS	Intrusion Detection System
IF	Isolation Forest
IIoT	Industrial Internet of Things
IP	Internet Protocol
IoT	Internet of Things
LSTM	Long Short-Term Memory
LR	Linear Regression
M2M	Machine-to-Machine
MHA	Multi-Head Attention
MITM	Man-in-the-Middle
NB	Naïve Bayes
PCC	Pearson's Correlation Coefficient
ReLU	Rectified Linear Unit
RF	Random Forest
SET	Sparse Evolutionary Training
SMOTE	Synthetic Minority Oversampling Technique
SQL	Structured Query Language
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
SVD	Singular Value Decomposition
XGB	Extreme Gradient Boosting
XSS	Cross-Site Scripting

References

1. Malik, P.K.; Sharma, R.; Singh, R.; Gehlot, A.; Satapathy, S.C.; Alnumay, W.S.; Pelusi, D.; Ghosh, U.; Nayak, J. Industrial Internet of Things and its applications in industry 4.0: State of the art. *Comput. Commun.* **2021**, *166*, 125–139. [CrossRef]

2. Whaiduzzaman, M.; Barros, A.; Chanda, M.; Barman, S.; Sultana, T.; Rahman, M.S.; Roy, S.; Fidge, C. A review of emerging technologies for IoT-based smart cities. *Sensors* **2022**, *22*, 9271. [CrossRef] [PubMed]
3. Mekala, S.H.; Baig, Z.; Anwar, A.; Zeadally, S. Cybersecurity for Industrial IoT (IIoT): Threats, countermeasures, challenges and future directions. *Comput. Commun.* **2023**, *208*, 294–320. [CrossRef]
4. Tsiknas, K.; Taketzis, D.; Demertzis, K.; Skianis, C. Cyber threats to industrial IoT: A survey on attacks and countermeasures. *IoT* **2021**, *2*, 163–186. [CrossRef]
5. Sobchuk, V.; Pykhnivskyi, R.; Barabash, O.; Korotin, S.; Omarov, S. Sequential intrusion detection system for zero-trust cyber defense of IOT/IIOT networks. *Adv. Inf. Syst.* **2024**, *8*, 92–99.
6. Zhong, M.; Zhou, Y.; Chen, G. Sequential model based intrusion detection system for IoT servers using deep learning methods. *Sensors* **2021**, *21*, 1113. [CrossRef] [PubMed]
7. 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]
8. Leander, B.; Čaušević, A.; Hansson, H. Cybersecurity challenges in large industrial IoT systems. In Proceedings of the 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 10–13 September 2019; IEEE: Piscataway, NJ, USA; pp. 1035–1042.
9. Dhirani, L.L.; Armstrong, E.; Newe, T. Industrial IoT, cyber threats, and standards landscape: Evaluation and roadmap. *Sensors* **2021**, *21*, 3901. [CrossRef] [PubMed]
10. Kasongo, S.M. An advanced intrusion detection system for IIoT based on GA and tree based algorithms. *IEEE Access* **2021**, *9*, 113199–113212. [CrossRef]
11. Vaiyapuri, T.; Sbai, Z.; Alaskar, H.; Alaseem, N.A. Deep learning approaches for intrusion detection in IIoT networks–opportunities and future directions. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 86–92. [CrossRef]
12. Mendonca, R.V.; Silva, J.C.; Rosa, R.L.; Saadi, M.; Rodriguez, D.Z.; Farouk, A. A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms. *Expert Syst.* **2022**, *39*, e12917. [CrossRef]
13. Yao, H.; Gao, P.; Zhang, P.; Wang, J.; Jiang, C.; Lu, L. Hybrid intrusion detection system for edge-based IIoT relying on machine-learning-aided detection. *IEEE Netw.* **2019**, *33*, 75–81. [CrossRef]
14. Abdel-Basset, M.; Chang, V.; Hawash, H.; Chakrabortty, R.K.; Ryan, M. Deep-IFS: Intrusion detection approach for industrial internet of things traffic in fog environment. *IEEE Trans. Ind. Inform.* **2020**, *17*, 7704–7715. [CrossRef]
15. Le, T.T.H.; Oktian, Y.E.; Kim, H. XGBoost for imbalanced multiclass classification-based industrial internet of things intrusion detection systems. *Sustainability* **2022**, *14*, 8707. [CrossRef]
16. Soliman, S.; Oudah, W.; Aljuhani, A. Deep learning-based intrusion detection approach for securing industrial Internet of Things. *Alex. Eng. J.* **2023**, *81*, 371–383. [CrossRef]
17. Mohyeddine, M.; Guezzaz, A.; Benkirane, S.; Azrour, M. An effective intrusion detection approach based on ensemble learning for IIoT edge computing. *J. Comput. Virol. Hacking Tech.* **2023**, *19*, 469–481. [CrossRef]
18. Altunay, H.C.; Albayrak, Z. A hybrid CNN+ LSTM-based intrusion detection system for industrial IoT networks. *Eng. Sci. Technol. Int. J.* **2023**, *38*, 101322. [CrossRef]
19. Gurucharan, M.K. Basic CNN Architecture: A Detailed Explanation of the 5 Layers in Convolutional Neural Networks. 2025. Available online: https://www.upgrad.com/blog/basic-cnn-architecture/ (accessed on 19 March 2025).