Received: 30 September 2023 | Revised: 16 January 2024 | Accepted: 2 February 2024

IET Communications

DOI: 10.1049/cmu2.12735

ORIGINAL RESEARCH

Improved 5G network slicing for enhanced QoS against attack in SDN environment using deep learning

Mohammed Salah Abood[1] | **Hua Wang**[1] | **Bal S. Virdee**[2] | **Dongxuan He**[1] | **Maha Fathy**[1] | **Abdulganiyu Abdu Yusuf**[3] | **Omar Jamal**[4] | **Taha A. Elwi**[5,6] | **Mohammad Alibakhshikenari**[7] | **Lida Kouhalvandi**[8] | **Ashfaq Ahmad**[9]

[1]School of Information and Electronics, Beijing Institute of Technology, Beijing, China

[2]Center for Communications Technology, London Metropolitan University, London, UK

[3]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China

[4]College of Information Technology, University of Babylon, Babylon, Iraq

[5]Islamic University Centre for Scientific Research, The Islamic University, Najaf, Iraq

[6]International Applied and Theoretical Research Center (IATRC), Baghdad Quarter, Iraq

[7]Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Leganés, Madrid, Spain

[8]Department of Electrical and Electronics Engineering, Dogus University, Istanbul, Turkey

[9]Department of Chemistry, College of Science, King Saud University, Riyadh, Kingdom of Saudi Arabia

Correspondence
Mohammed Salah Abood and Hua Wang, School of Information and Electronics, Beijing Institute of Technology, Haidian District, Beijing, China.
Email: mohammedsalah@bit.edu.cn and wanghua@bit.edu.cn

Bal S. Virdee, Center for Communications Technology, London Metropolitan University, London N7 8DB, UK.
Email: b.virdee@londonmet.ac.uk

Mohammad Alibakhshikenari, Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Leganés, Madrid, Spain.
Email: mohammad.alibakhshikenari@uc3m.es

Abstract
Within the evolving landscape of fifth-generation (5G) wireless networks, the introduction of network-slicing protocols has become pivotal, enabling the accommodation of diverse application needs while fortifying defences against potential security breaches. This study endeavours to construct a comprehensive network-slicing model integrated with an attack detection system within the 5G framework. Leveraging software-defined networking (SDN) along with deep learning techniques, this approach seeks to fortify security measures while optimizing network performance. This undertaking introduces network slicing predicated on SDN with the OpenFlow protocol and Ryu control technology, complemented by a neural network model for attack detection using deep learning methodologies. Additionally, the proposed convolutional neural networks-long short-term memory approach demonstrates superiority over conventional ML algorithms, signifying its potential for real-time attack detection. Evaluation of the proposed system using a 5G dataset showcases an impressive accuracy of 99%, surpassing previous studies, and affirming the efficacy of the approach. Moreover, network slicing significantly enhances quality of service by segmenting services based on bandwidth. Future research will concentrate on real-world implementation, encompassing diverse dataset evaluations, and assessing the model's adaptability across varied scenarios.

1 | INTRODUCTION AND RELATED WORKS

Fifth-generation (5G) networks represent the latest advancements in mobile communication technologies as defined by the International Telecommunication Union. The inaugural commercial network debuted in Finland in June 2018, heralding a new era of ubiquitous connectivity and paving the way for yet unimaginable services. The 5G infrastructure offers accelerated transmission speeds, enhanced coverage, and facilitates the seamless operation of diverse applications, including remote functionalities such as surgery, manufacturing, and driving. This technological leap promises to transform the fabric of reality into a virtual realm.

Of particular interest to consumers is the heightened data transfer capacity of 5G coupled with reduced latencies. A higher data rate ensures the smoother operation of nearly all services compared to lower rates. For instance, the upcoming proliferation of 4K and 8K videos at high frame rates in cloud gaming hinges upon both robust data capabilities and low latencies. Lower latency not only enriches the mobile gaming experience but also unlocks an array of novel services that necessitate a synergy between low latency and ultra-reliability. In the first phase, 5G technology was concerned with providing high-speed data rates, which includes the three main categories described below [1].

1. Enhanced mobile broadband (eMBB) includes a high data rate and high traffic.
2. Ultra-reliable, low-latency communication (uRLLC) includes low latency, low error rate, and superior reliability.
3. Massive machine-type communication (mMTC) involves many connected devices and saves power [2].

Enhanced mobile broadband aims to provide almost fibre speeds over the air. It improves the radio path with new base stations and end devices (phones, mobile routers). URLLC and mMTC are both for machine-to-machine communications, and their benefits are more difficult to understand for large audiences [2]. The mMTC provides connections for many Internet of Thing (IoT) sensors, like temperature, humidity, pressure, movement, vibration, magnetic and location. All of that works with minimal power consumption and maximizes battery life. On the mobile network side, mMTC means that up to 1 million sensors can be located in one square kilometre. The URLLC is needed especially for autonomous driving and other solutions that need reliable and low-latency communication, like power grid control [3].

The concept of slicing was innovated and refined specifically to enable 5G to effectively cater to these diverse requirements. It involves the segmentation of the 5G network capacity, departing from a uniform 'best-effort' approach. This segmentation allows for distinct traffic categories, with some benefiting from higher transfer speeds while others experience reduced delays. Network slicing stands as a pivotal technique for delivering the aforementioned services in a resource-optimized manner within the mobile network infrastructure. Both the radio path and core network can undergo partitioning into multiple slices, each capable of possessing its unique characteristics and resource allocation. By implementing slicing, individualized capacities can be provided and assured, ensuring a tangible enhancement in the quality of service (QoS) for specific customers or types of traffic. Efficient QoS had not been available before 5G [4].

Slicing and low delays are logical to be bundled together, as many services requiring low delays also require guaranteed capacity. QoS. However, all three main categories, eMBB, mMTC, and uRLLC, are designed to be sliceable features, as they require different delays, power consumption, and throughput.

In addition, the slicing network provides additional network security and the ability to apply different types of security on each slice to detect attacks [5]. This paper applied the network slicing process using the OpenFlow protocol, SDN, and Ryu Control to segment and manage network traffic. Neural network techniques were also employed to detect and identify the types of attacks to which 5G networks are exposed.

Many works have proposed different techniques for slicing 5G networks and preventing cyber-attacks from improving network services. The concept of 5G network slicing is employed to manage network traffic and direct connections to the most suitable slice [6]. A proposed mathematical model can offer on-demand slice isolation and ensure minimal delay for 5G core network slices, proactively mitigating distributed denial-of-service attacks in the 5G core through slice isolation. The authors in [7] introduce novel provisioning models for third-party slices and discuss their isolation properties. Ref. [8] suggests an effective and secure service-oriented authentication that supports network slicing and fog computing for 5G-enabled IoT services. They also introduce a privacy-preserving slice selection mechanism to preserve both component slice types and access user service types. Ref. [9] employs the constrained application protocol (CoAP) and message queue telemetry transport (MQTT) application protocols to provide efficient mechanisms and methods for over-the-air (OTA) delivery of software updates and security patches to IoT devices. The authors also evaluate which protocol suits the proposed models and applications better. In [10], the authors provide a prototype of software chaining using network function virtualization (NFV), while [11] demonstrates information-centric networking using SDN for service chaining. However, all of these contributions provide specific use cases that may not make optimal use of underlying resources, and no details on the algorithms were provided without utilizing attack detection to improve QoS. The authors in [12] have implemented VNFs of cloud radio access networks (CRAN) and made them publicly available at the Juju Network Store, allowing the implementation of the first stage of resource management (resource selection). Nevertheless, this endeavor hinges on virtual machines, which exhibit higher memory consumption compared to containers. Furthermore, it lacks the provision of a controller block to trigger the collection of metrics on the performance of the NFVs for optimizing the slice of performance metrics concerning the network function virtualization (NFV), crucial for the optimization of slices.

Our system employs a dual approach: first, network slicing leveraging OpenFlow control within the software-defined networking (SDN) environment; second, the application of deep learning techniques to detect potential attacks on individual slices. This integration aims to enhance network services by fortifying security measures and refining operational efficiencies.

1.1 | Motivations

There are several motivations for introducing network slicing and attack detection, which can be deployed with a 5G network:

1. Network slicing represents a promising feature enabling business customers to procure customized connectivity aligned with their distinct operational needs within a shared infrastructure. For instance, while some businesses require high throughput, others prioritize extensive connectivity, low latency, or heightened reliability.
2. Beyond managing user traffic, the significance of 5G slicing lies in fortifying systems against potential attacks and fault occurrences, as any disruption or compromise typically impacts solely the targeted slice, safeguarding the broader network.
3. Our ongoing initiative involves the development of a model aimed at identifying various types of network attacks to which the system may be vulnerable. This endeavour employs neural networks in conjunction with network segmentation methodologies to bolster the network's security posture.

1.2 | Problem statement

Enhancing the network infrastructure to deliver three critical services—enhanced mobile broadband (eMBB), massive machine type communication (eMTC), and ultra-reliable low latency communication (uRLLC)—poses a significant challenge. This challenge revolves around the complex task of accommodating these diverse services on a unified physical network while concurrently upholding robust network security measures to counter potential attacks.

1.3 | Main objectives

The main goal of this study is to construct a network-slicing model integrated with an attack detection system designed to identify and mitigate potential threats within the 5G network. The specific objectives include: locating a suitable 5G data collector responsible for gathering data from segmented slices of the 5G network, identifying an appropriate neural network classifier, establishing a 5G network using software-defined networking (SDN), and developing network slices governed by OpenFlow control.

1.4 | Contributions

The main contributions of the proposed system are:

1. Create 5G network slices based on SDN, Ryu Control, and OpenFlow protocol.
2. Detect attacks based on the neural network model by building a classifier model that classifies the packets as attack or not, using deep learning.
3. Build 5G network core, relying on the SDN environment to represent a 5G core network.
4. Employ OpenFlow protocol to separate the data plane from control for dividing the network into slices with different bands and services.

We will organize the remainder of this paper as follows: Section 2 gives theoretical background about SDN architecture, the challenges faced in SDN, its fundamental characteristics, OpenFlow protocol specification, and the parameters used in SDN Plus, slicing network technique, machine learning technique, and deep learning convolutional neural networks (CNN) algorithm. We describe the proposed system for optimal 5G network slicing to improve QoS against attacks in the SDN environment illustrated in Section 3. Implementation results and evaluation that describe the results of the proposed system and then the implementation of the model evaluated are elaborated in Section 4. Finally, we will conclude the paper and suggest some future work to be carried out in Section 5.

2 | THEORETICAL BACKGROUND

Network slicing is a technique used in modern computer networks to create multiple unique/distinct logical and virtualized networks over a common multi-domain infrastructure, including access, core, and transport, and can be deployed across multiple operators [13]. The utilization of network slicing is prevalent within the framework of 5G wireless networks, facilitating operators to tailor network services according to distinct user and application requirements, encompassing diverse needs in throughput, latency, and reliability. Not only does it involve securing networks against potential attacks, but it also necessitates employing numerous techniques, as emphasized in our research. Leveraging software-defined networking (SDN), analytics, and automation, mobile network operators (MNOs) can efficiently generate network slices designed to accommodate specific applications, services, user sets, or network functionalities.

2.1 | Software-defined networks

The term "software-defined networking" (SDN) denotes a network design and management approach that segregates the control plane, responsible for managing network traffic and selecting routes [14], from the data plane, handling actual data packet transmission. Within SDN, network administration is centralized through a software-based controller, enabling dynamic resource configuration to meet user and application requirements. This centralization streamlines network oversight, facilitating enhanced monitoring, traffic optimization, and rapid responses to evolving traffic patterns and security threats. With its inherent advantages in delivering more adaptable, scalable,

TABLE 1 A comparison between traditional and SDN networks.

NO.	Criteria	Traditional network	SDN
1.	Network management	Difficult	Easy
2.	Global network view	Difficult	Central view at the controller
3.	Maintenance cost	Higher	Less
4.	Time required for update/error handling	Long time	Short time
5.	Controller utilization	Not relevant	Important
6.	Controller availability	Not relevant	Important

Abbreviation: SDN, software-defined networking.

and efficient network governance, SDN has experienced escalating adoption, notably within cloud computing and data centre environments, in recent years. SDN can create and control a virtual network or traditional hardware via software.

So-called "programmable networks" are introduced to facilitate current networks to evolve. This concept was stated a few years ago. Previously, the development of various forms of technology-enabled the communication networking to be programmed. The SANE/Ethane project introduced a novel architecture for enterprise networks. The main focus lies in using centralized controllers to manage policies and security within a network. Therefore, this project is considered the origin of what is today known as SDNs. Table 1 draws a comparison between the traditional and SDN networks [15, 16].

Initially conceived to address diverse networking challenges, SDNs are built upon the foundational concept of segregating control and data planes. Within SDNs, four fundamental principles typically underpin their architecture:

1. *Separation of control and data plane*: A logical separation should be made between these two planes, after which an interface is used to connect them. External entities obtain the controlling aspect instead of forwarding devices.
2. *Network programmability*: A basic concept in the structure of SDNs is the open API, as there should be easy access, configuration, and modification of networking elements through software and scripts.
3. *Network abstraction*: The network can be virtually seen by hierarchically higher positioned elements so that the service or application has an overall awareness of the network. However, any physical attribution is irrelevant to the configuration or computation.
4. *Logically centralized control*: Each forwarding device has a link to a controlling one and has to function according to its policy.

In SDNs, the controlling and forwarding functions are separated within the networks, which enable the direct programming of the network control. Separating the control and data planes is fundamental in the structure of SDNs. Through this process, the switch becomes simply a forwarding device, whereas the controller implements the control logic while being physically or logically centralized. The latter form of controller delineates network behaviour and boasts several advantages. This mode proves straightforward and less prone to errors when altering network policies using software, particularly when executed from a single location without device reconfiguration. Secondly, higher-level policies can be upheld through control programs that autonomously adapt to dynamic network changes. Thirdly, developing appropriate network functionalities becomes more adaptable and uncomplicated due to the central control logic's comprehensive understanding of the network (topology and resource states). Controllers could adjust the flow table dynamically to avoid congestion. Controllers possess the capability to dynamically adjust flow tables, avert congestion or apply alternate routing algorithms to manage traffic. Consequently, a pivotal value proposition of SDNs lies in their capacity to program networks for controlling the underlying data plane. The benefits of SDN have been underscored across diverse scenarios, spanning enterprises and data centres, with notable illustrations such as Google B4 [17]. As shown in Figure 1, the division of the SDN architecture consists of three layers: the infrastructure, control, and application layers.

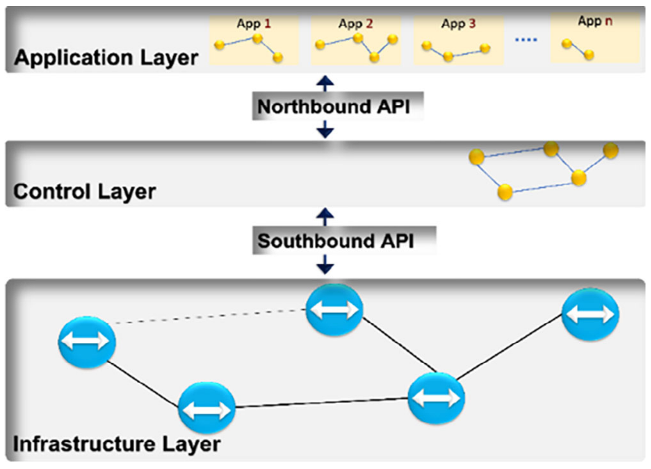


FIGURE 1 The three-layer architecture in software-defined networking.

2.2 | OpenFlow protocol

The OpenFlow protocol [18] is an initially standardized protocol for SDNs. Although there are many alternatives, such as extensible messaging and presence protocol (EMPP) [19], Open vSwitch Database (OVSDB) [20, 21], and OpFlex [22], it has

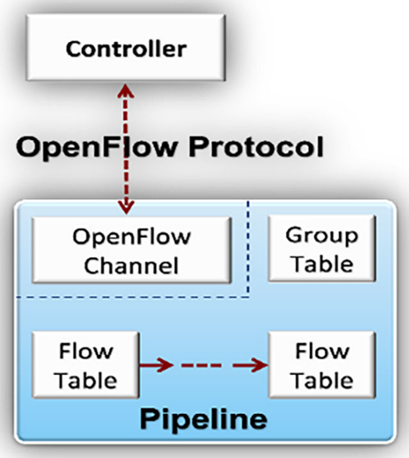


FIGURE 2 The OpenFlow switch.

been set as the standard, one and multiple companies support it in their SDN solutions. This protocol was introduced by McKeown et al. to enable an easier networking experiment within campus networks [18]. There is more than one version of the protocol specifications available, the most common of which is 1.0 (released on 31st December 2009). Others include 1.1 [23], 1.2 [24], 1.3 [25], 1.4 [26], and 1.5 [27] (and are accessible via the direct URL links provided below). The details of the changes between one version and another are provided in the OpenFlow 1.5.1 specification document [28]. The Open Flow is an open protocol that enables the programming of switch flow tables by a software application. It comprises three items: the OpenFlow-compliant switch, controller, and channel. The switch uses the flow table for forwarding the packet. Flow tables are lists of flow entries, with each a match field, a counter, and a set of instructions. In SDNs, the Open Flow switch simply forwards received packets. Figure 2 shows how the OpenFlow Switch has several flow tables and one group table for the packet lookup and forwards.

- OpenFlow Switch Specification 1.1.0, [Online]. Available: https://opennetworking.org/wpcontent/uploads/2013/02/of-config-1.1.pdf
- OpenFlow Switch Specification 1.2.0, [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.2.pdf
- OpenFlow Switch Specification 1.3.0, [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf
- Open Networking Foundation, [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf
- Open Networking Foundation, [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.0.pdf
- Open Networking Foundation, [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf

There is also an OpenFlow channel for external controllers. The switches' flow table consists of more than one flow entry or main field (match fields, counters, instructions) that determines how packets' processing and forwarding will occur.

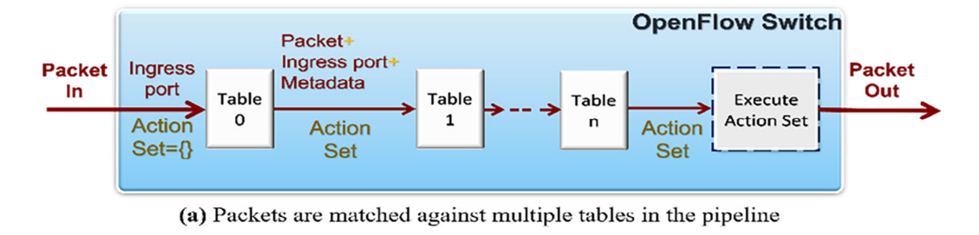
1. *Match fields*: It is where the received packets are matched. They include information from the 15-tuple packet's headers like (ingress port, metadata, Ethernet Src, Ethernet Dest, Ethernet type, virtual local area network (VLAN) ID, VLAN priority, multiprotocol label switching (MPLS) label, MPLS traffic class, IP Src, IP Dest, IP Proto, IP type of service (TOS) field, Transport Src. Port, Transport Dest).
2. *Counters*: It collects statistical information about a certain flow, including how many packets come in, the number of bytes, and flow duration.
3. *Instructions*: These should be executed when a packet is matched. They determine how the packet is handled (forwarded or dropped).

Match fields present a description of the packets that the entries are associated with. This involves ingress ports and certain header fields, like IP and Mac addresses. The network administrators set them through the controller, using either a particular value or a wild card matching any flow.

Figure 3 shows how packets are processed through the pipeline. First, the metadata field and the action set get an empty initialization from the first table (Table 0) to the last (Table n). The packets are made into matches to the consecutive flow tables, where the entry with a higher priority is chosen. The end of the pipeline is reached whenever no packet is matched or no "Goto" instruction is set. There are three processes for all flow tables: First is matching the packets to the high-priority entries, next is applying the instruction to the packet, and finally, the packet is transferred to the following table for any further acts.

2.3 | Machine learning

Machine learning (ML) is the process of learning machines to treat the data quickly and efficiently to obtain maximal productivity with minimum resources. In other words, ML is a research domain of computer science related to the invention of effective pattern realization techniques capable of adapting to the dynamic changes of problems in size and type. ML algorithms are applied in many computing processes where straightforward algorithms are insufficient. Several applications were introduced depending on the ML concepts, such as attack classification, spam filtering and search engines [29]. We can define the classification technique as identifying to which category or class the new data belongs. In other words, the classification task maps items in the dataset into one of a predefined group of categories. This process is performed in two steps: building a classification model depending on a set of training features and their related classes. This model predicts the class



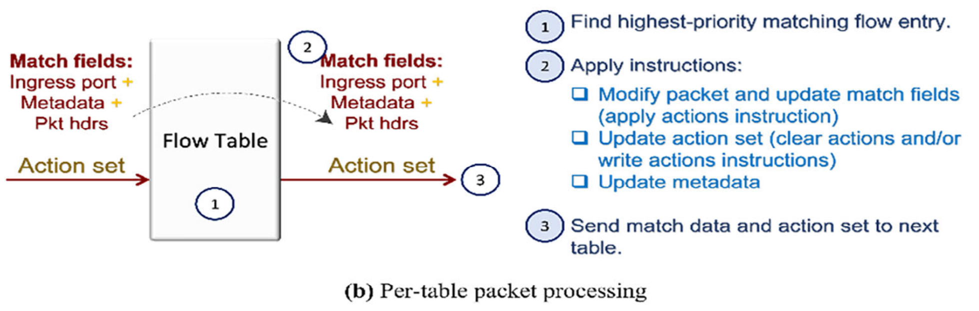

Match fields: Ingress port + Metadata + Pkt hdrs
Action set
Flow Table
Match fields: Ingress port + Metadata + Pkt hdrs
Action set
1 Find highest-priority matching flow entry.
2 Apply instructions:
Modify packet and update match fields (apply actions instruction)
Update action set (clear actions and/or write actions instructions)
Update metadata
3 Send match data and action set to next table.
(b) Per-table packet processing

FIGURE 3 OpenFlow packet processing pipeline.

of unclassified data [30]. The classification technique is applied in many industries and fields, such as attack, spam and medical text classification, using several algorithms such as support vector machine [31–33], decision tree, Random Forest [34], and Naïve Bayes [35]. Historically, many approaches have been used for classification, including Naïve Bayes, decision trees, artificial neural networks, Random Forests, and association rules.

- *Decision tree (DT)*: Among the prevalent classification methods, the decision tree stands as a widely utilized approach. It operates by organizing features based on their values. The tree structure encompasses three node types: the root node, internal nodes, and leaf nodes [36]. Positioned at the apex, the root node lacks incoming edges; internal nodes possess a single incoming edge and one or more outgoing edges. Conversely, leaf nodes, for text classification, possess one incoming edge but no outgoing edges. Supervised machine learning employing decision trees involves iterative division of data into increasingly smaller segments, constructing hierarchical decision trees extensively used in classification tasks. Decision trees employ if-else rules, effectively abstracting data to streamline decision-making processes [37]. In document classification, the structure of the decision tree hinges upon a defined set of rules derived from extracted document features. These rules dictate the decision-making process, enabling effective classification based on document attributes.
- *Random Forest*: Employing a multitude of decision trees during its preparatory phase, the random forest methodology swiftly generates a diverse set of outcomes within machine learning. It serves as an effective countermeasure against the tendency of decision trees to overfit their training datasets. This simulation algorithm applies the widely-used bootstrap aggregation strategy to train tree learners. Instead of solely relying on a training set $X = x1,\ldots xn$ with corresponding $Y = y1,\ldots yn$, frequently bagging ($\boldsymbol{B}$ times) picks a random sample to substitute the training set and matches trees for $b = 1,\ldots,\boldsymbol{B}$ to such samples. Examples, with substitution, $\boldsymbol{n}$ illustration of X, Y training; name these $\boldsymbol{Xb}$, $\boldsymbol{Yb}$. Train a tree $\boldsymbol{fb}$ on $\boldsymbol{Xb}$, for classification or regression. Predictions for unknown samples $\boldsymbol{x'}$can be produced after training.

$$f = \frac{1}{B}\sum fb(x')Bb = 1 \tag{1}$$

where $\hat{f}$is predictions from each tree, $\boldsymbol{B}$ is repeatedly bagging, b is sampling for ($b = 1,\ldots,$), $\boldsymbol{fb}$ is regression tree, and x' is unseen [38].

- *Convolutional neural networks*: Deep learning techniques are based on neural networks, a subset of machine learning (ML). They are made up of node layers that have input layers, hidden layers, and output layers. Each node has a threshold and a weight that are connected. An individual node is activated and sends data to the network's next layer if its output exceeds the predefined threshold value. Otherwise, no data is passed to the next network layer [39]. CNNs excel in hierarchically extracting features from data through convolution, gradually interpreting complex patterns. They generate feature maps highlighting specific data aspects and utilize pooling layers to condense these features, enhancing pattern recognition. With specialized layers and functions, CNNs are versatile tools applicable not only in image analysis but also in diverse fields like natural language processing, medical imaging, and autonomous vehicles [40].

Various neural nets are used for different use cases and data types. Recurrent neural networks, for instance, are frequently employed in voice and natural language processing. In contrast, convolutional neural networks (ConvNets or CNNs) are often used for classification and computer vision tasks. Convolutional neural networks (CNN) now provide a more scalable approach to classification and object recognition tasks, leveraging principles from linear algebra, specifically matrix multiplication, used for the classification object. In network slicing based on 5G, how does CNN work?

Convolutional neural networks (CNNs) are a type of neural network commonly used in image and video processing applications. In the context of network slicing based on 5G, CNNs can be used for tasks such as image recognition, object detection, and video processing, which are important components of many emerging 5G use cases, such as autonomous vehicles, smart cities, and remote healthcare. CNN is distinguished from other neural networks by its superior performance. They have three main types of layers, which are:

- *Convolutional layer*: The convolutional layer is the core building block of a CNN, where most computation occurs. It requires a few components: input data, a filter, and a feature map. We also have a feature detector, also known as a kernel or a filter, which will move across the features, checking if the feature is present. This process is known as a convolution. The structure of the CNN can become hierarchical when another convolution layer follows the initial convolution layer [41, 42].
- *Pooling layer*: Pooling layers, also known as down sampling, conduct dimensionality reduction, reducing the number of parameters in the input. Like the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter has no weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array. Pooling can be divided into two categories:
  - Max pooling: As the filter moves across the input, it selects the feature with the maximum value to send to the output array. That is why this method is frequently preferred over average pooling.
  - Average pooling: As the filter moves across the input, it calculates the average value within the features to send to the output array.
- *Fully-connected layer*: The full-connected layer's name accurately depicts its function. In partially connected layers, the feature values of the input features are not directly connected to the output layer. In the fully-connected layer, however, each node in the output layer connects directly to a node in the previous layer. This layer performs categorization based on the features retrieved by the previous layers and their various filters. While convolutional and pooling layers typically utilize ReLu functions to classify inputs, FC layers typically use a SoftMax activation function to produce a probability ranging from 0 to 1 [43].

2.4 | Hyperparameter optimization

Hyperparameter tuning, statistically speaking, takes a snapshot of a model's current performance and compares it to past snapshots. All hyperparameters must be set to their default values to start an ML algorithm training. When the model hyperparameters are fine-tuned, the model's performance on the validation set is maximized. Hyperparameters are parameters that are established before the learning process begins. In contrast, the values of model parameters are determined from the data via training. Model parameters relate to the weights and coefficients the algorithm produces using the data. Each algorithm has its own set of hyperparameters, such as the depth parameter for a decision tree. Hyperparameters cannot be modified manually to get optimal model performance. To find the optimal configuration, automatic optimization is required. This optimization is done through processes such as random search or Bayesian optimization. In contrast, the parameters of parametric models are calculated while fitting them to the data [44].

Because hyperparameters may directly influence the training algorithm's behaviour, they are considered important. Choosing the right hyperparameters has a significant impact on the model's performance. Separating the data into three sets (training, testing, and validation sets) is crucial to update the default parameters to ensure that the required accuracy is acquired to avoid data leaks [45].

2.5 | Data standardization

Data standardization is a preprocessing stage, a scaling procedure, or a mapping technique. Where can we locate a new set from an established one? It can be incredibly useful for predicting or forecasting purposes. As we all know, there are various approaches for projecting or forecasting, but they all differ greatly. The standardization technique is needed to get them closer together to retain the wide variance in prediction and forecasting. However, standardization methods such as z-score, max–min, decimal normalization, and numerosity reduction normalization are still used [46, 47]. Z-score normalization [48] is a method that generates standardized values or samples of data from unstructured data by utilizing terms such as mean and standard deviation. So, as seen in Equation (2) [46], the data can be normalized using the z-score parameter:

$$v_i' = \frac{(v - \mu)}{\sigma} \tag{2}$$

v_i' is a normalized value, v is the original value, μ is the mean of data, and σ is the standard deviation of the data.

2.6 | Evaluation measures

The NIDS efficiency evaluation involved several features defined in the confusion matrix [49, 50].

- *True positives (TP)*: It means correct values in actual and prediction for positive values, which means actual value yes and predicate value yes.

- *True negatives (TN)*: It means correct values in actual and prediction for negative values, which means the actual value is no and the predicate value is no.
- *False positives (FP)*: It means incorrect values in actual and prediction for positive values, which means the actual value is no and the predicate value is yes.
- *False negatives (FN)*: It means incorrect values in actual and prediction for positive values, which means the actual value is no and the predicate value is yes.

Each of the accuracy (ACC), precision (P), recall (R) and F1-measure ($F1$) metrics is used in evaluating [51]. These features are obtained in the following way:

- Accuracy (ACC): It shows the percentage of true detection over the total traffic trace:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{3}$$

- Precision (P): It shows how many intrusions predicted by a NIDS are actual intrusions. The higher P then, the lower the false alarm is:

$$P = \frac{TP}{TP + FP} \times 100\% \tag{4}$$

- Recall (R): It shows the percentage of predicted intrusions versus all intrusions presented. The higher the R-value, the better.

$$R = \frac{TP}{TP + FN} \times 100\% \tag{5}$$

- $F1$-measure ($F1$): It gives a better measure of the accuracy of a NIDS by considering both the precision (P) and the recall (R). A higher $F1$ value is more favourable.

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} \times 100\% \tag{6}$$

- False positive rate (FPR) against true positive rate (TPR): The FPR and TPR are calculated as follows [52]:

$$FPR = \frac{FP}{FP + TN} \times 100\% \tag{7}$$

$$TPR = \frac{TP}{FP + TN} \times 100\% \tag{8}$$

3 | THE PROPOSED SYSTEM

The proposed system improves QoS and security in 5G networks by slicing the network into slices according to the type of service provided within the SDN environment and detecting the attacks that these slices are exposed to. This system consists of five main stages: Build an attack detector model, build a 5G network core, create a 5G slice network, collect slice network data, and detect an attack on slice using the attack detector model. The general framework for improving service quality and security in 5G networks includes five main pillars, as shown in Figure 4.

- Collect 5G data from the UNSW_NB15 Dataset and NSL_KDD dataset.
- Using the collected data, build an attack detector model depending on CNN. To discover the types of attacks that the network can be exposed to.
- Build 5G network core, relying on the SDN environment to represent a 5G network.
- Create 5G network slices using the OpenFlow protocol designed to separate the data plane from control for dividing the network into slices with different bands and services.
- Detect the attack on each slice by using the attack detector model.

3.1 | 5G dataset

The dataset used in this research was collected from UNSW-NB15, a network intrusion dataset. It contains nine attacks: Backdoors, Fuzzers, DoS, and worms. The dataset contains raw network packets. The number of records in the training set is 175,341 records, and the testing set is 82,332 records from the different types, attack and normal [53]. *URL*: https://research.unsw.edu.au/projects/unsw-nb15-dataset

3.2 | Build an attack detector model

Detection of attacks in a 5G network by analysing the packets and classifying them into normal and abnormal. The deep learning algorithm is relied upon in classification because it is considered more efficient in higher dimensional areas and is used in situations with more dimensions than samples. Figure 5 illustrates the design of the system that has been suggested. This section presents (offline phase), that is, the proposed system training and the stages used to detect the attacks present in the data sets used in the training of the attack model.

3.2.1 | Data preprocessing

Deep learning algorithms do not function too well for raw data processing. Therefore, the data must be pre-processed before being fed to a CNN algorithm. In other words, some transformations must be applied, as shown in Table 2. The raw data is transferred into a clean, pre-processed data collection. Some ML models require information in a specified format. Where INT stands for initialization or initiation, representing the protocol state value for a particular flag. It encompasses four distinct cases, indicating whether it signifies initialization,

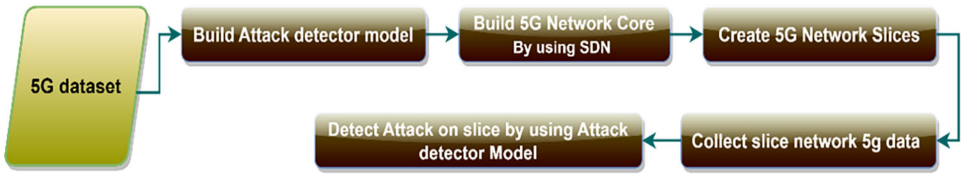

5G dataset
Build Attack detector model
Build 5G Network Core By using SDN
Create 5G Network Slices
Collect slice network 5g data
Detect Attack on slice by using Attack detector Model

FIGURE 4 General framework.

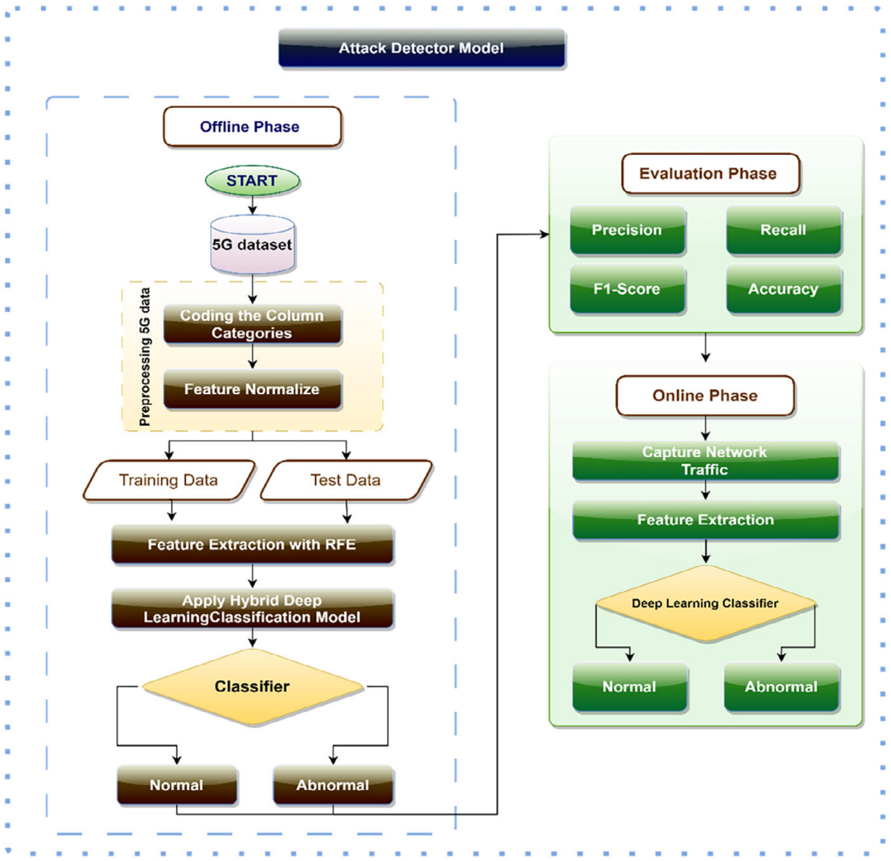


FIGURE 5 Attack detector model.

TABLE 2 Distribution of categories in features.

No	Protocol type	Service	Flag
1	Ddp	–	INT
2	ipv6-frag	–	INT
3	Cftp	–	INT
4	Wsn	–	INT
5	Tcp	ftp	INT
6	Tcp	–	INT
7	Pvp	–	INT

finishing, connection, or reconnection. In our case it is only initiation (INT) as in Tables 2 and 3.

- Coding the features categories: A transformation has been applied to convert the categorical values into numerical ones in order to apply to the induction model, as shown in Table 3.

3.2.2 | Feature extraction with RFE

Feature selection is a principal concept in ML that significantly affects the model performance. Inappropriate or partially related features may harm the model's efficiency, particularly in attack detection. Therefore, data cleaning and feature selection are essential steps in the model design. Feature selection is the process by which certain features are determined manually or automatically. The presence of inappropriate features in the data can reduce accuracy and cause models to learn based on irrelevant features. Recursive feature elimination (RFE) is used for feature selection. RFE works by searching for a subset of features by starting with all (42 features) in the 5G training dataset and successfully removing features until the desired number remains. This is achieved by fitting the given machine-learning algorithm used in the model's core, sorting features by relevance, eliminating the least significant features, and re-fitting

TABLE 3 Coding the features categories.

No	Protocol type	Service	Flag	Protocol type code	Service code	Flag code
1	Ddp	–	INT	20	0	3
2	ipv6-frag	–	INT	53	0	3
3	Cftp	–	INT	12	0	3
4	Wsn	–	INT	128	0	3
5	Pvp	–	INT	87	0	3
6	wb-expak	–	INT	126	0	3
7	Mtp	–	INT	72	0	3
8	pri-enc	–	INT	83	0	3
9	sat-mon	–	INT	94	0	3
10	Cphb	–	INT	15	0	3

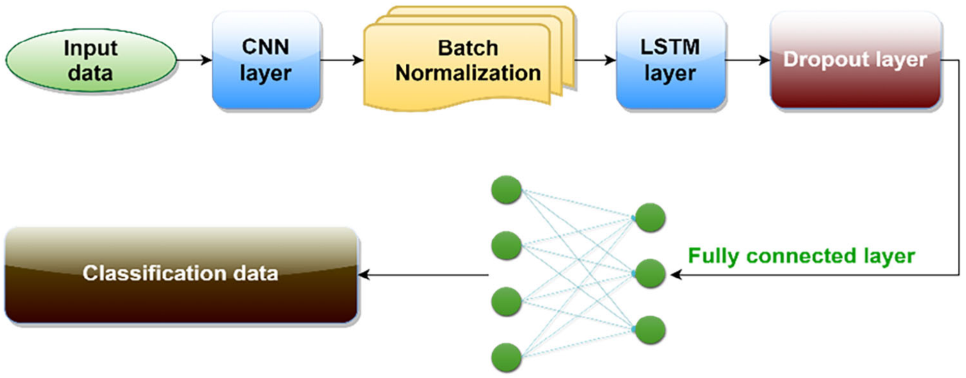


FIGURE 6 Hybrid deep learning classification model.

ALGORITHM 1 Feature Extraction with RFE alg.

```
Input: Features

Output: Best feature
Begin
While (more 5G dataset records exist) do
{
$D$ = next filed
Add feature to feature-list
}
Compute RFE score for each feature.
Select the best feature with the maximum score
Return best features
End
```

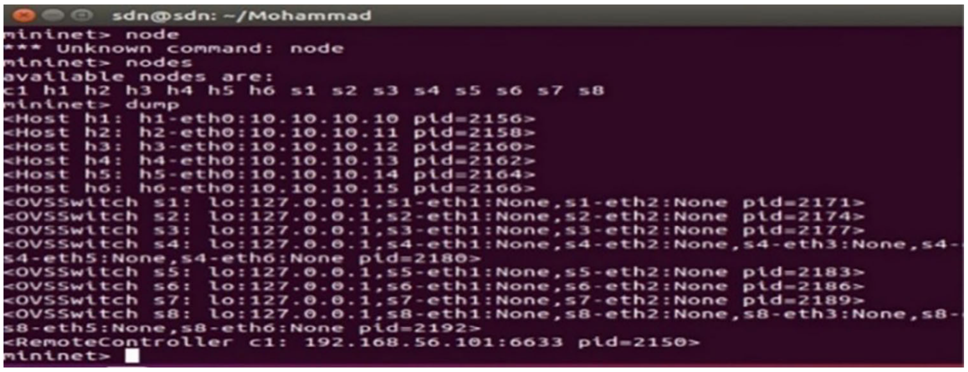
the model to achieve this. This procedure is continued until only the most closely related traits remain, as shown in Algorithm 1.

3.2.3 | Hybrid deep learning classification model

This part explains the architecture of our hybrid model to classify the network attack, depicted in Figure 6. The architecture is composed of two stages. In the first stage, the CNN is used to extract the spatial features, which contain four convolution layers with output dimensions of 32, 64, 128, and 128, respectively, with a kernel size of 3×3 for each convolution layer. In comparison, long short-term memory (LSTM) can extract temporal characteristics with a size 32. The data will pass through the convolution layer, where the filters will extract the most critical features to generate a feature map. Then the output will be sent to an LSTM layer to extract temporal features, followed by a dropout layer to prevent overfitting. This combination of CNN and LSTM layers, followed by a fully connected layer that uses the softmax layer, is used at the output for classification purposes, as shown in Algorithm 2. To enhance the detection model's capability, we used a hyperparameter.

3.3 | Build 5G network core using SDN

The first part, in this stage, is applied to create a separate device from the MiniNet on VMware on a local computer. The MiniNet network simulator involves a GUI editor called MiniEdit, a tool for demonstrating how the MiniNet could create networks. MiniEdit is distinguished by its simplified user interface. To add hosts to the network scenario, switches and controllers are added using their specified tools. The links are added between the nodes upon the canvas to create a connection between the terminals, switches, and controllers, forming



```
sdn@sdn: ~/Mohammad
mininet> node
*** Unknown command: node
mininet> nodes
available nodes are:
c1 h1 h2 h3 h4 h5 h6 s1 s2 s3 s4 s5 s6 s7 s8
mininet> dump
<Host h1: h1-eth0:10.10.10.10 pid=2156>
<Host h2: h2-eth0:10.10.10.11 pid=2158>
<Host h3: h3-eth0:10.10.10.12 pid=2160>
<Host h4: h4-eth0:10.10.10.13 pid=2162>
<Host h5: h5-eth0:10.10.10.14 pid=2164>
<Host h6: h6-eth0:10.10.10.15 pid=2166>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=2171>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None pid=2174>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=2177>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None,s4-
s4-eth5:None,s4-eth6:None pid=2180>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None pid=2183>
<OVSSwitch s6: lo:127.0.0.1,s6-eth1:None,s6-eth2:None pid=2186>
<OVSSwitch s7: lo:127.0.0.1,s7-eth1:None,s7-eth2:None pid=2189>
<OVSSwitch s8: lo:127.0.0.1,s8-eth1:None,s8-eth2:None,s8-eth3:None,s8-
s8-eth5:None,s8-eth6:None pid=2192>
<RemoteController c1: 192.168.56.101:6633 pid=2150>
mininet> 
```

FIGURE 7 Core network detail.

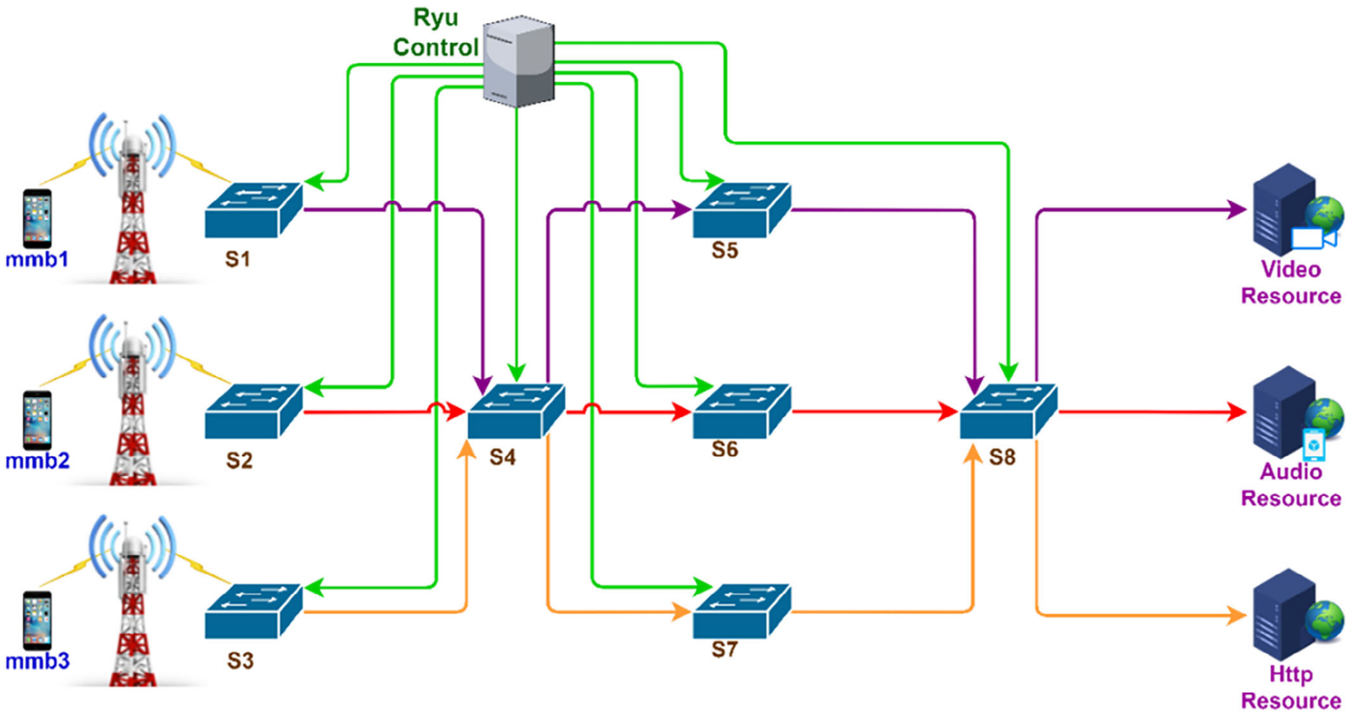


FIGURE 8 5G software-defined networking core network.

ALGORITHM 2 Hybrid Deep Learning Classification Model alg.

```
Input: Record of packet

Output: Classified packet
Begin
While (more 5G dataset records exist) do
{
$D$ = next packet feature passes through convolution filter layer
Extract the most critical features to generate a feature map
Do normalization
Pass through LSTM to extract temporal features
Prevent overfitting by a dropout layer
Classified packet
}
Return classified packets
End
```

a network. The Ryu control was used to control traffic flow between switches. Also, Figure 7 explains the details of the network.

This part uses SDN, an approach to networking that uses software-based controllers or APIs to communicate with underlying hardware infrastructure and direct traffic on a network. That differs from traditional networks, which use dedicated hardware devices (i.e., routers and switches) to control network traffic. Software defined networking can create and control a virtual or traditional hardware network via software.

SDN network allows organizations to segment different virtual networks within a single physical network. It also offers a novel method of regulating data packet routing through a centralized server, as illustrated in Figure 8 and Algorithm 3 and 4. The proposed system uses miniedit, a graphic user interface

ALGORITHM 3 5G network core

```
Input: information

Output: 5G core network (switching, routing, and control)

Begin
{
Create template host and host configuration
Create template link
{
http_link_config = dict(bw = 1)
audio_link_config = dict(bw = 10)
video_link_config = dict(bw = 100)
}
Create template Switch OpenFlow
{
addSwitch('s1',protocols = 'OpenFlow13',dpid = "001")
addSwitch('s2',protocols = 'OpenFlow13',dpid = "002")
addSwitch('s3',protocols = 'OpenFlow13',dpid = "003")
addSwitch('s4',protocols = 'OpenFlow13',dpid = "004")
addSwitch('s5',protocols = 'OpenFlow13',dpid = "005")
addSwitch('s6',protocols = 'OpenFlow13',dpid = "006")
addSwitch('s7',protocols = 'OpenFlow13',dpid = "007")
addSwitch('s8',protocols = 'OpenFlow13',dpid = "008")
}
Add links between switch
Add links between switch and tower
Add controller = RemoteController("c1")
}
Return 5G Network
End
```

ALGORITHM 4 5G Network slices

```
Input: 5G core network

Output: 5G network slices

Begin
{
Input OpenFlow switch port
Apply OpenFlow control
Mapped slice_to_port
Apply Ryn control on the switch
Select the datapath for each slice
Run the traffic on each slice
Return 5G network slice
}
End
```

(GUI) program, to design the network and convert the designed topology to the Python language with the OpenFlow protocol and Ryu control to manage the traffic flow. This network used eight Switch OpenFlow and three data sources (video, audio, and text). A controller works to divide the network into three segments according to the needs of the beneficiary. 5G network slices, in this study, will use SDN techniques and OpenFlow protocol to slice networks, which creates multiple unique logical and virtualized networks over a common multi-domain infrastructure. So, that can be used to support a certain application, service, group of users, or network. With its numerous applications, network slicing is one of the most essential 5G technologies. It will support new services with vastly different requirements, such as connecting vehicles, voice calls, and video, which require different throughput, latency, and reliability. Here, the network is sliced into three logical networks:

- The first one uses eMBB; these applications are very video-centric and consume much bandwidth, and will generate the most traffic on the mobile network; their tracks are shown in purple.
- The second slice is for audio applications; its tracks are shown in red.
- The last part is for web text applications (HTTP resources); its tracks are shown in orange.

Where the band of each slice is initially defined, and then the slice is determined for each application through a protocol OpenFlow.

3.4 | Improving QoS based on slicing and applying the attack detection model

The proposed system explains that network slicing provides security to the 5G mobile communications networks in various ways. The first security technique that network slicing provides is the isolation of slices from each other. The isolation limits any security challenge to a single slice rather than the entire network and improves QoS against attacks, reducing the scale of security impacts. The isolation essentially protects the resources and traffic in each network portion. Secondly, each slice has an attack detector model based on deep learning techniques for attack detection that better deals with the security vulnerabilities and problems they face. Third, the infrastructure for 5G slices allows devices or functions to be moved from one slice to another. With this capability, it is possible to isolate a network element subjected to a cyberattack and assign it to a different segment to prevent further damage due to the attack.

Ultimately, the distinction or difference between our proposed system and conventional systems lies in multiple aspects that enhance the functionality and bolster the security of fifth-generation networks. A key distinction lies in our approach, which combines two pivotal concepts.

- Firstly, it involves segmenting the network into distinct slices and assigning each slice to transmit specific data types to ensure enhanced quality. Consequently, in the event of an attack on one slice, its impact remains isolated without affecting other slices.
- Secondly, our methodology integrates neural network theories, leveraging optimized hyperparameters to enhance the detection of attacks on individual slices—an innovative approach not previously employed in traditional systems.

4 | IMPLEMENTATION, RESULTS, AND EVALUATION

This section introduces the experimental results of the proposed system. Various experiments are performed for the proposed techniques, starting with building an attack detection model, using SDN to build a 5G core network, slicing the 5G network, and improving QoS against attacks by using a slicing and attack detection model. The proposed system tests have been executed using the Python language 3.9 version, virtual machine (VM) in Oracle V.M. Virtualbox, and the Java programming language 8.0.2. The proposed system utilizes Mininet to create SDN devices. The Ryu controller has been connected to a mininet emulator to build the SDN environment.

4.1 | Attack detector model result

The system implements six stages to generate an attack detector.

4.1.1 | Preprocessing 5G dataset

This training data set contains noise and some data loss, which causes errors in processing and extracting important features. Therefore, pre-processing steps are essential to obtain the best result.

4.1.2 | Encoding

A transformation has been applied to convert the categorical values into numerical ones in order to apply them to the classification model, as shown in Table 3. Our dataset contains three features that have multi categories, such as protocol_type, which has 133 (like HTTP, DNS, SMTP, FTP, etc.) categories, and service has 13 categories (like TCP, U.D.P., UNAS, OSPF, SCTP, etc.) and feature flag has nine categories.

4.1.3 | Features standardization

Called (feature normalization), the standardization step in the proposed system is data standardization, which has an important role in removing the spacing between values and the difference in measurements between attributes, resulting in a bias in the classification process. The Z-score function imposed a convergence procedure between the values and increased classification accuracy. The continuous values were converted to discrete values to reduce the burden in the classification process. Here, μ is the mean value of the feature, and σ is the standard deviation of the feature. If a value equals the mean of all the feature values, it is normalised to 0. While if it is less than the mean, it is negative; if it is greater than the mean, it is positive. The size of such negative and positive integers is decided by the original feature's standard deviation, as in Equation (2). If the unnormalized data had a large standard deviation, the normalized values would be closer to 0. Table 4 shows the original features data for the dataset, while Table 5 shows the standardized data.

Five columns (fields), which are (duration, protocol_type, rate, sttl, and sload) have been represented by a diagram to show the difference before and after the normalization, as shown in Figure 9.

TABLE 4 Original data of features for dataset (before normalization).

Duration	Protocol_type	Service	Flag	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload	sloss
9.00×10^{-6}	20	0	3	2	0	200	0	111111.1	254	0	88888888	0	0
9.00×10^{-6}	53	0	3	2	0	200	0	111111.1	254	0	88888888	0	0
9.00×10^{-6}	12	0	3	2	0	200	0	111111.1	254	0	88888888	0	0
3.00×10^{-6}	128	0	3	2	0	200	0	333333.3	254	0	2.67E+08	0	0
8.00×10^{-6}	87	0	3	2	0	200	0	125000	254	0	1E+08	0	0
5.00×10^{-6}	126	0	3	2	0	200	0	200000	254	0	1.6E+08	0	0
8.00×10^{-6}	72	0	3	2	0	200	0	125000	254	0	1E+08	0	0
5.00×10^{-6}	83	0	3	2	0	200	0	200000	254	0	1.6E+08	0	0
8.00×10^{-6}	94	0	3	2	0	200	0	125000	254	0	1E+08	0	0
2.00×10^{-6}	15	0	3	2	0	200	0	500000	254	0	4E+08	0	0

4.1.4 | Feature selection

After analysing the 5G dataset, a total of (42) features were derived. However, a new challenge was confronted, namely sorting out the relevant features from the unrelated ones to improve how accurate the proposed model is. In ML, feature selection is one of the most important concepts significantly affecting the model's performance. The process of selecting the features that have the greatest impact on the prediction is called feature selection. This may be done either automatically or manually. Including unnecessary features may cause the model accuracy to decline, as the model will be trained using elements of no relevance.

The feature selection techniques have been used in this thesis to reduce overfitting, improve accuracy, and reduce training time, allowing for training more quickly. Initially, all features of the proposed model are used, obtaining an accuracy rate of approximately 94%. This is considered to be a low rate for predicting models. After using the recursive feature elimination model to select the most relative feature (34 features) with no logical changes to In the model code, as shown in Table 6, the accuracy rate became 99%, a more sufficient and effective rate as shown in Table 7.

4.1.5 | Select attack detection model

To select the best compatible attack classification model with the collected dataset, three algorithms are tested during the experimental phase: the DT, RF, hybrid CNN and LSTM. Each model was analysed using the collected dataset, and its parameters were enhanced and tuned.

Table 7 and Figure 10 show the results of each algorithm before and after using feature selection. The results conclude that the best algorithm for attack detection is the hybrid deep learning classification model used in this research.

4.2 | Network slicing result

In this section, the proposed system implements network slicing in an SDN environment to enable the isolation of network resources. To show how the different requirements can be fulfilled on a shared physical infrastructure by using network slicing. A multi-hop topology is used for emulation.

After building the network with three hosts, three sources, and eight switches, the bandwidth is 1, 10, and 100 Mbps. The system isolates the network topology into three slices: the upper slice (h1 -> s1 -> s4 -> s5 -> s8 -> source 1 (video source), 100 Mbps), the middle slice (h2 -> s2 -> s4 -> s6 -> s8 -> source 2 (audio source) 10 Mbps)and the lower slice (h3 -> s3 -> s4 -> s7 -> s8 -> source 3 (text source) 1 Mbps).

Ryu control will divide the physical network into three logical networks. Figure 10 and 11 shows the examination of the network in terms of the connection, as for example h1 station only connects to h4 (video source) and vice versa, as well as, Figure 12 shows the size of each slice. As an example, the upper slice has a 100 Mbps bandwidth size.

TABLE 5 Standardized data of features for the dataset (after normalization).

Duration	Protocol_type	Service	Flag	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload	sloss
−0.227	−2.87516	−0.61122	1.060234	−0.18109	−0.23158	−0.05509	−0.14354	0.454253	1.093685	−0.80213	0.176738	−0.39743	−0.09202
−0.227	−4.96052	−0.61122	1.060234	−0.18109	−0.23158	−0.05509	−0.14354	0.454253	1.093685	−0.80213	0.176738	−0.39743	−0.09202
−0.227	0.939521	−0.61122	1.060234	−0.18109	−0.23158	−0.05509	−0.14354	2.165058	1.093685	−0.80213	1.117266	−0.39743	−0.09202
−0.227	−1.14584	−0.61122	1.060234	−0.18109	−0.23158	−0.05509	−0.14354	0.561179	1.093685	−0.80213	0.235521	−0.39743	−0.09202
−0.227	0.837797	−0.61122	1.060234	−0.18109	−0.23158	−0.05509	−0.14354	1.138575	1.093685	−0.80213	0.552949	−0.39743	−0.09202
−0.227	−1.90877	−0.61122	1.060234	−0.18109	−0.23158	−0.05509	−0.14354	0.561179	1.093685	−0.80213	0.235521	−0.39743	−0.09202
−0.227	−1.34929	−0.61122	1.060234	−0.18109	−0.23158	−0.05509	−0.14354	1.138575	1.093685	−0.80213	0.552949	−0.39743	−0.09202
−0.227	−0.7898	−0.61122	1.060234	−0.18109	−0.23158	−0.05509	−0.14354	0.561179	1.093685	−0.80213	0.235521	−0.39743	−0.09202
−0.227	−4.80793	−0.61122	1.060234	−0.18109	−0.23158	−0.05509	−0.14354	3.448161	1.093685	−0.80213	1.822662	−0.39743	−0.09202
−0.227	0.023998	−0.61122	1.060234	−0.18109	−0.23158	−0.05509	−0.14354	0.561179	1.093685	−0.80213	0.235521	−0.39743	−0.09202

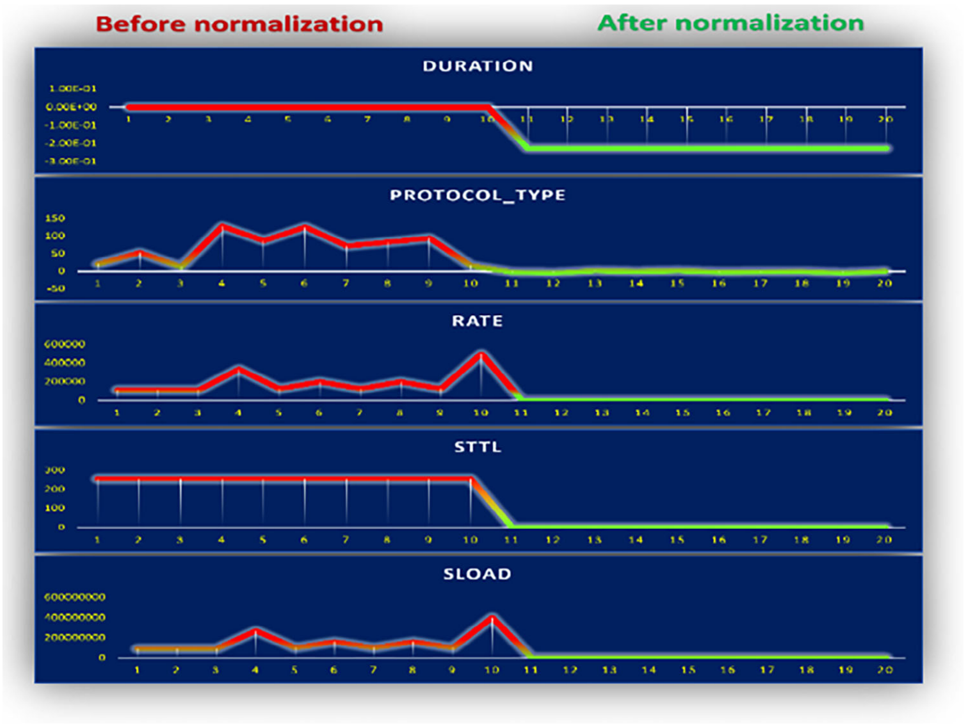


FIGURE 9 Before and after the dataset normalization.

TABLE 6 Feature selection result.

Item	Result
Num features	34
Selected features	T T T T T T T T T T T T T T T T T T T T T T T F F F T T F T T T T T T T F F F T T F
Feature ranking	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 6 5 1 1 3 1 1 1 1 1 1 1 8 9 2 1 1 7

Legend: T = True / F = False.

TABLE 7 Results of classification.

Algorithm	Before feature selection				After feature selection			
	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy
DT	0.95	0.94	0.94	0.94	0.97	0.97	0.98	0.98
RF	0.93	0.94	0.94	0.94	0.96	0.97	0.97	0.97
CNN and LSTM	0.92	0.93	0.93	0.9312	0.98	0.99	0.98	0.99

4.3 | Model evaluation

The proposed system in this study was evaluated in two phases:

4.3.1 | Evaluation of our model by using metrics

In the first stage, the results were evaluated using measures of precision, recall, $F1$ score, and accuracy; three ML models were tested during the experimental phase: RF, DT, and deep learning CNN with the LSTM algorithm. Each model was divided the data set into training (80%) and test (20%) sets. In the beginning, the ML algorithms were used without their default parameters and feature selection, as explained in Table 7. After that, parameter enhancement, normalization data, and feature selection were applied for each algorithm, as shown in Table 7 above. The result in Table 7 explains that the DT RF algorithms achieved high results for attack classification. The use of the feature selection technique and the tuning of hyperparameters have had a significant effect on improving the results. The deep learning CNN with the LSTM algorithm achieved better performances. Table 7 presents the classification results for the

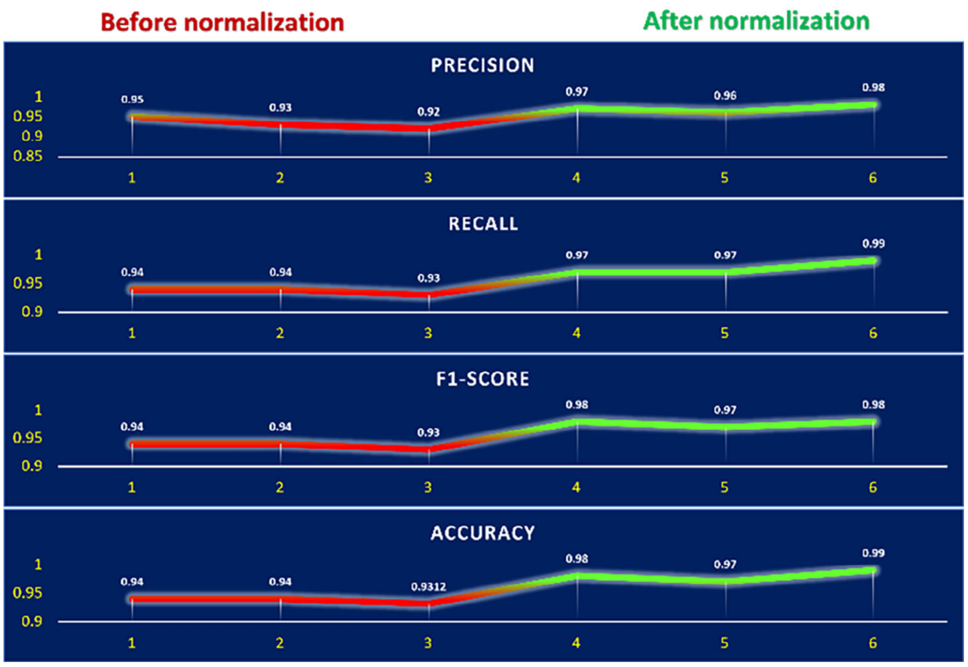
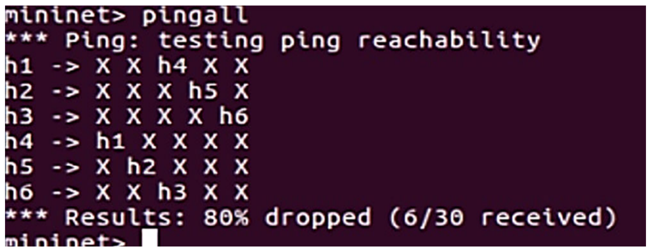
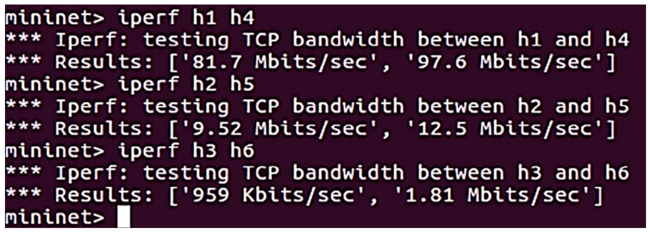


FIGURE 10 Before and after normalization of classification results.



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X h4 X X
h2 -> X X X h5 X
h3 -> X X X X h6
h4 -> h1 X X X X
h5 -> X h2 X X X
h6 -> X X h3 X X
*** Results: 80% dropped (6/30 received)
mininet> 
```

FIGURE 11 Verifying connectivity for each slice.



```
mininet> iperf h1 h4
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['81.7 Mbits/sec', '97.6 Mbits/sec']
mininet> iperf h2 h5
*** Iperf: testing TCP bandwidth between h2 and h5
*** Results: ['9.52 Mbits/sec', '12.5 Mbits/sec']
mininet> iperf h3 h6
*** Iperf: testing TCP bandwidth between h3 and h6
*** Results: ['959 Kbits/sec', '1.81 Mbits/sec']
mininet> 
```

FIGURE 12 Verifying bandwidth for each slice.

CNN that show rates of (0.98, 0.99, 0.98, and 0.99) for precision, recall, and $F1$-score and accuracy measures, respectively, after feature selection.

4.4 | Evaluation of our model with related literature

In this part, our proposal is evaluated through comparison with previous works. In comparison with Leila Mohammad Pour's system [54], which used these datasets to find the attack, our

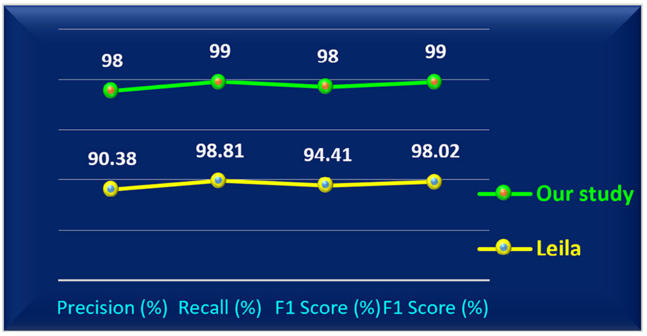


FIGURE 13 Comparison between Leila's result against the proposed.

system's results were better than Leila's result, where the degree of accuracy of its results was 98.02%. In comparison, the system accuracy result is 99%, as shown in Figure 13.

Moreover, compared to the work presented in [55], whereby the GA-LR-DT was used, the XGBoost-DT attained a test accuracy score of 90.85% compared to 81.42% obtained by the GA-LR-DT. Furthermore, the results obtained in this paper are superior to those obtained in [56]. The Sigmoid PIO selected 14 optimal features of the UNSW-NB15 and obtained an accuracy score of 91.30% through the validation dataset. A comparison between the proposed system in this paper against those surveyed in the literature shows that our system is more accurate, as shown in Figure 14.

4.5 | Summary

The simulation endeavours produced insightful outcomes across various phases of system implementation and

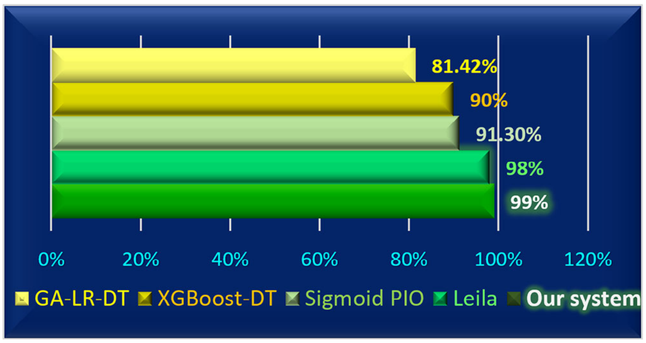


FIGURE 14 Comparison between previous works algorithm.

evaluation. Notably, the attack detector model exhibited robust performance following meticulous preprocessing, encoding, standardization, and feature selection processes. Through these steps, the accuracy rates surged to an impressive 99%, signifying the model's capability to discern and classify potential threats with heightened precision.

Moreover, the integration of network slicing within the SDN environment yielded compelling results. The creation of distinct slices with varied bandwidth allocations effectively demonstrated the capacity to isolate and allocate resources tailored to specific requirements. This dynamic allocation was exemplified by the upper, middle, and lower slices, each serving different data types with bandwidths of 100, 10, and 1 Mbps, respectively. In evaluating the model using comprehensive metrics encompassing precision, recall, $F1$-score, and accuracy, the comparison with established machine learning models showcased the system's superiority. Particularly, the fine-tuning of algorithms and feature selection techniques elevated the performance of the proposed deep learning CNN with LSTM, achieving remarkable precision and recall rates of 0.98 and 0.99, respectively, and an overall accuracy rate of 99%.

Furthermore, in juxtaposition with existing literature and comparative analyses, the system's accuracy rates consistently surpassed those of previous models using similar datasets. This substantiates the system's efficacy and prowess in outperforming established methodologies, thereby reinforcing its robustness and reliability in detecting attacks and optimizing network resource allocation. The simulation results collectively underscore the effectiveness and efficiency of the proposed system, reaffirming its competence in fortifying security measures, optimizing resource utilization through slicing, and surpassing benchmarks set by prior studies.

5 | CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK

This paper introduces a method aimed at enhancing quality of service (QoS) resilience against attacks in an SDN environment through the utilization of 5G network slicing alongside CNN and LSTM algorithms. The proposed model adopts a dual approach: segmenting the network into default groups and identifying services within each segment to optimize fifth-generation network quality. Our study demonstrated the superior performance of the CNN-LSTM approach compared to traditional ML algorithms such as DT or RF, indicating its potential for real-time attack detection.

Network slicing, furthermore, elevates network QoS by segmenting services based on bandwidth allocation. Future endeavours include implementing the proposed model in a live SDN system to evaluate performance in terms of throughput and latency across diverse datasets.

AUTHOR CONTRIBUTIONS

Mohammed Salah Abood: Conceptualization; resources; software. **Hua Wang**: Conceptualization; supervision; resources. **Bal S. Virdee**: Conceptualization; funding acquisition; investigation; resources. **Dongxuan He**: Data curation. **Maha Fathy**: Supervision. **Abdulganiyu Abdu Yusuf**: Formal analysis. **Omar Jamal**: Validation. **Taha A. Elwi**: Formal analysis; investigation; supervision. **Mohammad Alibakhshikenari**: Funding acquisition; investigation; supervision; formal analysis. **Lida Kouhalvandi**: Software; validation. **Ashfaq Ahmad**: Software; validation.

ACKNOWLEDGEMENTS

The research presented in this article is supported and funded by several institutions, including London Metropolitan University, the CONEX-Plus project at Universidad Carlos III de Madrid, and the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No. 801538. Additionally, funding is provided by the National Key Research and Development Program of China under Grant 2020YFB1807900, and the National Natural Science Foundation of China under Grant 62101306. Besides that, the authors sincerely appreciate funding from Researchers Supporting Project number (RSPD2024R666), King Saud University, Riyadh, Saudi Arabia.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Bal S. Virdee https://orcid.org/0000-0001-7203-0039
Dongxuan He https://orcid.org/0000-0002-5429-3318
Mohammad Alibakhshikenari https://orcid.org/0000-0002-8263-1572
Lida Kouhalvandi https://orcid.org/0000-0003-0693-4114

REFERENCES

1. Dangi, R., et al.: Study and investigation on 5G technology: A systematic review. Sensors 22(1), 26 (2021)
2. Ting, T.-H., et al.: Guidelines for 5G end to end architecture and security issues. arXiv:1912.10318 (2019)
3. Paskauskas, R.A.: ENISA: 5G design and architecture of global mobile networks; threats, risks, vulnerabilities; cybersecurity considerations. Open Res. Europe 2, 125 (2022)

4. Chochliouros, I.P., et al.: Dynamic network slicing: Challenges and opportunities. In: Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations. AIAI 2020 IFIP WG 12.5 International Workshops: MHDW 2020 and 5G-PINE 2020, pp. 47–60. Springer, Cham (2020)
5. Chen, G., et al.: Slicing resource allocation based on dueling DQN for eMBB and URLLC hybrid services in heterogeneous integrated networks. Sensors 23(5), 2518 (2023)
6. Sattar, D., Matrawy, A.: Towards secure slicing: Using slice isolation to mitigate DDoS attacks on 5G core network slices. In: Proceedings of the 2019 IEEE Conference on Communications and Network Security (CNS), pp. 82–90. IEEE, Piscataway, NJ (2019)
7. Schneider, P., Mannweiler, C., Kerboeuf, S.: Providing strong 5G mobile network slice isolation for highly sensitive third-party services. In: Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6. IEEE, Piscataway, NJ (2018)
8. Ni, J., Lin, X., Shen, X.S.: Efficient and secure service-oriented authentication supporting network slicing for 5G-enabled IoT. IEEE J. Sel. Areas Commun. 36(3), 644–657 (2018)
9. Thantharate, A., Beard, C., Kankariya, P.: Coap and MQTT based models to deliver software and security updates to IoT devices over the air. In: Proceedings of the 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1065–1070. IEEE, Piscataway, NJ (2019)
10. Ruckert, J., et al.: Demo: Software-defined network service chaining. In: Proceedings of the Third European Workshop on Software Defined Networks, pp. 139–140. IEEE, Piscataway, NJ (2014)
11. Arumaithurai, M., et al.: Prototype of an ICN based approach for flexible service chaining in SDN. In: Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 5–6. IEEE, Piscataway, NJ (2015)
12. Nikaein, N., et al.: Network store: Exploring slicing in future 5G networks. In: Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture, pp. 8–13. Association for Computing Machinery, New York, NY (2015)
13. Zahoor, S., et al.: Comprehensive analysis of network slicing for the developing commercial needs and networking challenges. Sensors 22(17), 6623 (2022)
14. Zhu, Z., et al.: MHSDN: A hierarchical software defined network reliability framework design. IET Inf. Secur. 17(1), 102–117 (2023)
15. Casado, M., et al.: Ethane: Taking control of the enterprise. ACM SIGCOMM Comput. Commun. Rev. 37(4), 1–12 (2007)
16. Kreutz, D., Ramos, F.M., Verissimo, P.: Towards secure and dependable software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 55–60. Association for Computing Machinery, New York, NY (2013)
17. Jain, S., et al.: B4: Experience with a globally-deployed software defined WAN. ACM SIGCOMM Comput. Commun. Rev. 43(4), 3–14 (2013)
18. McKeown, N., et al.: OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. 38(2), 69–74 (2008)
19. Saint-Andre, P.: RFC 6121: Extensible messaging and presence protocol (XMPP): instant messaging and presence. RFC Editor. Internet Engineering Task Force (2011)
20. Sur, D., et al.: Full-stack SDN. In: Proceedings of the 21st ACM Workshop on Hot Topics in Networks, pp. 130–137. Association for Computing Machinery, New York, NY (2022)
21. Pfaff, B., Davie, B.: The Open vSwitch Database Management Protocol. Internet Engineering Task Force (2013)
22. Duffy, J.: Cisco reveals OpenFlow SDN killer; OpFlex protocol for ACI offered to IETF. OpenDaylight. Network World (2014)
23. ONF: OpenFlow Switch Specification 1.1.0, pp. 1–58. Open Networking Foundation, Palo Alto, CA (2012)
24. ONF: OpenFlow Switch Specification 1.2.0, pp. 1–3205. Open Networking Foundation, Palo Alto, CA (2013)
25. Heller, B.: OpenFlow Switch Specification Version 1.3.0, pp. 1–36. Open Networking Foundation, Palo Alto, CA (2012)
26. OpenFlow 1.4 Specifications, pp. 1–36 (2013)
27. Foundation, O.N.: OpenFlow Switch Specification (Version 1.5.0), pp. 1–277. Open Networking Foundation, Palo Alto, CA (2014)
28. Foundation, O.N.: OpenFlow Switch Specification (Version 1.5.1), pp. 1–36. Open Networking Foundation, Palo Alto, CA (2015)
29. Bahassine, S., Madani, A., Kissi, M.: Arabic text classification using new stemmer for feature selection and decision trees. J. Eng. Sci. Technol. 12(6), 1475–1487 (2017)
30. Abooraig, R., et al.: Automatic categorization of Arabic articles based on their political orientation. Digital Invest. 25, 24–41 (2018)
31. Winter, P., Hermann, E., Zeilinger, M.: Inductive intrusion detection in flow-based network data using one-class support vector machines. In: Proceedings of the 2011 4th IFIP International Conference on New Technologies, Mobility and Security, pp. 1–5. IEEE, Piscataway, NJ (2011)
32. AlEroud, A., Alsmadi, I.: Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach. J. Network Comput. Appl. 80, 152–164 (2017)
33. Hussain, M., et al.: A comparison of SVM kernel functions for breast cancer detection. In: Proceedings of the 2011 Eighth International Conference Computer Graphics, Imaging and Visualization, pp. 145–150. IEEE, Piscataway, NJ (2011)
34. Rigatti, S.J.: Random forest. J. Insur. Med. 47(1), 31–39 (2017)
35. Das, K., Behera, R.N.: A survey on machine learning: Concept, algorithms and applications. Int. J. Innovative Res. Comput. Commun. Eng. 5(2), 1301–1309 (2017)
36. Du, R., Safavi-Naini, R., Susilo, W.: Web filtering using text classification. In: Proceedings of the the 11th IEEE International Conference on Networks, ICON2003, pp. 325–330. IEEE, Piscataway, NJ (2003)
37. Rokach, L., Maimon, O.: Data Mining and Knowledge Discovery Handbook. Springer, New York (2010)
38. Musa, R.A., Manaa, M.E., Abdul-Majeed, G.: Predicting autism spectrum disorder (ASD) for toddlers and children using data mining techniques. J. Phys.: Conf. Ser. 1804, 012089 (2021)
39. Sarker, I.H.: Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. SN Comput. Sci. 2(6), 420 (2021)
40. Krichen, M.: Convolutional neural networks: A survey. Computers 12(8), 151 (2023)
41. Yamashita, R., et al.: Convolutional neural networks: An overview and application in radiology. Insights into Imaging 9, 611–629 (2018)
42. Kattenborn, T., et al.: Review on convolutional neural networks (CNN) in vegetation remote sensing. ISPRS J. Photogramm. Remote Sens. 173, 24–49 (2021)
43. Unzueta, D.: Convolutional layers vs. fully connected layers. Towards Data Sci. (2021)
44. Yang, L., Shami, A.: On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing 415, 295–316 (2020)
45. Andonie, R.: Hyperparameter optimization in learning systems. J. Membr. Comput. 1(4), 279–291 (2019)
46. Gupta, P.C.: Cryptography and Network Security. PHI Learning Pvt. Ltd, Patparganj, New Delhi (2014)
47. Tibouchi, M., Wang, X.: Applied cryptography and network security. In: Proceedings of the 21st International Conference on ACNS 2023, vol. 13906. Springer International Publishing, Berlin, Heidelberg (2023)
48. Meena, G., Choudhary, R.R.: A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. In: Proceedings of the 2017 International Conference on Computer, Communications and Electronics (Comptelix), pp. 553–558. IEEE, Piscataway, NJ (2017)
49. Potdar, K., Pardawala, T.S., Pai, C.D.: A comparative study of categorical variable encoding techniques for neural network classifiers. Int. J. Comput. Appl. 175(4), 7–9 (2017)
50. Saad, M., et al.: Mempool optimization for defending against DDoS attacks in PoW-based blockchain systems. In: Proceedings of the 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 285–292. IEEE, Piscataway, NJ (2019)

51. Koay, A., et al.: A new multi classifier system using entropy-based features in DDoS attack detection. In: Proceedings of the 2018 International Conference on Information Networking (ICOIN), pp. 162–167. IEEE, Piscataway, NJ (2018)
52. Al-Mafrachi, B.H.A.: Detection of DDoS Attacks Against the SDN Controller Using Statistical Approaches. Wright State University, Dayton, Ohio (2017)
53. Moustafa, N., Slay, J.: UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6. IEEE, Piscataway, NJ (2015)
54. Mohammadpour, L., et al.: A survey of CNN-based network intrusion detection. Appl. Sci. 12(16), 8162 (2022)
55. Khammassi, C., Krichen, S.: A GA-LR wrapper approach for feature selection in network intrusion detection. Comput. Secur. 70, 255–277 (2017)
56. Alazzam, H., Sharieh, A., Sabri, K.E.: A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer. Expert Syst. Appl. 148, 113249 (2020)

How to cite this article: Abood, M.S., Wang, H., Virdee, B.S., He, D., Fathy, M., Yusuf, A.A., Jamal, O., Elwi, T.A., Alibakhshikenari, M., Kouhalvandi, L., Ahmad, A.: Improved 5G network slicing for enhanced QoS against attack in SDN environment using deep learning. IET Commun. 1–19 (2024). https://doi.org/10.1049/cmu2.12735