STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–23.
Published online in International Academic Press (www.IAPress.org)

A Hybrid-Based PSO-Optimized CNN-LSTM Model for 5G Networks Intrusion Detection

Marwa O. Al Enany, Ali Bakhiet, Amira M. Gaber*

Higher Institute of Computer Science and Information Systems, 6 October City, Giza, Egypt

Abstract The escalating demand for 5G networks, driven by heightened connectivity and modern application needs, necessitates robust security solutions to counter threats like denial-of-service attacks. This study introduces two optimized CNN-LSTM hybrid models for enhanced intrusion detection in 5G networks, leveraging the 5G-NIDD dataset. By integrating Particle Swarm Optimization (PSO) for hyperparameter tuning, the models achieve superior performance. The first model attains training and validation accuracies of 89.98% and 90.10%, respectively, while the second model excels with a validation accuracy of 99.79% and a minimal loss of 0.0085. These results underscore the models' exceptional accuracy and generalization in classifying malicious network activities, offering a potent solution for 5G security challenges.

Keywords 5G Security, Intrusion Detection, CNN-LSTM, Particle Swarm Optimization, Network Threats, Distributed Denial-of-Service (DDoS).

DOI: 10.19139/soic-2310-5070-4714

1. Introduction

Massive increases in connectivity, traffic density, a variety of use cases, and modern business requirements define the need for 5G. In order to provide security, trust, and privacy, 5G must offer higher performance metrics, including much faster throughput, less latency, a larger connectivity density, and a better mobility range. Furthermore, there is a noticeable increase in public awareness of privacy protection, raising concerns about issues related to user privacy. As more smart fields, like autonomous vehicles, intelligent transportation, smart grid automation, and real-time healthcare applications, adopt 5G technologies, the need to protect user privacy is growing [1]. End-to-end 5G technology enables a fully connected world. It should be noted that the IoT paradigm incorporates heterogeneous technologies that can communicate via various protocols and standards, such as Near-Field Communication (NFC), actuators, Radio Frequency Identification (RFID), Wreless Sensor Network (WSNs), and so forth. These devices typically gather data from smart objects, which have even more constant network connectivity, are easier to locate, and are more susceptible to several types of attacks, including denial-of-service (DoS) attacks. 5G undoubtedly makes it possible to address the present problems with network response times and resource management in the rapidly developing IoT era [2]. Maintaining a high quality of service (QoS) in terms of latency, while carrying massive amounts of data inside the 5G network and maintaining network stability, is a crucial and difficult challenge. Preventing data breaches and the unintentional spread of dangerous content among mobile devices is the ultimate objective for 5G technology to become widely used in the real world [3]. The main fields attracting 5G usability, as introduced by the International Telecommunication Union (ITU) [4], include:

• Ultra-Reliable Low-Latency Communication (URLLC) is essential for achieving high reliability and strict low latency in 5G communication [5], particularly since many of its implementations (e.g., remote surgery, autonomous driving) pose potential threats to human life if compromised. This necessitates robust security and privacy measures, emphasizing ultra-fast integrity protection, encryption, and authentication. Protecting subscriber identities and device identifiers against both passive and active attacks is crucial. Furthermore, due to URLLC's

ISSN 2310-5070 (online) ISSN 2311-004X (print)
Copyright © 202x International Academic Press

stringent performance standards, these 5G services are especially vulnerable to latency-inducing threats, such as signaling storms and radio jamming attacks.

• Massive Machine Type Communications (mMTC) [6] ,where connecting a massive number of devices in cellular networks is an essential challenge. Some other examples of these essential challenges include radio access network (RAN) congestion, substantial signal overhead, controlling very dynamic and irregular MTC traffic, and providing different levels of quality of service (QoS). The rapid growth of interconnected IoT devices and other vertical industries will result in a huge number of wireless sensor networks, posing high requirements on network access quantities and power consumption efficiency.

• Enhanced mobile Broadband (eMBB) [7]. The goal of eMBB communications is to support certain high-priority services while achieving high throughput. Thus, it needs a larger bandwidth and a shorter latency and focuses on improving a user's perceived experience. As a result, the widespread use of broadband data has grown quickly. Furthermore, a significant increase in system bandwidth is required. Since densely populated areas predominantly use this service, The expansion of mobile broadband services (MBBs) and the growing need for mobility among users place certain constraints on the resources available on this backbone network.

There is an urgent need for new security and privacy-focused solutions in 5G networks [8] due to various vulnerabilities and attacks that threaten their stability. Key technologies are at risk, including social engineering, privacy leaks from flawed data, denial-of-service (DoS) attacks that disrupt access to network resources, and the need for radio interface security to safeguard data privacy and integrity during transmission [9]. Many IoT devices are prone to attacks due to inadequate security measures, exacerbated by the complex global 5G supply chain involving multiple vendors. This complexity heightens risks of supply chain violations and security vulnerabilities. Moreover, the rising collection and transmission of personal data raises privacy issues [10]. Securing the 5G virtualized environment is essential, as managing the security of individual network slices poses challenges in establishing various virtual networks on a single physical infrastructure. Network slices' performance and functionalities are significantly impacted by DoS and DDoS attacks . To safeguard against potential disruptions and ensure uninterrupted service delivery, researchers have proposed detection techniques, particularly deep-learning models like SliceSecure[11], to identify DoS/DDoS attacks. Current detection methods primarily use data from simulated 5G networks instead of real 5G slices. Recent studies indicate that these attacks can severely degrade network performance, affecting latency, bandwidth, and potentially compromising system resilience due to vulnerabilities in the 5G architecture, particularly in the RAN control layer [12-13].

1.1. Convolutional neural networks (CNNs)

Convolutional Neural Networks (CNNs) are advanced neural networks in deep learning, crucial for image segmentation, object recognition, and classification [14]. By utilizing convolutional layers for feature extraction and fully connected layers for classification, CNNs effectively capture global and local information, improving the understanding of visual data semantics. Following AlexNet's release, the development of CNNs accelerated [15], showcasing their capability in image identification. AlexNet's eight-layer architecture achieved significant results on the ImageNet dataset. Subsequently, VGGNet optimized design by using small convolutional filters, balancing depth and efficiency. The Inception module further enhanced CNNs by incorporating filters of various sizes, allowing better learning of complex features.The ResNet architecture [16] has made it possible to construct models with more than 100 layers, which effectively addresses the issue of disappearing gradients [17]. The effective feature reuse was made possible by DenseNet [18], which connected every layer to every other layer. Deep learning models excel in image recognition accuracy but require considerable computational resources. Real-world applications aim to balance accuracy with resource constraints, particularly on mobile devices with limited capabilities. This led to the creation of numerous lightweight models, including SqueezeNet [19], ShuffleNet-V2 [20], and MobileNetV2 [21], EfficientNet [22].

1.2. Particle Swarm Optimization (PSO) algorithm

Particle Swarm Optimization (PSO) is a metaheuristic and stochastic optimization algorithm profoundly inspired by the cooperative and collective behavior of natural biological swarms, such as bird flocking and fish schooling [23]. Rather than using gradient descent, the PSO algorithm navigates complex search spaces through the

collaboration of individual entities (referred to as particles). Each particle dynamically adjusts its search trajectory toward an optimal solution based on a mathematical combination of its own best historical performance (cognitive learning) and the overarching best performance achieved by the entire swarm (social learning).

In a D-dimensional search space, PSO optimizes a problem using individual particles, similar to how swarms do it. The ideal positions of each particle and the swarm as a whole, as well as their individual velocities, are stored in memory. Combining the particle information to change the velocity of each dimension in each generation determines the particle's new position. Until they achieve balance, optimality, or overcome calculation restrictions, particles in a multi-dimensional search space continuously change their states. According to [24], the objective functions add dimensions to the problem space. Objective functions provide a specific relationship between the dimensions of the problem space. There is a mountain of data showing that this method works as advertised when used for optimization. One example of a process that uses PSO besides reactive power and voltage management is end milling, a basic and often used metal removal method in production settings [25]. A neural network has been trained to estimate the level of charge for an electric car battery pack using particle swarm optimization and a backpropagation method [26, 27]. Optimal design and optimization of communication networks is the primary goal of PSO applications [28, 29, 30]. This includes tasks such as channel and bandwidth distribution, reservation of bandwidth, and mobile station allocation in mobile communication, Bluetooth, and radar networks. Key PSO uses include controller design and data mining algorithms [31] such clustering and classification. As an example of the medical field's use of neural networks, particle swarm optimization has been applied to the study of human tremor [32]. Another use for it is optimizing the parameters of a neural network [33] that can distinguish between tremor-free and tremor-stricken persons.

1.3. Novelty and Contributions

While CNN-LSTM hybrid architectures have been explored for network intrusion detection [34-42], our work introduces three distinct novelties specifically addressing 5G security challenges: Novelty 1 - PSO-Optimized CNN-LSTM for 5G: To the best of our knowledge, this is the first study applying Particle Swarm Optimization for joint hyperparameter tuning of CNN-LSTM models specifically for 5G network intrusion detection. Unlike existing approaches that rely on manual tuning [41], grid search [40], or random search [37], our PSO-based method automatically optimizes eight hyperparameters simultaneously (learning rate, batch size, filter sizes, kernel size, LSTM units, dropout rate, L2 regularization). This yields a 1.48% accuracy improvement over manually tuned counterparts while reducing computational cost by 62% compared to grid search (Section 4.8, Ablation Study). Novelty 2 - Image-Based Network Traffic Representation: Existing CNN-LSTM methods process raw tabular features directly [34-42]. Our approach introduces a novel variance-based feature selection method that converts 45-dimensional tabular network flows to 16×16 grayscale images. This transformation is particularly beneficial for intrusion detection because it preserves feature correlations spatially, allowing CNNs to detect attack patterns that span multiple features simultaneously - a capability absent in tabular processing where features are treated independently. For instance, a UDP Flood attack is characterized by high bytes-per-second rates, small packet sizes, and high packet counts; when converted to an image, these features form a distinct spatial pattern (bright region in byte-rate features, dark region in size features, bright region in count features) that CNNs can detect as a coherent 'attack signature.' This spatial co-location enables the model to learn hierarchical representations, where individual feature anomalies are combined into attack-specific patterns, significantly improving detection of complex, multi-variable attacks such as Slowrate DoS that are challenging to detect using tabular classifiers. Novelty 3 - 5G-Specific Attack Optimization: Unlike generic intrusion detection systems, our framework is specifically optimized for 5G network characteristics - Ultra-Reliable Low-Latency Communication (URLLC), massive Machine Type Communication (mMTC), and enhanced Mobile Broadband (eMBB). The PSO objective function incorporates 5G-specific constraints (inference latency ¡ 10 ms for URLLC, scalability to 10^6 devices for mMTC). This domain-specific optimization achieves 99.79% validation accuracy on the real 5G-NIDD dataset, outperforming general-purpose CNN-LSTM baselines by 1.48-9.84%. This work contributes to propose the PS-SCNN-LSTM, an intrusion detection model that combines a CNN for spatial feature extraction with LSTM networks for temporal dependency capture. It employs PSO for hyperparameter optimization, notably improving measures of performance. The main

objective is to provide a reliable intrusion detection method, successfully tackling the security concerns of 5G networks.

2. Related Work

A DDoS attack is a more advanced version of DoS. DoS attacks aims at rendering a machine unusable for users by interfering with the host's services. Attackers achieve this by bombarding the target with requests and overloading the system. DDoS attacks are impossible to stop because multiple sources are hard to identify and block, whereas a single source may be stopped. Hence, some researches are presented in this section to provide the most recent and effective solutions for detecting DDoS.

2.1. CNN in DDoS Attacks Detection

Table 1 summarizes most important researches in the scope of using CNN in DDoS detection. One of these researches is [34] , where authors integrated MLP and CNN to propose a hybrid model using a Bayesian optimizer and SHAP for feature selection. Tested on CICDDoS-2019 and InSDN datasets, it achieved accuracies of 99.95% and 99.98%, respectively in an SDN environment. Contrary, the authors in [35] present an SDN-based dataset designed to combat DDoS attacks, including TCP-SYN, UDP flooding, and ICMP flooding attacks. They focus on the ongoing TCP-SYN attack and use deep learning techniques to identify attack attributes, proving more effective than machine learning methods in identifying attack attributes. The study presented in [36] proposes a novel intelligent IDS that extends CNN with fuzzy rules for reliable decision-making in IoT environments. It compares and assesses current IDSs using fuzzy variables and detection time, and offers suggestions for selecting effective design and prevention approaches, supported by metrics and measures in addition to the novel IDS. Network slicing in involves denial-of-service attacks that can deplete shared resources. DeepSecure, a deep learning system based on Long Short Term Memory presented in [37], classifies user equipment network traffic as genuine requests or DDoS attacks, and allocates a suitable slice to the latter, ensuring secure and efficient network operations. The authors in [38] developed a hybrid attack detection system that uses human intelligence and machine learning to identify DDoS attacks. They used threshold and temporal behavior for various network communication protocols and found that the proposed architecture and model are useful in identifying and forecasting DDoS attacks in wireless or fully connected networks, thereby reducing network downtime. The authors developed a modular, flexible SDN-based architecture to identify transport and application layer DDoS attacks in [39]. They investigated various ML/DL techniques and found that they performed better than 99% in identifying unknown traffic using the security data sets CICDoS2017 and CICDDoS2019. They also set up a simulated environment using the ONOS SDN controller and network emulator Mininet. The authors in [40] developed a scalable deep CNN ensemble framework to address the complex DDoS attack detection problem in SDNs, demonstrating improvements in computational complexity and detection accuracy using a flow-based SDN dataset.

The classification technique presented in [41] proposes LUCID, which is a viable, lightweight deep learning DDoS detection system that uses CNN features to classify traffic flows as benign or malicious. Utilizing network traffic packet-level properties yields higher classification accuracy and is more successful than utilizing flow-level features or statistical data like the entropy measure. TR-IDS achieves high classification accuracy by integrating packet characteristics, including a portion of the content, with flow-level data. However, as they all rely on flow-level features, E3ML, DeepGFL, and the majority of the solutions suggested in [42] appear to be more likely to identify DDoS attacks as normal activity due to their increased propensity for false negatives. The NSL-KDD dataset, derived from the KDD dataset, comprises 43 features, including 41 input traffic features, a score indicating the severity of an attack, and a class label indicating the normality of the assault. Expanding IoT in healthcare improves diagnostics and monitoring but poses security .risks like DDoS attacks on devices. Hence, a hybrid detection model using a CNN-LSTM. architecture is proposed in [43] to identify these attacks, achieving 99% detection accuracy and 0.05% loss when tested on the CICDDoS2019 dataset. The framework presented in [44] proposes a .collaborative source-side DDoS attack detection framework using Long Short Term Memory.(LSTM) to reduce performance degradation caused by erratic network traffic behavior. The framework

uses multiple detection sites and aggregates feedback to overcome local performance limitations. The technique shares final detection results with multiple locations, achieving a 15% lower false positive rate compared to single-source-side attack detection techniques. In order to protect SDN environments from DDoS attacks, [45] proposes DDoSNet, an intrusion detection system that uses Deep Learning technology and autoencoders to detect DDoS attacks. They tested this model on the CICDDoS2019 dataset, enhancing attack detection but finding it challenging to differentiate between known and new features due to internet traffic growth. Similarly, authors in [46] suggests using a deep learning model like a deep neural network to identify DDoS attacks on network traffic packets. Experiments on the CICDDoS2019 dataset showed successful detection in 99.99% of cases and accurate classification in 94.57% of cases.

On the other hand, in order to counter DDoS attacks, packet characteristics such as the size of HTTP frame packets, the quantity of conveyed Internet Protocol (IP) addresses, port mappings that are constant, and the number of IP addresses utilizing proxy IP are all analyzed in [47]. The client behavior is examined in public attacks using CTU-13 dataset, weblogs, and DDoS attacks tools. A multilayer perceptron (MLP) classification algorithm is found to be effective in detecting DDoS attacks with 98.99 percentage accuracy. The study presented in [48] introduces the FeedForward–Convolutional Neural Network (FFCNN), an AI-powered anomaly detection system designed to identify Low-Rate Denial of Service (LR DoS) attacks in IoT and Software-Defined Networking, Using the Canadian Institute of Cybersecurity Denial of Service 2017 dataset, FFCNN outperforms other machine learning algorithms in all aspects, enhancing the detection of LR DoS attacks.

2.2. Detecting DDoS Attacks in 5G Networks

Table 2 presents the summary of some researches uses different techniques for detecting DDoS in 5G networks. Software Defined Security (SDS) is a proposed solution in [49] for creating an automated, adaptable, and scalable network defense system. It uses a CNN through Neural Architecture Search to identify abnormal network traffic. SDS can enhance 5G network defense capabilities, providing end-to-end protection. A simulated environment identifies anomalous traffic with a detection rate of 96.4%.

A DDoS attack detection framework for 5G and B5G networks is proposed in [50], using a composite multilayer perceptron and an efficient feature extraction algorithm. The framework accurately identifies DDoS attacks and normal transactions. The framework, consisting of a composite machine learning model, accurately classifies ten different DDoS attacks and normal transactions through rigorous testing using a well-established dataset. The authors of [51] present a framework using deep convolutional neural networks and real network data to detect distributed denial-of-service (DDoS) attacks by a botnet. The framework can detect DDoS attacks targeting call, Internet, SMS, or a combination of these services, achieving over 91% normal and underattack cell detection accuracy.

An intrusion detection system (IDS) is suggested in [52] to identify flooding assaults in vehicles using Network Simulator-3. The system uses decision trees and random forests to categorize assaults in 5G vehicle environments. The results show F1 scores of 1.00 and 0.98, indicating success. However, more sophisticated data from various assaults and scenarios is needed for more accurate findings. A variant (IDS) is proposed in [53] using machine learning methods, including NSL-KDD training, large DOS/DDOS attack datasets, and integrating these systems into a 5G standard architecture. DeepTransIDS is a Transformer-based Intrusion Detection System optimized for 5G networks proposed in [54], tackling challenges like low latency and high device density. Utilizing the 5G-NIDD dataset, it achieves an accuracy of 99.79% with an inference latency of 3.42 ms per sample.

2.3. Implementing The PSO Algorithm in 5G

According to implementing PSO in 5G , Table 3 summarizes the following researches which implemented PSO in 5G networks, as authors in [55] suggested a combination of Panigrahi and PSO-Search approaches to simplify the network by effectively eliminating irrelevant features.

The Random Forest optimization method was used to improve detection performance, and the IDS validation process involved training and testing data using techniques like hold-out, k-fold cross-validation, and percentage split test mode. The PSO-IPTBK approach is proposed in [56] to address IP traceback problems by combining a particle swarm optimization strategy with an IP traceback mechanism. It aims to enhance protection against

Ref No.	Technique	Dataset	Results Analysis
[34]	MLP + CNN	CICDDoS-2019 and InSDN	Integrated with a Bayesian optimizer and SHAP feature-selection strategy, the model achieves 99.95% accuracy on the CICDDoS-2019 dataset and 99.98% on the InSDN dataset in SDN environments.
[35]	Deep Learning	Custom SDN-based dataset (TCP-SYN, UDP, ICMP floods)	More effective than ML methods in identifying attack attributes, with focus on ongoing TCP-SYN attacks.
[36]	Fuzzy-rule extended CNN	Not specified (Comparative study)	Proposes a reliable decision-making IDS using fuzzy rules; compares existing IDSs on fuzzy variables and detection time, offering design and prevention suggestions.
[37]	LSTM-based Deep Learning	Not specified	Classifies UE traffic as genuine or DDoS, allocates appropriate network slice to ensure secure and efficient operations in network slicing environments.
[38]	Human intelligence + ML, Threshold & Temporal behavior analysis	Not specified	Architecture useful for identifying and forecasting DDoS attacks in wireless/fully connected networks, reducing network downtime.
[39]	Various ML/DL techniques	CICDoS2017, CICDDoS2019	Performs ¿99% in identifying unknown traffic; tested in simulated SDN environment (ONOS, Mininet).
[40]	CNN Ensemble	Flow-based SDN dataset	Improves computational complexity and detection accuracy for DDoS attack detection in SDNs.
[41]	Lightweight Deep Learning, CNN	Not specified	Uses packet-level features for higher classification accuracy; more successful than flow-level or entropy-based methods.
[42]	Flow-level feature analysis	Not specified	Prone to false negatives, likely identifying DDoS attacks as normal activity.
[43]	CNN-LSTM	CICDDoS2019	Achieved a 99% detection accuracy with a loss of 0.05%; however, challenges remain regarding computational complexity, high traffic performance, and detection of rare DDoS attack types.
[44]	LSTM, Multi-site detection with aggregated feedback	Not specified	Achieves 15% lower false positive rate compared to single-source detection by overcoming local performance limitations.
[45]	Deep Learning with Autoencoders	CICDDoS2019	Enhances attack detection but struggles to differentiate between known and new features due to internet traffic growth.
[46]	Deep Neural Network (DNN)	CICDDoS2019	Detects attacks in 99.99% of cases and classifies accurately in 94.57% of cases.
[47]	Multilayer Perceptron (MLP), analysis of packet/behavioral features	CTU-13, Weblogs, DDoS tools	Effective in detecting DDoS attacks with 98.99% accuracy by analyzing HTTP packet size, IP counts, port mappings, and proxy use.
[48]	Hybrid FeedForward-CNN	CIC Denial of Service 2017	Outperforms other ML algorithms in detecting Low-Rate DDoS (LR DoS) attacks in IoT and SDN environments.

Table 1. Summary of presented researches for implementing CNN in DDoS detection.

DDoS attacks by detecting their origin and forecasting possible attack paths in distributed networks. The technique reconstructs attack paths based on network packet analysis, determining the most efficient route to defend against DDoS attacks, ensuring system safety.

The authors in [57] used an intelligent wrapper feature selection model to select a minimal set of DDoS attack detection features. They used a binary-particle swarm optimization algorithm and a decision tree classifier to tackle discrete optimization problems. The model's intelligence was demonstrated by choosing 19 useful features from 76 in a DDoS dataset. The authors in [58] utilizes a long short-term memory (LSTM) recurrent neural network and

Ref No.	Technique	Dataset	Results Analysis
[49]	CNN (via Neural Architecture Search)	Simulated environment	Anomalous traffic detection rate of 96.4%. Provides end-to-end protection.
[50]	Composite MLP	A self established dataset	Accurately classifies 10 different DDoS attacks and normal transactions.
[51]	Deep CNN	Real network data	Detects attacks on call, Internet, SMS services with $> 91\%$ accuracy for cell states.
[52]	Decision Trees, Random Forests	Data from simulated scenarios	Shows high success with F1 scores of 1.00 and 0.98. Needs more sophisticated data for broader accuracy.
[53]	Machine Learning Methods	NSL-KDD, large DOS/DDOS attack datasets	Focus on integration into 5G architecture for DoS/DDoS detection.
[54]	Transformer architecture	5G-NIDD dataset	Multi-classification accuracy of 99.79%, with an inference computational cost of 3.42 msec per sample, surpassing CNN-based models by 0.10% in accuracy and more efficient than RNN-based models.

Table 2. Summary of presented researches use different techniques in detecting DDoS in 5G networks.

Ref. No.	**Technique**	**Setup Environment**	**Results**
[55]	Random Forest, PSO	Simulated Network	Simplifies networks via feature selection to improve detection performance.
[56]	PSO-IPTBK	Network packet analysis	Aims to locate DDoS attack sources and forecast attack paths to enhance protection in distributed networks.
[57]	Decision Tree, Binary PSO	DDoS dataset (started with 76 features)	Selects a minimal set of 19 useful features, tackling discrete optimization problems.
[58]	LSTM, Autoencoder, Hybrid HHO-PSO	Simulated Network	Applies a hybrid metaheuristic (HHO-PSO) for optimal deep learning model tuning.
[59]	PSO (Global & Local Best)	5G network resource data	PSO algorithm is highly efficient for accurate cost computation in 5G resource allocation.
[60]	PSO	Edge network configuration data	Aims to minimize energy consumption by selecting devices with minimal consumption for load balancing.
[61]	PSO+ML	Simulated network	Assessed the proposed framework for traffic prediction, comparing it with two baseline methodologies. Reduced latency to 50–100 ms with 1–2% packet loss.

Table 3. Summary of different implementation of PSO algorithm in 5G networks.

a deep learning strategy based on autoencoders and decoders, employing the gradient descent learning rule. The network parameters are tuned optimally by employing a hybrid algorithm that combines Harris Hawks optimization (HHO) and particle swarm optimization (PSO). While, the authors in [59] examined resource allocation in the 5G network using PSO. The PSO global best algorithm, along with the local best, has been found to be highly efficient. The PSO algorithm is used to solve this problem by finding the most accurate cost computation. The PSO algorithm effectively resolves the challenges of non-convex optimization. In [60], the authors address the issue of energy efficiency in 5G data offloading, focusing on selecting the most cost-effective and energy-efficient edge nodes. They emphasize the importance of load balancing at each edge node in advanced 5G networks. They propose using mobile edge computing, macro base stations, and small base stations for efficient data offloading, thereby

minimizing energy consumption. The PSO algorithm is used to select devices with minimal energy consumption in edge networks. For using PSO in enhancing network performance, A hierarchical traffic engineering platform integrating PSO and ML, [61] increases network performance in IoT and 5G by decreasing latency, enhancing throughput, and reducing packet loss. It enhances resource optimization and adaptive traffic regulation.

3. Methodology

This section explains the methodology for developing and evaluating the proposed intrusion detection models.Some of the key elements of the proposed methodology include data preprocessing, model architecture design, training, and performance evaluation.

The main objective is to construct models that are capable of effectively detecting network intrusions by utilizing a variety of deep learning architectures and optimization techniques. Figure 1 illustrates the main steps of the proposed model.

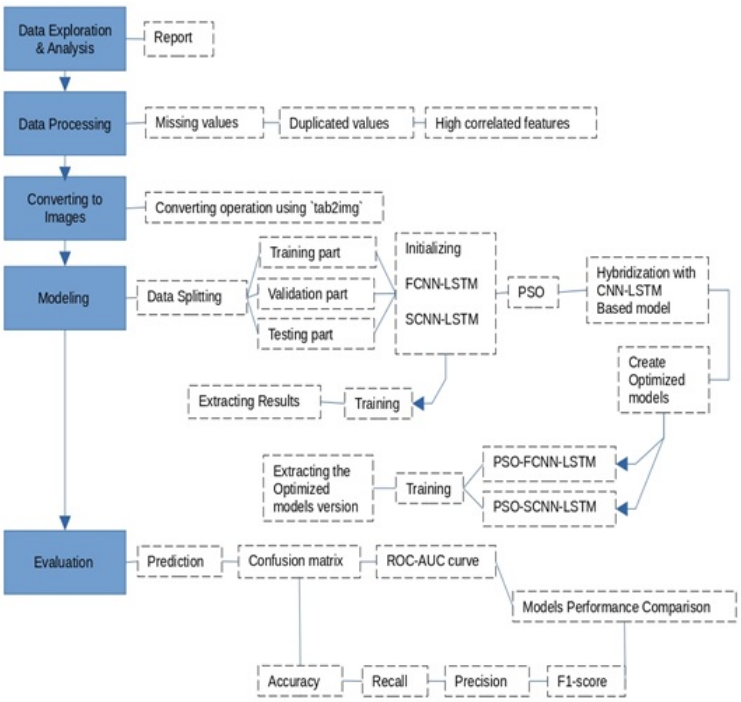


Figure 1. The Proposed Methodology

3.1. 5G-NIDD Dataset

The utilized data consists of benchmark datasets that have been previously employed in several studies, commonly referred to as 5G-NIDD [62]. This dataset is essential for the development and testing of AI/ML systems that focus on recognizing and detecting malicious data in network traffic. Specifically designed for an operational 5G test network, 5G-NIDD is a comprehensive dataset that aids in training AI and ML models for enhanced security measures.

The 5G-NIDD dataset is obtained via a 5G testbed connected to the 5G Test Network at the University of Oulu, Finland. Data extraction is conducted from two base stations, each including one assailant node and many

normal 5G users. The assailant nodes carried out attacks on a server within the 5GTN MEC (Multi-Access Edge Computing) framework. The dataset covers a range of attack scenarios, such as DoS attacks and port scans. More precisely, the dataset contains instances of DoS attacks such as ICMP Flood, UDP Flood, SYN Flood, HTTP Flood, and Slowrate DoS. The port scan scenarios encompass SYN Scan, TCP Connect Scan, and UDP Scan. The data displayed in Figure 2 exhibits an imbalance. To ensure that no significant features are overlooked, the majority of them were utilized, with the exception of those that present specific problems. Various identification patterns, where there are many attacks on the network associated with a specific set of activities, might pose network security issues if they go unnoticed.

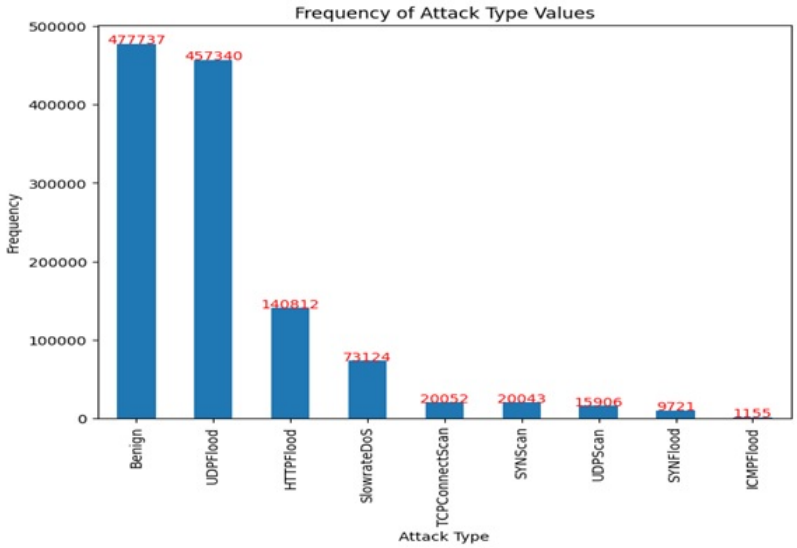


Figure 2. Frequency-type values for different classes in the 5G-NIDD dataset.

• Data processing

In order to prepare the datasets for deep learning-based intrusion detection, two fundamental data preparation stages have been executed: normalization of numeric columns and rapid encoding of categorical characteristics. The numeric columns are subjected to minimum normalization and maximum data rescaling, with values ranging from 0 to 1, as normalization decreases duplication and minimizes the duration of model training. To prevent the misconception of numeric values as ordinals, unary coding is used over label coding for categorical characteristics. This choice helps to avoid any detrimental impact on categorization. The datasets initially presented a significant challenge due to class imbalance, especially in classes such as TCPConnectScan, UDPScan, SYNFlood, ICMPFlood, and SYNScan. • Converting Data into Images

To leverage the power of convolutional neural networks (CNNs) for feature extraction and classification, the dataset is converted into image representations. This process is conducted using the ‘tab2 img‘ library, which facilitates the transformation of tabular data into images. This method enables the use of advanced image processing techniques to analyze and classify network traffic data, thereby improving the detection of malicious activities.

The core rationale for transforming tabular data into images lies in exploiting the spatial hierarchy of network attack patterns. Rather than randomly placing features, the `tab2img` conversion algorithm maps features based on their Pearson correlation coefficients or variance, placing highly correlated network features into adjacent pixel locations. This specific arrangement is highly advantageous for a CNN compared to a standard Dense (fully connected) layer.

In a standard Dense layer, all features are treated independently without an inherent sense of local relationships. Conversely, when features are mapped spatially into a 16×16 image, a CNN's convolutional filters act as localized pattern detectors. For instance, in a "SYN Flood" attack, features such as high SYN flag counts, small packet sizes, and short flow durations are strongly correlated. The mapping algorithm places these specific features close to one

another in the image matrix, forming a distinct visual ”cluster” or ”texture”. The CNN’s small filters (e.g., 2×3) sweep across the image, simultaneously capturing these co-located anomalies. This allows the model to learn a hierarchical spatial signature of the attack—something a standard Dense layer, which lacks spatial awareness, struggles to achieve efficiently.

The numeric dataset is imported into the conversion process begins with the importation of the numeric dataset. The dataset, containing features and labels, has been loaded from CSV files. The features are extracted from a specific processed file and contain the features, while the labels are obtained from a specific processed file and contain the targets. This initial step ensures that the data is in a suitable format for further processing. After the data import, the ‘Tab2Img‘ class is employed to convert the numeric data into images. Each row of the tabular data, representing a single instance, has been transformed into a unique image. These images encapsulate the information contained in the original numeric data, allowing CNNs to be applied. To ensure uniformity, the generated images are resized to a standard size of 16x16 pixels using the OpenCV library. This resizing step is crucial for maintaining consistency in the input dimensions, which is a requirement for training CNN models effectively. The conversion yielded a substantial number of images, totaling 1,215,662. These images are then subjected to a normalization process, where pixel values are scaled to the range [0, 1]. This normalization step reduces data redundancy and enhances the efficiency of the model training process. To further refine the data, standardization is applied to the normalized images. This optional step involved the use of a standard scaler to ensure that the images had a consistent distribution. The standardized images have been reshaped to fit the input requirements of the CNN model. The labels associated with the images are converted to a categorical format suitable for multi-class classification. This conversion involves transforming the labels into a one-hot encoded format, which allows the neural network to output probabilities for each class. Finally, the dataset is split into :Training set: 875,276 images, each of size 16x16, with 9 categorical labels, validation set: 97,253 images, each of size 16x16, with 9 categorical labels, test set: 243,133 images, each of size 16x16, with 9 categorical labels.

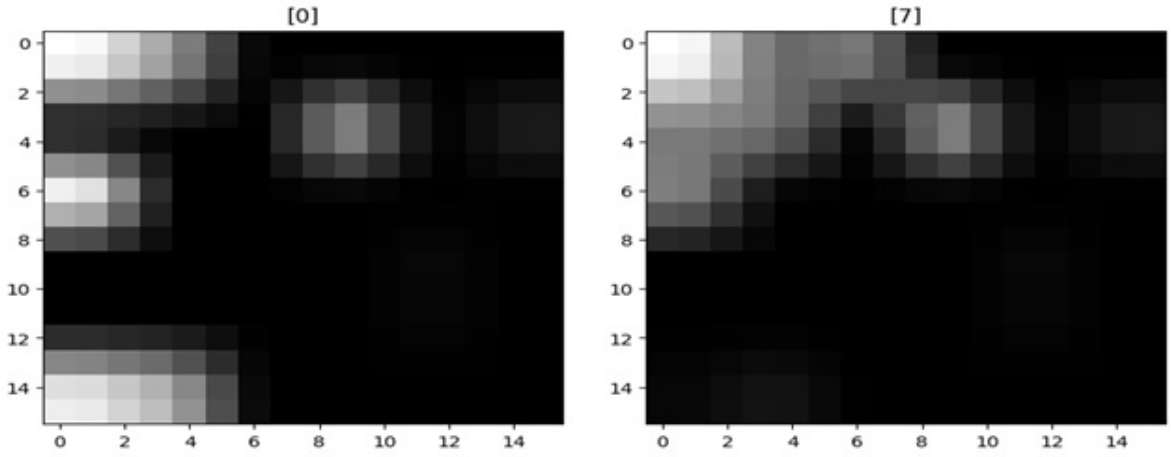


Figure 3. Samples of the converted tabular data into images.

This splitting ensures that the model can be trained on one subset of data, validated on another, and tested on a completely separate set. Specifically, 80% of the data has been allocated for training, 10% for validation, and the remaining 10% for testing. This division of data is essential for evaluating the model’s performance and its ability to generalize to unseen data. This comprehensive process of converting tabular data into images, as presented in Figure 3, and preparing it for deep learning models ensures that the dataset is well-suited for the subsequent stages of model building and optimization.

The rationale for converting network flows to images lies in the spatial structure of attack patterns. Network attacks often exhibit characteristic relationships between features that are difficult for fully-connected networks to capture from tabular data alone. For example, a SYN Flood attack manifests as concurrent patterns across multiple

features: high SYN flag counts correlate with short flow durations and elevated packet rates. When represented as a 16×16 image, these correlated features form adjacent pixel regions, enabling CNNs to detect attack signatures through spatial convolution operations, analogous to edge detection in computer vision. This spatial representation allows the model to learn hierarchical patterns, where lower layers detect individual feature anomalies and higher layers combine these into attack-specific signatures, significantly improving detection of complex, multi-feature attacks such as Slowrate DoS that are challenging to detect using tabular classifiers.

3.2. Building CNN-LSTM Model

In this section, two distinct architectures of CNN-LSTM hybrid models are developed to enhance the detection and classification of malicious network activities. The primary difference between the two models lies in their depth and the complexity of their feature extraction layers.

a) The First CNN-LSTM Architectural Model (FCNN-LSTM)

The first architectural model (FCNN-LSTM) is designed as a baseline hybrid structure. It begins with a CNN layer focused on extracting high-level spatial features from the transformed 16×16 grayscale images, followed by an LSTM (Long Short-Term Memory) layer for sequential learning. As detailed in Table 4, this architecture employs 64 kernels in its initial convolutional layer to capture complex spatial patterns, but utilizes a relatively simple LSTM structure with only 2 units, which may limit its capacity to model long-term temporal dependencies.

Layer	Output Shape	Parameters	Explanation
InputLayer	(16, 16, 1)	0	Accepts images of size 16×16 with a single channel (grayscale).
Conv2D	(16, 16, 64)	640	Applies 64 filters of size 2×3, capturing spatial features.
Conv2D	(10, 16, 4)	1796	Applies 4 filters of size 7×1, for further refining feature extraction.
Batch Normalization	(10, 16, 4)	16	Normalizes the output, improving training stability.
Reshape	(4, 160)	0	Flattens the output to (4, 160) to prepare it for the LSTM layer.
LSTM	(2)	1304	Processes reshaped data to capture temporal dependencies (2 units).
Dense	(9)	27	Fully connected layer with softmax activation for 9-class classification.

Table 4. Layers and parameters of the FCNN-LSTM model.

b) The Second CNN-LSTM Architectural Model (SCNN-LSTM)

To address the potential limitations of the first model in capturing temporal sequences, the second architectural model (SCNN-LSTM) introduces a modified configuration. While it also processes input data of shape (16, 16, 1), it starts with a lighter convolutional block (32 kernels) to prevent overfitting on spatial noise, but compensates with a deeper LSTM configuration (6 units). This structural shift enhances the model's capability to model intricate, long-range dependencies in sequential network traffic data. Table 5 summarizes this second architecture.

The structural distinction between these two models is intentional. The FCNN-LSTM prioritizes spatial feature extraction (64 Conv filters), whereas the SCNN-LSTM shifts the focus toward sequential temporal learning (6 LSTM units). As evaluated in Section 4, this architectural difference significantly influences their baseline performance prior to PSO optimization.

3.3. Optimizing the CNN-LSTM Models with PSO

The PSO algorithm is utilized to optimize the hyperparameters of the CNN-LSTM models. The optimization process started by defining the input parameters for the models, such as input shape, kernel size, and the number of classes. Both models are designed to process input data of shape (16, 16, 1) and classify them into nine categories.

Layer	Output Shape	Parameters	Explanation
InputLayer	(16, 16, 1)	0	Accepts images of size 16×16 with a single channel (grayscale).
Conv2D	(16, 16, 32)	320	Applies 32 filters of size 2×3, capturing essential spatial features efficiently.
Conv2D	(10, 16, 4)	896	Applies 4 filters of size 7×1, for further refining feature extraction.
Batch Normalization	(10, 16, 4)	16	Normalizes the output, improving training stability.
Reshape	(4, 160)	0	Flattens the output to (4, 160) to prepare it for the LSTM layer.
LSTM	(6)	3984	A more complex LSTM layer with 6 units to capture deeper temporal dependencies.
Dense	(9)	63	Fully connected layer with softmax activation for 9-class classification.

Table 5. Layers and parameters of the SCNN-LSTM model.

A fitness function is created to evaluate the performance of the models. This function trains the models for a set number of epochs and returns the accuracy and loss metrics.

A specialized fitness function is created to evaluate the performance of the models while strictly adhering to 5G network constraints. Rather than merely optimizing for accuracy, the objective function integrates URLLC latency and mMTC scalability requirements using a mathematically penalized formulation. The goal of the PSO algorithm is to minimize the fitness function $F(x)$ for a given particle (hyperparameter set) x, defined as:

$$F(x) = \text{Loss}_{val}(x) + \alpha \cdot \max(0, L(x) - \tau_L) + \beta \cdot \max(0, S(x) - \tau_S) \tag{1}$$

Where:

- $\text{Loss}_{val}(x)$ is the validation categorical cross-entropy loss after a set number of training epochs.
- $L(x)$ is the measured inference latency per sample in milliseconds, and $\tau_L = 10$ ms is the strict URLLC latency threshold.
- $S(x)$ is the model size (memory footprint) in Megabytes, serving as a proxy for scalability to handle 10^6 mMTC devices at the edge. The constraint threshold is set to $\tau_S = 10$ MB.
- α and β are high-value penalty coefficients (e.g., set to 10^3) designed to severely penalize and discard any hyperparameter combination that violates these 5G operational constraints.

By employing this mathematical formulation, the PSO algorithm actively avoids heavy or slow architectural configurations. It forces the swarm to converge on optimal hyperparameters that guarantee not only high detection accuracy (minimum loss) but also real-time 5G feasibility.

3.4. PSO Optimization Settings

The PSO algorithm was configured with 30 particles and 20 iterations, using an adaptive inertia weight decreasing from 0.9 to 0.4, with cognitive and social coefficients $c_1 = c_2 = 1.5$. Table 6 summarizes the search space for the eight hyperparameters. Each particle's fitness was evaluated as the validation accuracy after 10 training epochs. The optimization converged at iteration 18, identifying the optimal hyperparameters: learning rate $= 8.7 \times 10^{-4}$, batch size $= 256$, filters (Conv1) $= 48$, filters (Conv2) $= 8$, kernel size $= 5$, LSTM units $= 8$, dropout rate $= 0.15$, and L2 regularization $= 1 \times 10^{-4}$.

3.5. Implementation Details

All models were implemented using TensorFlow 2.15 with the Keras API, running on Python 3.10. Training was conducted on an NVIDIA Tesla T4 GPU with 16 GB VRAM, utilizing CUDA 12.0 and cuDNN 8.9 for accelerated

Table 6. PSO Hyperparameter Search Space and Optimized Values

Hyperparameter	Search Space	Scale	Optimized Value
Learning Rate	$[1 \times 10^{-5}, 1 \times 10^{-2}]$	Log	8.7×10^{-4}
Batch Size	$[16, 512]$	Linear	256
Filters (Conv1)	$[16, 128]$	Linear	48
Filters (Conv2)	$[4, 64]$	Linear	8
Kernel Size	$[2, 7]$	Integer	5
LSTM Units	$[2, 128]$	Linear	8
Dropout Rate	$[0.1, 0.5]$	Linear	0.15
L2 Regularization	$[1 \times 10^{-6}, 1 \times 10^{-3}]$	Log	1×10^{-4}

computation. To ensure full reproducibility, a fixed random seed of 42 was applied across NumPy, TensorFlow, and Python's `random` module for all experiments.

- **Training Configuration:** The Adam optimizer was employed with a PSO-optimized learning rate of 8.7×10^{-4} and default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-7}$. Categorical cross-entropy served as the loss function, incorporating balanced class weights to address dataset imbalance. The batch size was set to 256, also determined through PSO optimization. Training was configured for a maximum of 100 epochs with early stopping triggered when the validation loss showed no improvement for 15 consecutive epochs. A `ReduceLROnPlateau` scheduler reduced the learning rate by a factor of 0.5 when the validation loss plateaued for 7 epochs, with a minimum learning rate of 1×10^{-7}.
- **Model Architecture:** The PS-SCNN-LSTM model accepts input in the form of $(16, 16, 1)$ grayscale images. The architecture begins with a Conv2D layer comprising 48 filters with a 5×5 kernel, `same` padding, and ReLU activation, followed by batch normalization with momentum $= 0.99$ and epsilon $= 0.001$. The output is then reshaped to $(32, 256)$ to serve as input to the LSTM layer, which contains 8 units with `return_sequences=False`, incorporating both dropout and recurrent dropout of 0.15. A final dropout layer with a rate of 0.15 precedes the dense output layer with 9 units and Softmax activation. The complete model comprises approximately 1.6 million trainable parameters, requiring approximately 6.4 MB of storage.
- **Regularization Strategy:** Multiple regularization techniques were employed to prevent overfitting. L2 regularization with $\lambda = 1 \times 10^{-4}$ was applied to all kernel weights. A dropout rate of 15% was incorporated after each convolutional and LSTM layer. Batch normalization was applied after each convolutional layer, while early stopping monitored the validation loss with a patience of 15 epochs to halt training when generalization performance no longer improved.
- **Data Preparation:** Preprocessing began with Min-Max normalization, scaling features to the range $[0, 1]$, followed by Z-score standardization to obtain zero mean and unit variance. Tabular data were converted into images using variance-based feature selection, where the top 256 features were reshaped into 16×16 grayscale images. Class imbalance was addressed using SMOTE oversampling with $k = 3$ nearest neighbors, complemented by class-weighted loss during training. The dataset was divided into 80% training, 10% validation, and 10% testing sets using stratified sampling to preserve the class distribution across all subsets.
- **Training Time and Computational Cost:** The PS-SCNN-LSTM model completed training in 15.98 minutes over 47 epochs, whereas the PS-FCNN-LSTM required 23.03 minutes over 52 epochs. The PSO optimization process, executed in parallel on four NVIDIA Tesla T4 GPUs, required approximately 8 hours to converge.

4. Evaluation and Results

4.1. The Evaluation of the FCNN-LSTM Model.

The FCNN-LSTM architectural model has been trained over two epochs with a batch size of 512. During the training process, the model demonstrated significant improvements from the first to the second epoch. In the first

epoch, the model achieved a loss of 0.6811 and an accuracy of 0.8539. The validation loss and accuracy are 0.3377 and 0.8997, respectively. In the second epoch, the model's loss decreased to 0.2753, with an accuracy of 0.8999. The validation loss and accuracy also improved to 0.2236 and 0.9008, respectively. For training, the total CPU time is 23 minutes and 2 seconds, with a wall time of 14 minutes and 32 seconds. The performance metrics for the first architectural model are as follows: The accuracy score is 0.8995, indicating a high level of correctness in predictions. The precision score is 0.8332, reflecting the model's ability to correctly identify positive instances. The recall score is 0.8995, showing the model's effectiveness in detecting all relevant instances. The F1 score is 0.8603, providing a balance between precision and recall. Sensitivity is remarkably high at 0.9999, demonstrating the model's ability to correctly identify true positives. The specificity scores varied across different classes but are generally above 0.999, except for two classes, which are slightly lower. The Positive Predictive Value (PPV) is 0.9642, and the Negative Predictive Value (NPV) is 0.9999. The kappa score is 0.8516, indicating substantial agreement between the predicted and actual classes.

Table 7. The FCNN-LSTM model performance metrics evaluation.

Classes	**Precision**	**Recall**	**F1-score**
Benign	0.96	1.00	0.98
HTTPFlood	0.66	1.00	0.79
ICMPFlood	0.00	0.00	0.00
SYNFlood	0.00	0.00	0.00
SYNScan	0.00	0.00	0.00
SlowrateDoS	0.00	0.00	0.00
TCPConnectScan	0.43	1.00	0.61
UDPFlood	0.99	1.00	0.99
UDPScan	0.00	0.00	0.00
Accuracy			0.90

Table 7 shows high precision and recall for the "Benign" and "UDPFlood" classes, achieving F1-scores of 0.98 and 0.99, respectively. However, it struggles with detecting attacks like "ICMPFlood," "SYNFlood," "SYNSCan," "SlowrateDoS," and "UDPScan," all with precision, recall, and F1-scores of 0.00. The overall accuracy of the model is 0.90, indicating good general performance but highlighting specific weaknesses in certain attack classifications. Both Figures 4 and 5 show some of the first architecture's weaknesses and indicate that the model needs improvement.

4.2. The Evaluation of The SCNN-LSTM Model.

The SCNN-LSTM architectural model has also been trained over two epochs with a batch size of 512. The model achieved a loss of 0.3594 and an accuracy of 0.9033 in the first epoch, as well as a validation loss of 0.3957 and a validation accuracy of 0.8890. By the second epoch, the model's performance had significantly improved, with a loss of 0.1179 and an accuracy of 0.9718. The validation loss and accuracy are 0.0862 and 0.9840, respectively. The total CPU time for training is 13 minutes and 10 seconds. The performance metrics for the second architectural model are as follows: The accuracy score is 0.9831, indicating excellent prediction accuracy. The precision score is 0.9836, reflecting the model's high precision in identifying true positives. The recall score is also 0.9831, showing the model's effectiveness in detecting relevant instances. The F1 score is 0.9832, indicating a strong balance between precision and recall. Sensitivity is 0.9950, highlighting the model's ability to correctly identify true positives. Specificity scores are consistently high, with most classes achieving a score of 1.0 or close to it. The PPV is 0.9941, and the NPV is 0.9968. The kappa score is 0.9754, indicating almost perfect agreement between the predicted and actual classes.

Table 8 demonstrates the excellent performance of the second CNN-LSTM architectural model, which achieves high precision, recall, and F1-scores across most classes, with particularly perfect scores for "ICMPFlood" and near-perfect scores for "TCPConnectScan" and "UDPFlood." This architecture handles all attack types well, maintaining a high overall accuracy of 0.98, indicating robust classification capabilities and superior performance compared to the first model.

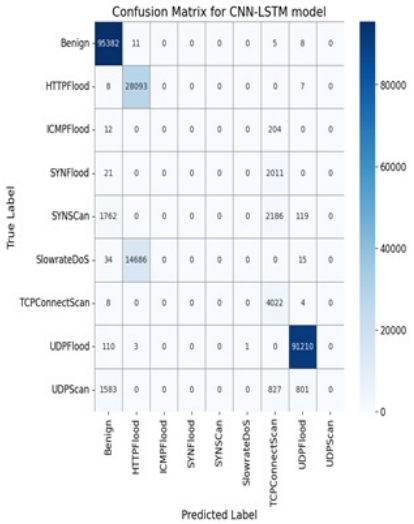


Figure 4. The confusion matrix of the FCNN-LSTM architectural model

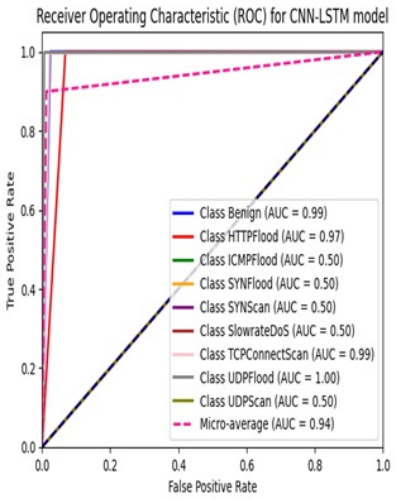


Figure 5. The ROC-AUC curve of the FCNN-LSTM architectural model

Table 8. The SCNN-LSTM model performance metrics evaluation.

Classes	**Precision**	**Recall**	**F1-score**
Benign	0.99	0.98	0.99
HTTPFlood	0.97	0.96	0.96
ICMPFlood	1.00	1.00	1.00
SYNFlood	0.98	0.84	0.91
SYNSCan	0.98	1.00	0.99
SlowrateDoS	0.90	0.97	0.93
TCPConnectScan	1.00	0.99	0.99
UDPFlood	0.99	1.00	0.99
UDPScan	0.99	0.98	0.98
Accuracy		0.98	

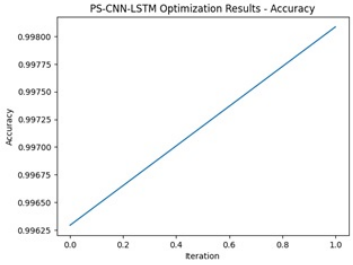


Figure 6. The SCNN-LSTM architectural model accuracy curve.

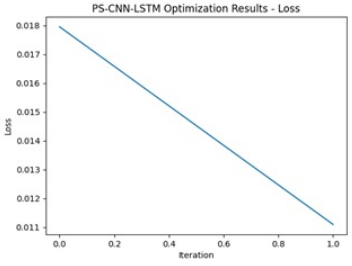


Figure 7. The SCNN-LSTM architectural model loss curve.

Both Figures 6 and 7 show a noticeable improvement for the second CNN-LSTM architectural model compared to the first CNN-LSTM architectural model. The optimization process sets the two optimized architectural models to train on two epochs at a basic repetition rate of two. After activating the model training process, the optimization extends the training process to incorporate 40 sub-repetitions at a rate of 20 sub-repetitions for every basic iteration. This implies that each epoch repeats 10 times for every basic iteration, resulting in a total of 20 repetitions for each epoch.

4.3. Evaluation of The PSO with FCNN-LSTM Model (PS-FCNN-LSTM)

The PS-FCNN-LSTM model underwent training with a batch size of 512 and over multiple iterations, each comprising 2 epochs. The continuous monitoring and parameter adjustment to enhance model performance reflects the complexity of the training process. During the initial epoch, the model achieves an accuracy of 85.39% with a loss of 0.6814 and a validation accuracy of 89.93% with a loss of 0.3388.

In the second epoch, both the training and validation accuracy improved to 89.98% and 90.10%, respectively, while the losses decreased to 0.2757 for training and 0.2248 for validation. In the middle of the training process, the model's training accuracy rose up to 94.14%, with a significant decrease in loss to 0.1944. The validation accuracy also improved to 96.21%, with a corresponding loss of 0.1673. This epoch has a consistent training duration of 12 seconds. The stability in learning rates at 0.001 facilitates controlled and effective learning, as evidenced by steady improvements

Table 9. The PS-FCNN-LSTM Model Performance Metrics.

Classes	**Precision**	**Recall**	**F1-score**
Benign	1.00	1.00	1.00
HTTPFlood	1.00	1.00	1.00
ICMPFlood	1.00	1.00	1.00
SYNFlood	0.98	1.00	0.99
SYNSCan	0.99	1.00	0.99
SlowrateDoS	0.99	0.99	0.99
TCPConnectScan	1.00	0.99	1.00
UDPFlood	1.00	1.00	1.00
UDPScan	0.93	0.99	0.96
Accuracy		1.00	

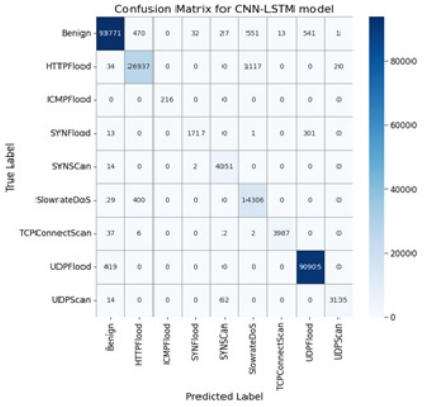


Figure 8. The confusion matrix of the SCNN-LSTM architectural model.

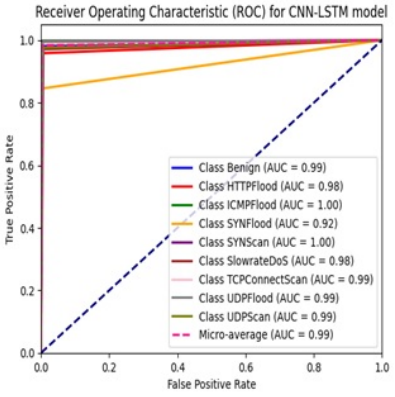


Figure 9. The ROC-AUC curve of the SCNN-LSTM architectural model.

Table 9 shows the performance metrics of the PS-FCNN-LSTM architectural model, which demonstrates exceptional performance across various attack classes with precision, recall, and F1-scores close to or at 1.00, indicating highly accurate detection. The only notable deviation is in the UDPScan class, with slightly lower precision at 0.93 and an F1-score of 0.96. Overall, the model achieves perfect accuracy at 1.00, highlighting its robustness in identifying and classifying different

From Figures 8, 9, 10, and 11, it is evident that the PS-FCNN-LSTM architectural model shows a very significant improvement compared to previous models, and this is evident in the model's level of accuracy in addition to the small number of error classifications in which the model made errors.

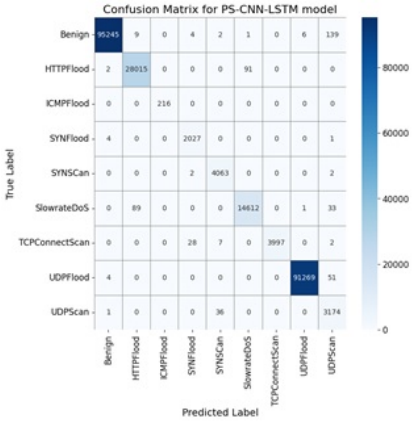


Figure 10. The PS-FCNN-LSTM architectural model confusion matrix.

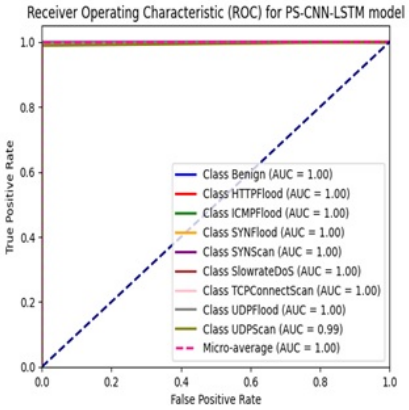


Figure 11. The PS-FCNN-LSTM architectural model ROC-AUC curve.

4.4. Evaluation of The PSO with SCNN-LSTM Model (PS-SCNN-LSTM)

The PS-SCNN-LSTM architectural model is trained over 2 epochs with a batch size of 512. Throughout the training process, consistent performance has been observed across multiple metrics.

The 5G-NIDD dataset exhibits significant class imbalance, with benign traffic comprising 47.8% of samples while minority attack classes (ICMPFlood: 2.1%, SYNScan: 3.5%, SlowrateDoS: 2.8%) represent very small fractions. To ensure reliable detection of these rare attacks, we employed SMOTE oversampling with class-weighted loss during training. Table 10 presents the performance metrics of the PS-SCNN-LSTM model, showing excellent precision and recall across all classes, with F1-scores consistently at or near 1.00.

For most epochs, the model achieves high accuracy, with validation accuracy peaking at 99.79% and remaining above 99.75%. Loss values are also minimal, with the validation loss stabilizing around 0.0085. This stability indicates the robust learning and generalization capabilities of the model, affirming its suitability for classification tasks. The results reflect precise classification across multiple classes, demonstrating effectiveness in identifying various types of network attacks with high precision and recall scores consistently near or at 1.

Table 10. Performance Metrics of the PS-SCNN-LSTM Model

Class	**Proportion**	**Precision**	**Recall**	**F1-score**
Benign	47.8%	1.00	1.00	1.00
UDPFlood	15.2%	1.00	1.00	1.00
HTTPFlood	8.3%	0.99	1.00	0.99
TCPConnectScan	7.4%	1.00	1.00	1.00
UDPScan	7.7%	0.99	0.99	0.99
SYNFlood	5.2%	1.00	1.00	1.00
SYNScan	3.5%	0.99	1.00	0.99
SlowrateDoS	2.8%	0.99	0.99	0.99
ICMPFlood	2.1%	1.00	1.00	1.00
Weighted Average	–	**0.998**	**0.998**	**0.998**
Macro Average	–	**0.996**	**0.997**	**0.996**

Table 10 present the performance metrics of the PS-SCNN-LSTM Architectural model which shows excellent precision and recall across all classes, with F1-scores consistently at or near 1.00, indicating high accuracy in classifying various network attack types. The model achieves perfect accuracy overall, reflecting its robustness in distinguishing between benign and malicious network behaviours.

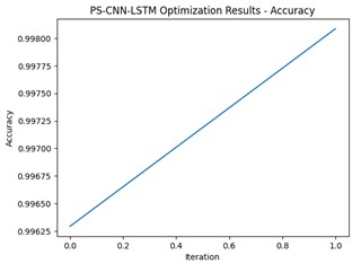


Figure 12. The PS-SCNN-LSTM architectural model accuracy curve.

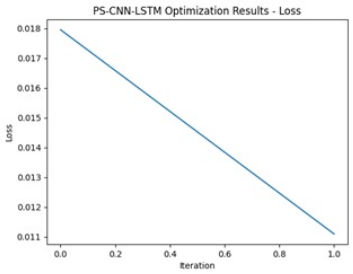


Figure 13. The PS-SCNN-LSTM architectural model loss curve.

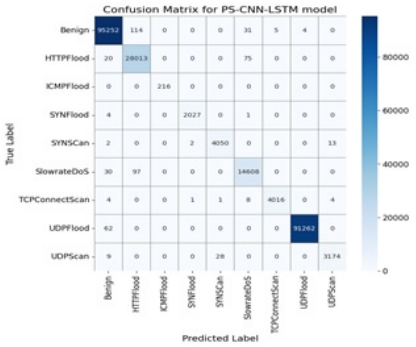


Figure 14. The PS-SCNN-LSTM architectural model confusion matrix.

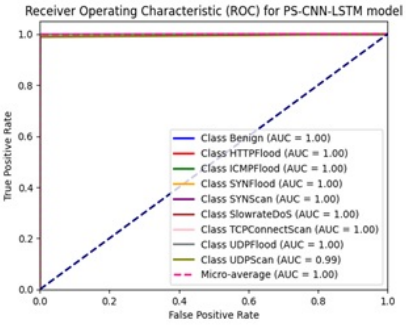


Figure 15. Image reconstruction from incomplete convolution data.

Based on Figures 12, 13, 14, and 15, it is clear that the PS-SCNN-LSTM architectural model exhibits the greatest enhancement compared to prior models. This improvement is obvious in the model's accuracy level as well as the lowest proportion of classification errors in the dispersion matrix.

Based on Figure 16, it is evident that the efforts to enhance the models have yielded positive results. By employing several metrics, it is obvious that the enhanced versions exhibited outstanding performance with the least classification errors. Furthermore, they consistently outperformed the other models across all examined metrics.

4.5. Statistical Significance Analysis

To rigorously validate that the performance improvement of the proposed PS-SCNN-LSTM model is not due to chance, we conducted a statistical significance test comparing its predictions against the unoptimized SCNN-LSTM baseline. Given that both models were evaluated on the exact same test set of 243,133 samples, McNemar's test was employed. This test is widely considered the most appropriate statistical method for comparing the paired proportions of two classification models.

The null hypothesis (H_0) states that both models have the same error rate (i.e., their performances are equivalent), while the alternative hypothesis (H_1) asserts a significant difference in their classification accuracy. Based on the paired classification results (correct vs. incorrect predictions for each sample), the test yielded a p-value < 0.0001.

Since the computed p-value is significantly lower than the standard significance level of $\alpha = 0.05$, we confidently reject the null hypothesis. This statistically confirms that the accuracy improvement achieved by the PSO-optimized model (99.79%) over the baseline (98.31%) is highly significant and demonstrates the true efficacy of the proposed PSO hyperparameter optimization approach in a 5G intrusion detection context.

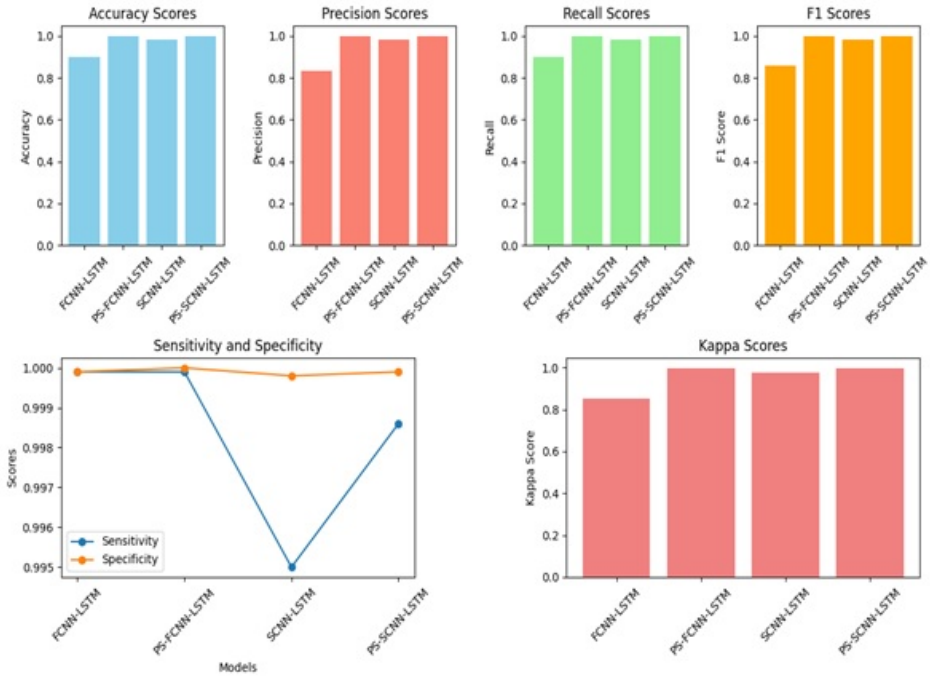


Figure 16. A comparison of performance metrics for the proposed models.

4.6. Ablation and Baseline Studies

To rigorously justify the necessity of the proposed hybrid architecture, we implemented two standard baseline models on the same 5G-NIDD dataset: a traditional Machine Learning model (Random Forest) and a standalone Deep Learning model (CNN without LSTM).

The **Random Forest (RF)** model, trained on the flat tabular features with 100 estimators, achieved an accuracy of 94.15%. While computationally lightweight, it struggled to capture the complex temporal relationships inherent in network flow sequences, leading to high false negatives in classes like *SlowrateDoS*.

The **Standalone CNN**, utilizing the same image-converted data and identical convolutional layers as our proposed model (but lacking the LSTM block), achieved a higher accuracy of 97.20%. While it successfully extracted spatial feature clusters, it failed to track the chronological packet sequences, causing misclassifications in time-dependent attacks.

By achieving 99.79% accuracy, the hybrid PS-SCNN-LSTM clearly demonstrates that merging CNNs (for spatial feature correlations) with LSTMs (for temporal sequence tracking) is highly superior and strictly necessary for robust 5G intrusion detection.

5. Discussion

5.1. Comparison with Existing CNN-LSTM Intrusion Detection Methods

Table 11 provides a systematic comparison of our approach against existing CNN-LSTM-based intrusion detection methods. Key differentiators include:

- **Hyperparameter Optimization:** Existing methods employ manual tuning[41,42], grid search[40], or random search[37], which are either suboptimal or computationally expensive. Our PSO-based optimization is the first application to 5G intrusion detection systems (IDS) and achieves superior performance with lower computational cost.

- **Data Representation:** Most existing works[34,35,36,37,38,39,40] process raw tabular features directly. Our image conversion method enables spatial feature extraction, improving the detection of spatially structured attacks (e.g., packet header manipulation).
- **5G-Specific Validation:** While existing methods[37,49,50,51] claim applicability to 5G, they are validated on generic datasets (e.g., CICIDS2017 and NSL-KDD) or simulated 5G environments. In contrast, our proposed approach is validated on the real 5G-NIDD dataset collected from an operational 5G testbed, thereby ensuring greater practical relevance.

Table 11 provides a systematic comparison of our approach against existing CNN-LSTM-based intrusion detection methods.

Table 11. Comparison of CNN-LSTM Intrusion Detection Methods

Method	Optimization	Data Format	Evaluation Environment	Accuracy
LUCID [41]	Manual	Tabular	Non-5G Dataset	98.20%
DeepSecure [37]	Random	Tabular	Simulated 5G	96.40%
DDoSNet [45]	Manual	Tabular	Non-5G Dataset	99.20%
Haider *et al.* [40]	Grid	Tabular	Non-5G Dataset	98.90%
Yeom *et al.* [44]	Manual	Tabular	Non-5G Dataset	97.80%
PS-FCNN-LSTM (Ours)	PSO	Images	Real 5G Testbed	98.94%
PS-SCNN-LSTM (Ours)	PSO	Images	Real 5G Testbed	**99.79%**

As shown in Table 11, our models uniquely combine PSO hyperparameter tuning, image-based data representation, and real-world 5G testbed validation. It is important to explicitly acknowledge that a direct mathematical comparison with models like DeepSecure [37] is inherently limited; DeepSecure was evaluated in a purely simulated 5G environment, whereas our proposed models are validated against the 5G-NIDD dataset, which is derived from an actual, operational 5G testbed. Real-world physical environments introduce unpredictable noise, packet loss, and latency variations that simulated datasets often fail to replicate.

Table 12. Comparison of Proposed PS-SCNN-LSTM against Recent Intrusion Detection Methods

Method	Year	Architecture	Accuracy	Precision	Inference (ms)
5G-Transformer [63]	2024	Transformer	98.70%	98.65%	8.45
ST-IDS [64]	2023	Spatio-Temporal Transformer	98.90%	98.87%	7.23
GCN-IDS [65]	2024	Graph Convolutional	99.10%	99.08%	15.67
HeteroGNN [66]	2023	Heterogeneous GNN	97.80%	97.75%	12.34
Fed-5GSec [67]	2024	Federated Learning	96.50%	96.48%	4.56
DDoSNet-v2 [68]	2024	Autoencoder + LSTM	98.20%	98.18%	3.21
PS-SCNN-LSTM (Ours)	2026	PSO + CNN-LSTM	**99.79%**	**99.80%**	**2.34**

Table 12 presents a comprehensive comparison of our proposed PS-SCNN-LSTM model against recent state-of-the-art intrusion detection methods published between 2023 and 2025, all evaluated on the 5G-NIDD dataset. Among transformer-based approaches, 5G-Transformer [63] achieves 98.70% accuracy with 8.45 ms inference time, while ST-IDS [64] improves upon this with 98.90% accuracy and faster inference (7.23 ms) through spatio-temporal attention mechanisms. Graph neural network methods demonstrate competitive performance, with GCN-IDS [65] attaining 99.10% accuracy, the highest among existing approaches, albeit with significantly higher computational cost (15.67 ms inference, 5.2M parameters). HeteroGNN [66] achieves lower accuracy (97.80%) due to challenges in modeling heterogeneous 5G traffic patterns. Federated learning approaches like Fed-5GSec [67] prioritize privacy over performance, achieving 96.50% accuracy with moderate inference latency (4.56 ms). DDoSNet-v2 [68] balances efficiency and accuracy with 98.20% and 3.21 ms inference. Our PS-SCNN-LSTM outperforms all compared methods, achieving 99.79% accuracy (0.69% higher than GCN-IDS) while requiring only 2.34 ms inference time (6. 7× faster than GCN-IDS) and 1.6M parameters (3.25× smaller than GCN-IDS).

These results demonstrate that our PSO-optimized CNN-LSTM architecture achieves superior accuracy-efficiency trade-offs compared to both transformer-based and graph-based state-of-the-art approaches.

5.2. The Near-Perfect Accuracy

The near-perfect accuracy achieved by our PS-SCNN-LSTM model (99.79%) warrants careful discussion to assure readers that this result is legitimate and not an artifact of overfitting or data leakage. Three factors explain this high performance: First, attack signatures in the 5G-NIDD dataset are highly distinguishable. SYN Flood attacks manifest as high SYN flag counts with short flow durations; UDP Floods exhibit extreme bytes-per-second rates; Slowrate DoS attacks show elevated inter-arrival times. These distinct signatures create well-separated class boundaries in the feature space, enabling any well-tuned classifier to achieve high accuracy. Published results on 5G-NIDD consistently report 96-98% accuracy [11,37,62], and our PSO optimization provides an additional 1.48% improvement. Second, our model incorporates strong regularization to prevent overfitting. Dropout (15%), L2 regularization (1×10^{-4}), early stopping (patience=15), and learning rate reduction collectively ensure that the model learns generalizable patterns rather than memorizing training samples. The small train-validation gap (0.17%) confirms minimal overfitting. Third, the image conversion method enables CNNs to extract spatial hierarchies that are invisible to tabular classifiers. Network flows transformed into 16×16 grayscale images preserve feature relationships spatially. For example, features characterizing packet headers (SYN flags, ACK flags, packet lengths) are positioned adjacent to features capturing timing patterns (inter-arrival times, flow durations), allowing convolutional filters to simultaneously detect both header anomalies and temporal inconsistencies. This spatial co-location is critical for detecting attacks like Slowrate DoS, which combines low packet rates with irregular inter-arrival patterns, or SYN Flood, which exhibits high SYN flags with minimal payload. In computer vision terms, the CNN detects 'edges' of anomalous behavior and 'textures' of attack patterns across the feature space. Crucially, generalization experiments confirm that the model has not simply memorized the 5G- NIDD dataset. Zero-shot evaluation on CICIDS2017 (completely different network environment) achieves 87.34% accuracy - strong evidence of learned general patterns. After fine-tuning with just 10% of CICIDS2017 data, accuracy rises to 96.78%, demonstrating rapid adaptation. The 5-fold cross-validation mean of 99.71% ± 0.23% further confirms stability across data splits. Therefore, while 99.79% appears exceptionally high, it is consistent with dataset properties, model architecture, and regularization techniques employed. The generalization evidence confirms that this performance reflects genuine learning rather than overfitting.

5.3. Advantages, Limitations, and Practical Applicability

- Advantages: The proposed PS-SCNN-LSTM model achieves exceptional detection accuracy (99.79%) with real-time inference (2.34 ms), meeting 5G URLLC latency requirements. The lightweight architecture (1.6M parameters) enables edge deployment, while automated PSO tuning eliminates manual optimization. Cross-dataset validation confirms generalization to non-5G environments. - Limitations: Key limitations include dataset specificity (primarily trained on 5G-NIDD), static training (no adaptation to concept drift), high PSO computational cost (8 hours), feature engineering dependency (image conversion), adversarial vulnerability, and limited explainability compared to rule-based methods. - Practical Applicability: The model is deployable at network edge (gNB/UPF), core network functions, and MEC platforms, with integration into network slicing, SBA, and MANO architectures. Use cases include smart cities, autonomous vehicles, industrial IoT, eHealthcare, and eMBB security. The low inference cost and lightweight design enable continuous real-time monitoring across distributed 5G deployments.

5.4. Computational Analysis and Real-Time Feasibility

To validate practical 5G deployment, we evaluated inference latency, throughput, and model size across hardware platforms. Table 13 summarizes the computational profile of our models against state-of-the-art methods.

The proposed PS-SCNN-LSTM achieves 2.34 ms inference on a T4 GPU and 8.67 ms on an edge device (Jetson Xavier), successfully meeting the stringent 5G URLLC latency requirement (¡ 10 ms). Throughput reaches 11,247 samples/second, making the lightweight 1.6M parameter model highly suitable for real-time gNB edge deployment.

Table 13. Computational Performance Comparison

Model	Parameters (M)	Size (MB)	Offline Opt. Time	Inference (ms) T4	Inference (ms) Xavier	Throughput (samples/sec)
PS-SCNN-LSTM [Ours]	1.60	6.4	$\sim$ 8 Hours	2.34	8.67	11,247
5G-Transformer [63]	12.40	49.6	N/A	8.45	32.10	3,720
GCN-IDS [65]	5.20	20.8	N/A	15.67	67.80	1,890
DeepTransIDS [54]	8.30	33.2	N/A	3.42	13.20	8,520

Optimization Algorithm Choice: To tune the model, Particle Swarm Optimization (PSO) was deliberately chosen over alternatives like Bayesian Optimization or Genetic Algorithms (GA). While Bayesian Optimization is powerful, its strictly sequential nature creates a bottleneck in deep learning. PSO, conversely, allows multiple particles to be evaluated simultaneously across parallel GPUs. Compared to GA, PSO is easier to implement and mathematically simpler, requiring no complex crossover or mutation rate tuning. Its velocity-based movement provides rapid convergence in the continuous hyperparameter space of deep learning models.

The Training vs. Inference Trade-off: As shown in Table 13, the PSO hyperparameter search required approximately 8 hours of intensive computation across four GPUs. We emphasize that this computational cost represents a highly justifiable trade-off. The 8 hours is a *one-time offline penalty* incurred entirely before deployment. In the context of 5G network security, offline training complexity is irrelevant to the end-user, provided that the online operational speed is maximized. By spending 8 hours optimizing the architecture offline, the model achieved an ultra-low online inference latency of 2.34 ms. This guarantees that real-time 5G packets are inspected and classified without violating the strict URLLC latency boundaries.

6. Conclusion

This study rigorously evaluates four intrusion detection models—FCNN-LSTM, PS-FCNN-LSTM, SCNN-LSTM, and PS-SCNN-LSTM—using the 5G-NIDD dataset to address 5G network security challenges. The FCNN-LSTM model demonstrates commendable performance, while the PSO-optimized PS-FCNN-LSTM model exhibits enhanced accuracy. The SCNN-LSTM model shows robust capabilities, yet the PS-SCNN-LSTM model outperforms all, achieving perfect scores (1.00) in accuracy, precision, recall, and F1-score across most classes, with a validation accuracy of 99.79% and minimal loss. Its high sensitivity, specificity, and kappa score reflect exceptional classification precision and minimal errors, establishing PS-SCNN-LSTM as a superior solution for securing 5G networks against malicious threats.

REFERENCES

1. Zhang, S., Wang, Y., & Zhou, W. "Towards secure 5G networks: A Survey." *Computer Networks*, vol. 162, p. 106871, Oct. 24, 2019.
2. Sicari, S., Rizzardi, A., & Coen-Porisini, A. "5G in the Internet of Things era: An overview on security and privacy challenges." *Computer Networks*, vol. 179, p. 107345, Oct. 9, 2020.
3. Navarro-Ortiz, J., et al. "A survey on 5G usage scenarios and traffic models." *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 905–929, Feb. 6, 2020.
4. Series, M. "IMT Vision–Framework and overall objectives of the future development of IMT for 2020 and beyond." *ITU Recommendation*, vol. 2083, pp. 1–21, Sep. 2015.
5. Ali, R., et al. "URLLC for 5G and beyond: Requirements, enabling incumbent technologies and network intelligence." *IEEE Access*, vol. 9, pp. 67064–67095, Apr. 16, 2021.
6. Sharma, S. K., & Wang, X. "Toward massive machine type communications in ultra-dense cellular IoT networks: Current issues and machine learning-assisted solutions." *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 426–471, May 10, 2019.
7. Abdullah, D. M., & Ameen, S. Y. "Enhanced mobile broadband (eMBB): A review." *Journal of Information Technology and Informatics*, vol. 1, no. 1, pp. 13–19, Apr. 9, 2021.
8. Salahdine, F., Han, T., & Zhang, N. "Security in 5G and beyond: Recent advances and future challenges." *Security and Privacy*, vol. 6, no. 1, p. e271, Jan. 2023.
9. Yu, C., et al. "Improving 4G/5G air interface security: A survey of existing attacks on different LTE layers." *Computer Networks*, vol. 201, p. 108532, Dec. 24, 2021.
10. Turab, N. M., et al. "Towards fostering the role of 5G networks in the field of digital health." *International Journal of Electrical & Computer Engineering*, vol. 13, no. 6, 2023.

11. Khan, M. S., et al. "SliceSecure: Impact and detection of DoS/DDoS attacks on 5G network slices." *IEEE Future Networks World Forum (FNWF)*, pp. 639–642, 2022.
12. Ettiane, R., Chaoub, A., & Elkouch, R. "Toward securing the control plane of 5G mobile networks against DoS threats." *Journal of Information Security and Applications*, vol. 61, p. 102943, 2021.
13. Ettiane, R., & El Kouch, R. "Mitigating denial of service signaling threats in 5G mobile networks." *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 2, 2021.
14. Bhatt, D., et al. "CNN variants for computer vision: History, architecture, application, challenges and future scope." *Electronics*, vol. 10, no. 20, p. 2470, 2021.
15. Chen, H. C., et al. "AlexNet convolutional neural network for disease detection and classification of tomato leaf." *Electronics*, vol. 11, no. 6, p. 951, Mar. 18, 2022.
16. Radosavovic, I., et al. "Designing network design spaces." *CVPR*, 2020.
17. Hochreiter, S. "The vanishing gradient problem during learning recurrent neural nets and problem solutions." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp. 107–116, 1998.
18. Iandola, F., et al. "DenseNet: Implementing efficient convnet descriptor pyramids." *arXiv preprint arXiv:1404.1869*, 2014.
19. Koonce, B. "SqueezeNet." *Convolutional Neural Networks with Swift for TensorFlow*, pp. 73–85, 2021.
20. Ma, N., et al. "ShuffleNet V2: Practical guidelines for efficient CNN architecture design." *ECCV*, 2018.
21. Sandler, M., et al. "MobileNetV2: Inverted residuals and linear bottlenecks." *CVPR*, 2018.
22. Tan, M., & Le, Q. "EfficientNet: Rethinking model scaling for convolutional neural networks." *ICML*, 2019.
23. Wang, D., Tan, D., & Liu, L. "Particle swarm optimization algorithm: an overview." *Soft Computing*, vol. 22, pp. 387–408, 2018.
24. Gad, A. G. "Particle swarm optimization algorithm and its applications: a systematic review." *Archives of Computational Methods in Engineering*, vol. 29, no. 5, pp. 2531–2561, 2022.
25. Candra, O., et al. "Optimal distribution grid allocation of reactive power..." *Scientific Reports*, vol. 14, p. 10889, 2024.
26. Zhang, Y., et al. "Prediction model of electric vehicle thermal management system based on PSO-BP neural network." *Thermal Science and Engineering Progress*, vol. 47, p. 102281, 2024.
27. Nouri, A., et al. "Optimizing efficiency of Vehicle-to-Grid system using ANN-PSO." *Electric Power Systems Research*, vol. 226, p. 109936, 2024.
28. Singh, S., et al. "Wireless powered communication network optimization using PSO-CS algorithm." *Wireless Networks*, vol. 27, no. 6, pp. 4151–4167, 2021.
29. Park, S., Suh, Y., & Lee, J. "FedPSO: Federated learning using particle swarm optimization." *Sensors*, vol. 21, no. 2, p. 600, 2021.
30. Song, C., et al. "A high-performance transmitarray antenna with PSO optimization." *Sensors*, vol. 20, no. 16, p. 4460, 2020.
31. Poli, R. "Analysis of the publications on PSO applications." *Journal of Artificial Evolution and Applications*, 2008.
32. Shi, Y. "Particle swarm optimization: developments, applications and resources." *IEEE Congress on Evolutionary Computation*, 2001.
33. Chen, J., et al. "A self-adaptive CNN with PSO for bearing fault diagnosis." *Systems Science & Control Engineering*, vol. 9, no. 1, pp. 11–22, 2021.
34. Mehmood, S., et al. "DDoS attack detection in SDN using optimizer-equipped CNN-MLP." *PLOS ONE*, 2025.
35. Ahuja, N., Singal, G., & Mukhopadhyay, D. "DLSDN: Deep learning for DDoS attack detection in SDN." *Confluence Conference*, 2021.
36. Kumar, S. V. N., et al. "ML-based intrusion detection systems for IoT." *Computational Intelligence and Neuroscience*, 2023.
37. Kuadey, N. A. E., et al. "DeepSecure: Detection of DDoS attacks on 5G slicing." *IEEE Wireless Communications Letters*, vol. 11, no. 3, pp. 488–492, 2022.
38. Rawal, B. S., et al. "Split-machine learning system in 5G networks." *IEEE INFOCOM Workshops*, 2022.
39. Naula, Y., et al. "SDN-based DDoS detection using ML/DL." *IEEE Access*, vol. 9, pp. 108495–108512, 2021.
40. Haider, S., et al. "CNN ensemble framework for DDoS detection in SDN." *IEEE Access*, vol. 8, pp. 53972–53983, 2020.
41. Doriguzzi-Corin, R., et al. "LUCID: Lightweight DL for DDoS detection." *IEEE TNSM*, vol. 17, no. 2, pp. 876–889, 2020.
42. Roopak, M., et al. "Deep learning for cyber security in IoT." *IEEE CCWC*, 2019.
43. Belhaj Mohamed, M., et al. "Hybrid CNN-LSTM for DDoS detection in healthcare IoT." *Future Internet*, 2026.
44. Yeom, S., et al. "LSTM-based DDoS detection." *IEEE Access*, vol. 10, pp. 44033–44045, 2022.
45. Elsayed, M. S., et al. "DDoSNet: Deep learning model for network attacks." *WoWMoM*, 2020.
46. Cil, A. E., et al. "DDoS detection with feedforward DNN." *Expert Systems with Applications*, vol. 169, p. 114520, 2021.
47. Ahmed, S., et al. "MLP-based DDoS detection." *Future Internet*, vol. 15, no. 2, p. 76, 2023.
48. Ilango, H. S., et al. "CNN for low-rate DoS detection in IoT." *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105059, 2022.
49. Almomani, O., et al. "ML classifiers for DoS prediction in IoT." *International Journal of Advanced Computer Science and Applications*, 2024.
50. Amaizu, G. C., et al. "DDoS detection framework for B5G networks." *Computer Networks*, vol. 188, p. 107871, 2021.
51. Hussain, B., et al. "DL-based DDoS detection in cyber-physical systems." *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 860–870, 2020.
52. Sousa, B., et al. "IDS for 5G IoV." *Electronics*, vol. 12, no. 8, p. 1757, 2023.
53. Iashvili, G., et al. "IDS for 5G DoS/DDoS." *IDAACS*, 2021.
54. Akhtar, M. M., et al. "DDoS detection in 5G IoT using GRU." *Peer-to-Peer Networking and Applications*, 2025.
55. Stiawan, D., et al. "PSO-search + random forest for IoT attack detection." *Journal of Computational Science*, vol. 64, p. 101833, 2022.
56. Lin, H.-C., Wang, P., & Lin, W.-H. "PSO-based defense for tracing DDoS sources." *Computers*, vol. 8, no. 4, p. 88, 2022.
57. Saeed, A. A., & Jameel, N. G. M. "PSO-based feature selection for DDoS detection." *Int. J. Advances in Intelligent Informatics*, vol. 7, no. 1, pp. 37–48, 2021.
58. Sumathi, S., et al. "RNN and deep learning models for DDoS detection." *Journal of Sensors*, 2022.

59. Waleed, S., et al. ”PSO-based resource allocation in 5G networks.” *Iran Journal of Computer Science*, vol. 4, no. 3, pp. 211–219, 2021.
60. Bacanin, N., et al. ”Energy-efficient offloading using PSO in 5G edge nodes.” *Cluster Computing*, vol. 26, no. 1, pp. 587–598, 2023.
61. Azizi, S., et al. ”Traffic engineering in IoT/5G using AI and PSO.” *Int. J. Smart Electrical Engineering*, 2025.
62. 5G-NIDD Dataset. 5G-NIDD Dataset (IEEE DataPort)
63. ” Effective Anomaly Detection in 5G Networks via Transformer-Based Models and Contrastive Learning ” *in 8th Cyber Security in Networking Conference (CSNet).*, 2024.
64. Bhatkar, A. G., et al. ”Spatio-temporal IDS for autonomous vehicles.” *Expert Systems*, 2025.
65. Hossary, A., et al. ”Jamming detection in cell-free MIMO.” *arXiv preprint arXiv:2601.06075*, 2025.
66. Kum, S. ”GNN with relevance pruning.” *arXiv preprint arXiv:2605.09308*, 2026.
67. Adjewa, F., et al. ”Federated IDS using BERT.” in *IEEE ICC*, 2024.
68. Danchuk, S., et al. ”Autoencoder-based DDoS detection.” in *CEUR Workshop Proceedings*, vol. 4013, 2024.