systems

Article

Deep Learning-Based Approach for Detecting DDoS Attack on Software-Defined Networking Controller

Amran Mansoor [1], Mohammed Anbar [1,*], Abdullah Ahmed Bahashwan [1], Basim Ahmad Alabsi [2] and Shaza Dawood Ahmed Rihan [2]

[1] National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia, Gelugor 11800, Penang, Malaysia; bahashwan@student.usm.my (A.A.B.)
[2] Applied College, Najran University, King Abdulaziz Street, Najran 11001, Saudi Arabia
[*] Correspondence: anbar@usm.my

Abstract: The rapid growth of cloud computing has led to the development of the Software-Defined Network (SDN), which is a network strategy that offers dynamic management and improved performance. However, security threats are a growing concern, particularly with the SDN controller becoming an attractive target for malicious actors and potential Distributed Denial of Service (DDoS) attacks. Many researchers have proposed different approaches to detecting DDoS attacks. However, those approaches suffer from high false positives, leading to low accuracy, and the main reason behind this is the use of non-qualified features and non-realistic datasets. Therefore, the deep learning (DL) algorithmic technique can be utilized to detect DDoS attacks on SDN controllers. Moreover, the proposed approach involves three stages, (1) data preprocessing, (2) cross-feature selection, which aims to identify important features for DDoS detection, and (3) detection using the Recurrent Neural Networks (RNNs) model. A benchmark dataset is employed to evaluate the proposed approach via standard evaluation metrics, including false positive rate and detection accuracy. The findings indicate that the recommended approach effectively detects DDoS attacks with average detection accuracy, average precision, average FPR, and average F1-measure of 94.186%, 92.146%, 8.114%, and 94.276%, respectively.

Keywords: deep learning; Recurrent Neural Networks; Distributed Denial of Service; Software-Defined Networking

check for updates

Citation: Mansoor, A.; Anbar, M.; Bahashwan, A.A.; Alabsi, B.A.; Rihan, S.D.A. Deep Learning-Based Approach for Detecting DDoS Attack on Software-Defined Networking Controller. *Systems* **2023**, *11*, 296. https://doi.org/10.3390/systems11060296

Academic Editor:Vladimír Bureš

Received: 4 May 2023
Revised: 6 June 2023
Accepted: 7 June 2023
Published: 9 June 2023

1. Introduction

SDN is an innovative network architecture that enhances network performance by decoupling the two major planes: the control plane and the data plane [1]. The SDN controller manages, controls, and enables OpenFlow forwarding rules, serving as a logical node [1]. As described by [2], an SDN comprises two primary tiers: the management plane (housing the SDN controller) and the information plane (consisting of OpenFlow switches).

Modern cloud computing and hyper-scale cloud architecture have significantly increased the popularity of SDN. This is attributed to its ability to enhance network efficiency and streamline administration processes. By separating the data and control planes, SDN enables logical management within a physical network architecture. However, despite its technological advancements, SDN introduces new security concerns, especially in light of existing and emerging threats [3–5]. One particular area of concern is the vulnerability of SDN controllers to DDoS attacks. In an SDN network, DDoS attacks pose a significant threat to the controller. Figure 1 illustrates the impact of DDoS attacks targeting the SDN network controller.

Additionally, DDoS attacks are typically executed by leveraging numerous compromised machines to generate a massive influx of traffic toward the target machine or SDN controller. Consequently, the targeted machine faces an overwhelming resource burden as it attempts to process the influx of network traffic, rendering it unable to fulfill its intended functions. Over time, DDoS attacks have become increasingly prevalent and destructive, capable of disrupting a wide range of network services. Furthermore, they are notoriously challenging to detect and mitigate effectively. Therefore, the timely detection and prevention of DDoS attacks represent critical concerns for network service providers. Regrettably, the SDN controller lacks an integrated security defense system that can efficiently distinguish between normal and abnormal network traffic [6,7].

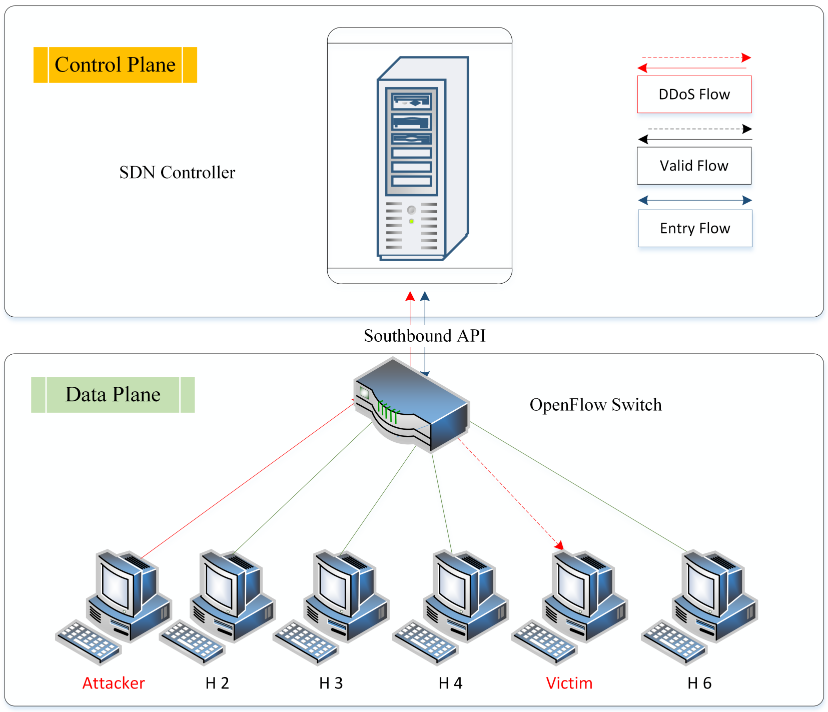


Figure 1. DDoS Attack in SDN Controller.

Furthermore, numerous researchers have proposed different approaches to detect DDoS attacks on SDN controllers [8–12]. However, despite the efficiency of these approaches, detecting DDoS attacks on SDN controllers remains a challenging problem. The majority of these approaches suffer from high false positives, resulting in low detection accuracy. This low accuracy can be attributed to two main factors: reliance on non-qualified features and evaluation using non-realistic datasets.

For example, Wan et al. [9] demonstrated that relying on non-qualified features resulted in the classifier's inability to differentiate between attack traffic and normal traffic for detecting DDoS attacks. Similarly, Iasc et al. [10] showed that using non-realistic datasets does not accurately reflect the essential features of real SDN traffic and may lead to unfair performance comparisons with other state-of-the-art approaches.

Therefore, it is crucial to address these limitations by considering qualified features and realistic datasets when developing DDoS attack detection approaches for SDN controllers.

Hence, this research makes three main contributions:

- Proposing an Information Gain Ratio (IGR) and Chi-square-based cross-feature selection mechanism. This mechanism aims to identify the most informative features of network traffic. Additionally, a feature intersection technique is introduced to further enhance the selection process, improving the detection of DDoS attacks on SDN network controllers.
- Developing a robust DL RNN model trained using the selected features. This model enables the accurate detection of UDP, TCP, and ICMP attacks by effectively capturing the intricate behaviors associated with these attacks.

- Evaluating the proposed model and demonstrating its promising findings through various evaluation metrics, including false positive rate, F1 score, detection accuracy, and precision. The results highlight the effectiveness and reliability of the proposed approach.

The remaining sections of this research paper are structured as follows: Section 2 underlines the background. Sections 3–5 underline the associated works, discussion on preliminaries of the proposed approach, and the DLADSC approach, respectively. Section 6 discusses the results of the experiments. Section 7 discusses the findings of the research and provides further analysis and interpretation of the results. Finally, Section 8 concludes the research paper, summarizing the main contributions and discussing potential future research directions.

2. Background

This section provides an overview of the SDN architecture, discusses the role of SDN controllers and the OpenFlow protocol, and highlights the significance of DL-based approaches in the context of SDN networks.

2.1. SDN

SDN is a network architecture that fundamentally revolutionizes the design and management of modern network architectures, especially in today's hyper-scale data centers where extremely large networks are prevalent [13]. It involves the centralized control, management, and configuration of all networking devices from a single controller [13]. The SDN architecture is typically represented in layers or planes, as illustrated in Figure 2, which depicts the three layers: forwarding or infrastructure, control, and applications.

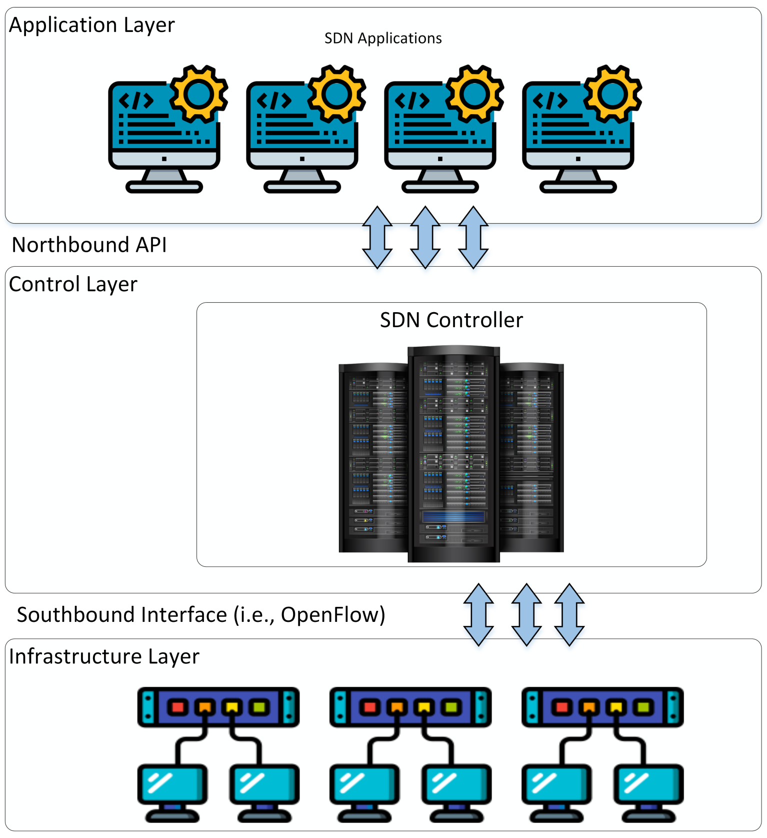


Figure 2. SDN Architecture.

2.2. SDN Controllers and OpenFlow Protocol

The logically centralized SDN controller performs tasks such as analyzing, managing, and collecting statistics on incoming network traffic packets as well as sending new instructions to the switches' tables. Being centrally located, the controller has the responsibility of directing all packets within the network and making decisions based on the gathered information. In most cases, the SDN controller serves as a reflection and model of the SDN framework. Communication with switches is facilitated through southbound application APIs, such as OpenFlow, while SDN applications such as load balancers and firewalls rely on northbound APIs [14].

There are various SDN controllers available in the market, ranging from open-source OpenFlow-based controllers to commercial products such as vCloud and vSphere (VMware), Trema (NEC), NVP (Nicira), Floodlight, and BNC (Big Switch Networks) [15]. The communication protocol between the control and forwarding planes in SDN networks is established by OpenFlow, which utilizes three message types: symmetric, asynchronous, and controller-to-switch messages [16]. Figure 3 illustrates the OpenFlow communication protocol in SDN networks.

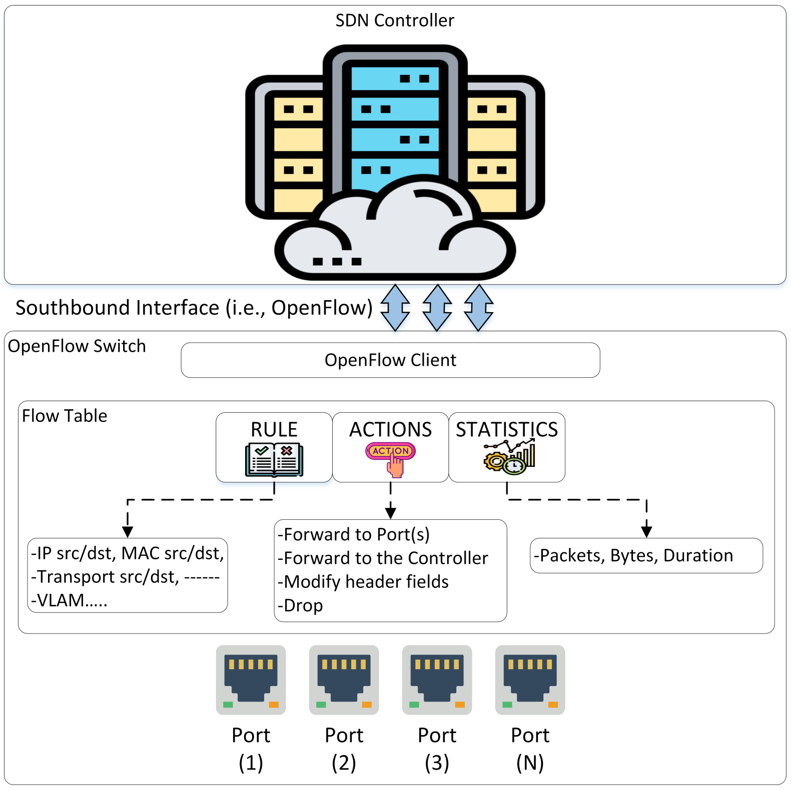


Figure 3. Communication via OpenFlow Protocol.

Forwarding devices, such as OpenFlow switches, are equipped with an abstraction layer and one or more flow tables. The OpenFlow protocol enables the establishment of a secure communication link between the SDN controller and the switches in the infrastructure layer. Based on the flow entries stored in the flow table, packets are routed and processed. Each flow entry consists of counts, match fields, and a sequence of operations. Counters can be used to monitor flow statistics, while match fields utilize data from the packet header to identify arriving packets. The OpenFlow switch parses the header fields of a packet and compares them to the match field entries in the flow table. When a matching flow entry is found, the switch executes the corresponding set of instructions. If no match is found, the switch will either discard the packet or forward it to the controller, depending on its instructions [15].

2.3. DL Approaches

Artificial intelligence (AI) is a rapidly evolving field with numerous practical applications and ongoing research interests. It enables the development of intelligent software capable of automating tasks, analyzing images, understanding conversations, making medical diagnoses, creating smart infrastructure, empowering individuals with physical disabilities, and supporting scientific research. Machine learning (ML) serves as the foundation for most AI solutions. With the vast amount of data available today, we can utilize it to train ML models that can make predictions and draw inferences based on the patterns and associations present in the data. AI, ML, and deep learning (DL) share a relationship where DL is a type of representation learning, and ML is utilized in many, though not all, AI systems. Each section of the Venn diagram in Figure 4 also provides an example of AI technology [17].

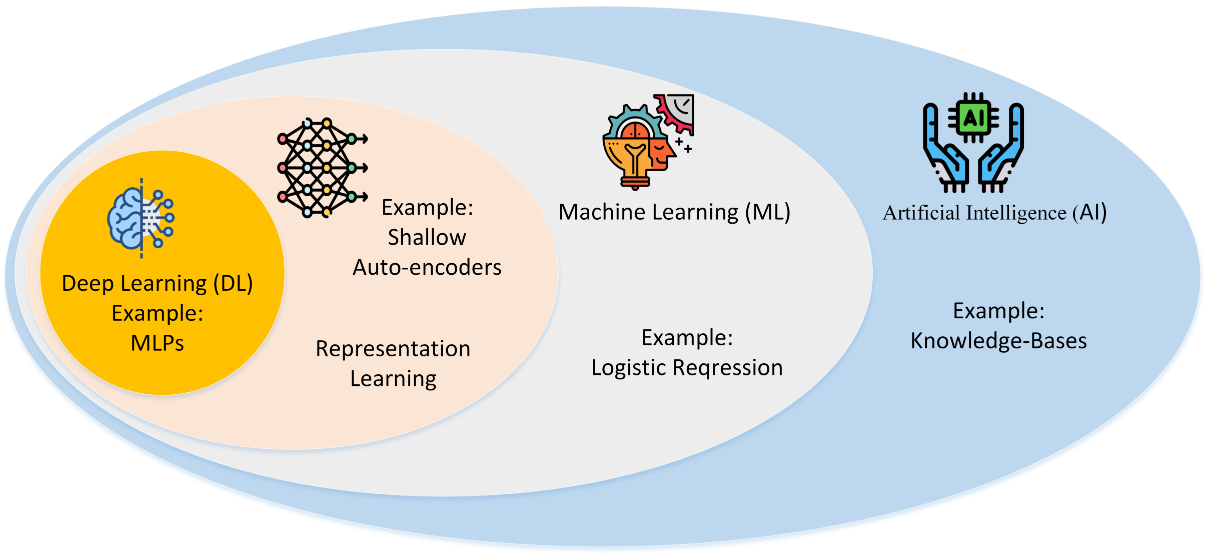


Figure 4. Venn Diagram Representing the Relationships Between AI, ML and DL.

DL enhances the capabilities of classical ML by introducing increased complexity to the model and modifying data operations to enable hierarchical data representation through multiple layers of abstraction [18,19]. One of the key advantages of DL is feature learning, where raw data features are automatically extracted, and higher-level features are formed through the combination of lower-level features [20]. DL excels in handling complex problems and models, allowing for efficient parallelization. The components of DL vary depending on the network architecture employed and may include pooling layers, convolutions, gates, fully connected layers, activation functions, memory cells, and encode/decode methods [21].

In addition to the existing neural network types, ongoing research is leading to the exploration of new ones. Neural networks can be categorized based on their structure, data flow, processing nodes (neurons), density, layers, and depth activation filters [22]. Some commonly used neural network types include Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks [23]. Each type of neural network has its own strengths and weaknesses, depending on the specific input/output data requirements, usage scenarios, and use cases. Table 1 provides a comparison of the inputs or outputs, architecture, practical cases, and suitable usage scenarios for CNNs, RNNs, and LSTMs.

Table 1. Comparison between CNN, RNN, and LTSM.

	CNN	**RNN**	**LSTM**
Inputs or Outputs	Both the inputs and the outputs sizes are fixed.	The size of the input and the output that results might differ.	The size of the input and the output that results might differ.
Model Architecture	Using filters and pooling, feed-forward neural networks.	A Recurrent Network that re-feeds the network with the findings.	LSTM and RNN are groups of gates that can keep data in memory for a protracted amount of time.
Practical Cases	Face identification, medical analysis, image analysis, image recognition and classification.	Text translation, sentiment analysis natural language processing, and speech analysis	Speech recognition, handwriting recognition, and image processing.
Suitable Usage Scenarios	Spatial data, such as a picture.	Data that are temporal or sequential, such as text or video.	Data that are temporal or sequential, such as text or video.

In summary, CNN, RNN, and LSTM are all popular DL approaches that have been widely used in various domains, and every one of these approaches has its strengths and limitations, making them suitable for different types of takes. The choice of DL models is based on the specific problem domain and the characteristics of the data being processed.

3. Related Works

This section reviews the existing ML and DL-based approaches for detecting DDoS against an SDN controller. This section consists of two subsections: ML-based approaches and DL-based approaches.

3.1. ML-Based Approach

ML-based approaches are developed and improved using ML algorithms. These approaches train IDSs to recognize malicious DDoS attacks by analyzing network traffic properties. For instance, a method to detect and reduce the severity of DDoS attacks in SDN was suggested by Khashab et al. [24]. They utilized six ML algorithms, including Random Forest (RF), Logistic Regression, Naïve Bayes, K-Nearest Neighbors (K-NN), Support Vector Machine (SVM), and Decision Trees. The results of the proposed approach indicated that RF outperformed the other algorithms and was the most suitable classifier for their method. However, the authors did not provide sufficient details about the dataset used or the specific types of DDoS attacks considered. Additionally, implementing the proposed approach on the SDN controller increases overhead during DDoS attack scenarios.

Sudar et al. [8] conducted research on using Support Vector Machine (SVM) and Decision Trees (DTs) for detecting DDoS attacks in SDN networks. They evaluated their proposed approach using the KDD CUP dataset. However, their method achieved inadequate performance, with Decision Trees achieving only a 78.

Santos et al. [12] utilized Multiple Layer Perceptron (MLP), Random Forest Algorithm, SVM, and DTs to detect various types of DDoS attacks, including point attacks, flow table switch attacks, SDN controller attacks, and bandwidth attacks. They evaluated their approach using a realistic dataset. However, the results indicated poor classification performance for MLP and SVM in detecting attacks on the controller, with an accuracy rate of only 90%.

Celesova et al. [25] suggested a method that utilizes a Deep Neural Network (DNN) to protect the data planes and control DDoS attacks in SDN networks. However, they used the UNSW-NB15 dataset, which is not specifically designed for the SDN network environment, to train, test, and evaluate their proposed system. Consequently, the method achieved low performance in terms of calculation metrics.

In contrast, Deepa et al. [11] proposed a hybrid ML algorithm that combines Support Vector Machine (SVM) and Self-Organizing Map (SOM) for detecting DDoS attacks in SDN networks. The hybrid model achieved a detection rate of 90.45%, an accurate detection rate of 96.77%, and a false alarm rate of 0.032%. However, the proposed hybrid approach showed low accuracy and detection rate performance. Additionally, the authors provided limited information about the dataset and the types of DDoS attacks considered.

The ML-based approaches are listed in Table 2 along with their drawbacks.

3.2. DL-Based Approaches

In recent years, DL algorithms have played a significant role in IDSs for detecting DDoS attacks in SDN networks. This is because DL algorithms have proven to be highly efficient and effective, surpassing the traditional ML-based approaches. Moreover, DL algorithms possess robust capabilities in mimicking the human brain, allowing them to acquire features and automatically learn deep structures from raw data. There are several DL-based approaches that have been proposed. For example, Alanazi et al. [10] proposed an ensemble approach based on a combination of Gated Recurrent Unit (GRU), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) to detect DDoS attacks in SDN networks. The ensemble was evaluated using the CICIDS2020 dataset and achieved high detection accuracy by selecting only four features.

Hsieh et al. [26] proposed a DL technique based on CNN and DNN to detect link flooding attacks. They tested their approach using a generated dataset and obtained poor results, with a detection accuracy rate of 95.03%. Additionally, the approach increased the overhead at the controller level.

Wan et al. [9] utilized a bidirectional Long Short-Term Memory (LSTM) network to detect such attacks in SDN networks. The proposed approach achieved successful detection with high accuracy and low false positive rates. However, the approach was tested using a non-SDN dataset.

Lee et al. [27] designed an Intrusion Detection System (IDS) based on four DL algorithms: Multilayer Perceptron (MLP), LSTM, Stacked Autoencoder (SAE), and Convolutional Neural Network (CNN), to detect DDoS attacks and secure shell (SSH) brute force attempts in an SDN network. MLP achieved a high detection accuracy of 98.3% and 99% for SSH attacks and DDoS attacks, respectively. However, the performance of other DL algorithms in detecting such attacks was poor due to a lack of information regarding the dataset and features.

Boukria et al. [28] used standard DL algorithms to detect attempts at attacks in the communication channel (southbound API) that links the SDN controller with the SDN infrastructure plane. The authors claimed that the proposed system achieved a detection accuracy of 99.6%. However, the approach was tested and evaluated using a non-SDN dataset that does not represent an SDN network. Additionally, it is limited to protecting the communication channels only. Table 2 outlines the DL-based techniques and their shortcomings.

Table 2. Summary of the Related Works of ML and DL Approaches.

Author and Ref.	**Method**		**SDN Dataset**	**SDN Domain**	**Detection Accuracy**		**Limitations**
	ML	**DL**			**High**	**Low**	
Sudar et al. [8]	✓	✗	✗	✓	✗	✓	Achieves low performance. Using non-SDN dataset.
Wan et al. [9]	✗	✓	✗	✓	✓	✗	The approach is evaluated using a non-SDN dataset.
Alanazi et al. [10]	✗	✓	✗	✓	✓	✗	The approach is evaluated using a dataset not suitable for the SDN network.
Deepa et al. [11]	✓	✗	✓	✓	✗	✓	Achieves low performance. Inadequate information about the dataset and the types of attacks involving DDoS.
Santos et al. [12]	✓	✗	✓	✓	✗	✓	Achieves low detection accuracy for SDN controller DDoS attacks.

Table 2. *Cont.*

Author and Ref.	Method		SDN Dataset	SDN Domain	Detection Accuracy		Limitations
	ML	DL			High	Low	
Khashab et al. [24]	✓	✗	✓	✓	✓	✗	The approach performs at Controller, increasing the workload during attacks. It is recommended to select relevant features that increase other ML algorithms' system performance and detection accuracy.
Celesova et al. [25]	✓	✗	✗	✓	✗	✓	Achieves low performance. Uses a non-SDN dataset.
Hsieh et al. [26]	✗	✓	✓	✓	✗	✓	Achieves low performance. The proposed approach operated at the controller, which increases the overhead during DDoS attacks.
Lee et al. [27]	✗	✓	-	✓	✓	✗	A lack of information about the dataset and the used traffic features.
Boukria et al. [28]	✗	✓	✗	✓	✓	✗	The approach is evaluated using a non-SDN dataset. It is limited to protecting the communication channels only.

As shown in Table 2, several approaches in the literature have achieved low detection accuracies, such as those of Santos et al. [12], Sudar et al. [8], Celesova et al. [25], Hsieh et al. [26], and Deepa et al. [11]. Additionally, some of these approaches have been evaluated using non-SDN datasets, including Celesova et al. [25], Sudar et al. [8], Boukria et al. [28], and Alanazi et al. [10]. Moreover, most of the existing approaches are implemented on SDN controllers, which can increase overhead during DDoS attacks. Overall, the proposed system addresses these limitations by offering an RNN-based approach to detect DDoS attacks in the context of SDN with low FPR and high detection accuracy.

4. Preliminaries

This preliminary section aims to address the security concerns that arise within the layers of SDN. SDN architectures encompass multiple layers, including the forwarding or infrastructure layer, the controlling layer, and the applications layer. However, each of these layers is susceptible to various security vulnerabilities and threats.

4.1. Security Issues in SDN

Despite the numerous capabilities and functionalities of Software-Defined Networking (SDN), security remains a critical concern. The SDN controller, being the central component responsible for managing data flow, poses the highest risk of a single-point failure. Compromising the SDN controller can have severe consequences on the entire network. Misconfigurations in SDN controllers can lead to significant repercussions, as their programmability exposes them to potential attacks, compromising network authentication, security, and integrity. Implementing security monitoring, analysis, and response mechanisms within SDN can help enhance network security. It is important to note that cyberattacks targeting SDN can result in more devastating consequences compared to traditional networks [29].

Table 3 provides an overview of the various types of potential attacks that can occur in SDN deployments, which are categorized based on threat levels across different SDN layers [30]. Each layer of SDN has its own specific security requirements, such as preventing configuration errors. Failure to meet these requirements exposes the network to various security threats and attacks. If the communication link between switches and controllers is compromised, all system levels become vulnerable to flooding attacks. Attacks on enforcement security can impact the upper layers, while attacks on the authorization process may deny users access to the controller.

Table 3. Security threats on SDN different layers.

Security Threats in SDN Layers		Data Layer	Controller Layer	Application Layer
Illegitimate Access	Unauthorized application.	✗	✓	✓
	Hijacking SDN controller.	✓	✓	✗
Configuration Issues	Lack of TLS implementation.	✓	✓	✓
	Enforcement of policy.	✓	✓	✓
DoS Attack	Flooding controller.	✓	✓	✗
	Flooding switch.	✓	✗	✗
Modification Data	Data flow tampering.	✓	✓	✗
Data Leakage	Credential management.	✗	✓	✗
	Forwarding policy discovery.	✗	✓	✗

Various forms of malicious attacks can compromise the security of SDN networks, including DDoS attacks, ARP spoofing attacks, packet sniffing, API exploitation, and guessing the passwords or brute force attacks [30]. This research takes into account a mechanism for aggregating feature selection that contributes to detecting DDoS attacks. We recommend a DL RNN-based approach, which will be discussed as follows.

4.2. RNN-Based Approach

Alongside the increasing reliance on SDN technology, ensuring the security of these networks has become paramount. One of the major threats faced by the SDN is DDoS attacks, which can disrupt normal network functioning and compromise the availability of services. IDS play a crucial role in detecting such attacks. However, traditionally, signature-based IDS struggle to keep up with the evolving nature of DDoS attacks, as attackers continuously employ new strategies. To address this challenge, DL has gained prominence in the field of IDS [14].

In this research, we are using RNN as the preferred DL algorithm due to the following reasons. Firstly, RNNs have internal memories that allow them to retain important information from the input they receive. This characteristic enables them to make accurate predictions about future data points, making them well-suited for capturing the temporal dependencies present in the dataset. Secondly, the dataset used in this research consists of a time series of sequential data, where the order of the data points is significant. RNNs are particularly effective in handling such sequential data, as they can process information in a sequential manner and learn patterns and dependencies over time. Lastly, RNNs are considered one of the most promising neural network algorithms due to their ability to incorporate internal memory. This internal memory allows RNNs to capture long-term dependencies in the data, making them robust and powerful in modeling complex relationships [31].

5. DLADSC Approach

This section presents the proposed approach, a Deep Learning-Based Approach for Detecting DDoS Attacks on a Software-Defined Networking Controller (DLADSC), which aims to detect DDoS attacks by identifying distinctive features that differentiate DDoS network traffic from regular traffic. DLADSC consists of three stages: data preprocessing, cross-feature selection, and an RNN-based model for detecting DDoS attacks. Each stage plays a crucial role in the overall detection process.

The first stage involves data preprocessing, where the input data are prepared for further analysis. This includes applying necessary transformations and normalization techniques to ensure the data are suitable for feature extraction and selection. By cleaning and organizing the data, they become more suitable for accurate detection.

The second stage focuses on cross-feature selection, which is essential for identifying the most important features in detecting DDoS attacks on the SDN controller. This is achieved through the use of feature selection techniques such as Chi-square and IGR. By selecting the most relevant features, the detection accuracy can be improved significantly, leading to a more effective identification of DDoS attacks.

Once the features are selected, they are utilized in training the RNN-based detection model. The RNN model is designed to capture the temporal patterns and dependencies present in network traffic, making it well-suited for detecting DDoS attacks. By utilizing the selected features, the model can learn and classify incoming traffic as either normal or malicious.

Figure 5 provides a visual representation of these three stages, illustrating the flow of the proposed approach. In the following sections, each stage will be described in detail, highlighting their significance in achieving accurate and robust DDoS attack detection on SDN controllers.

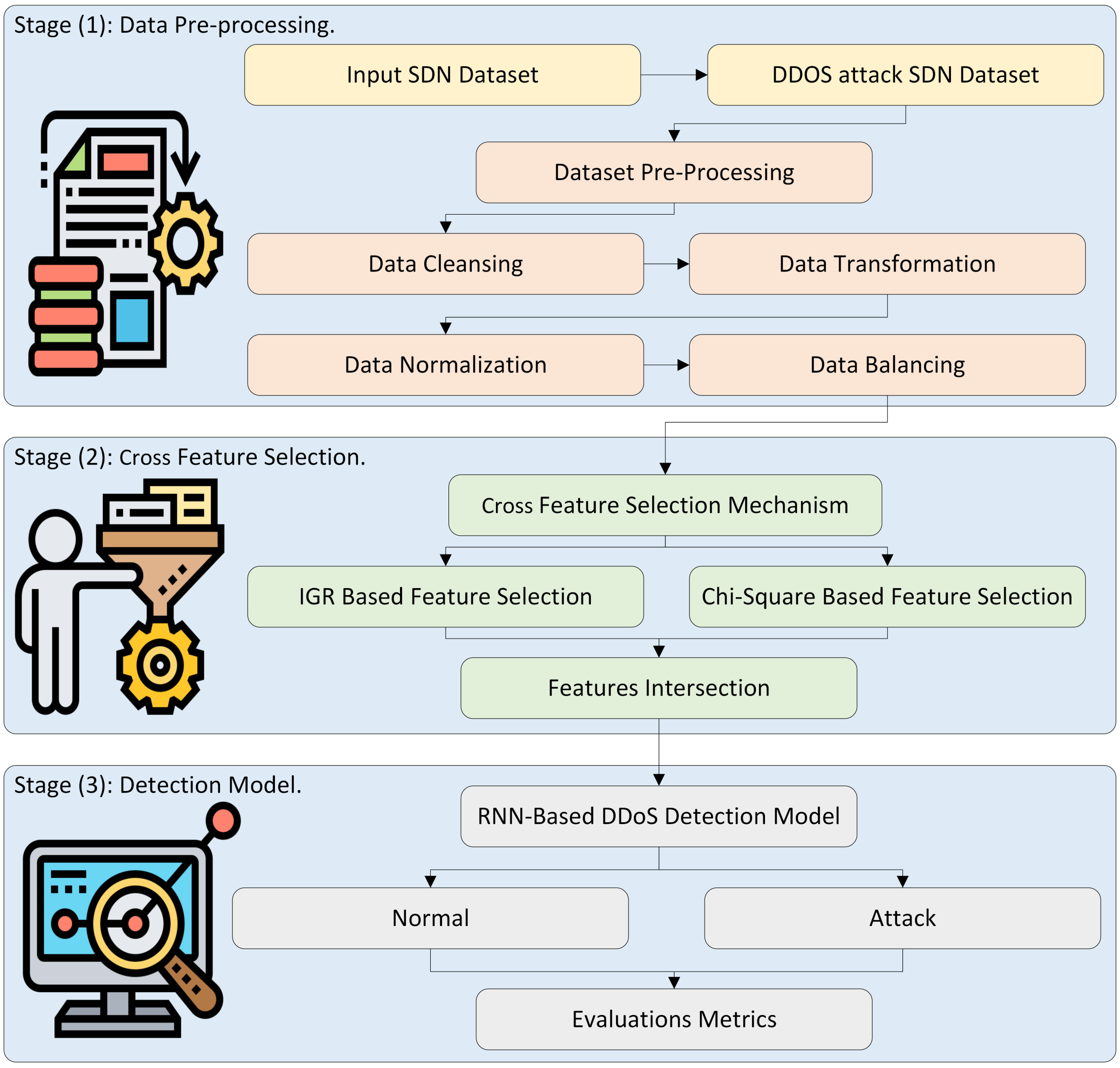

Stage (1): Data Pre-processing.
Input SDN Dataset
DDOS attack SDN Dataset
Dataset Pre-Processing
Data Cleansing
Data Transformation
Data Normalization
Data Balancing
Stage (2): Cross Feature Selection.
Cross Feature Selection Mechanism
IGR Based Feature Selection
Chi-Square Based Feature Selection
Features Intersection
Stage (3): Detection Model.
RNN-Based DDoS Detection Model
Normal
Attack
Evaluations Metrics

Figure 5. The architecture of the proposed approach.

5.1. Dataset Preprocessing

The proposed approach has been evaluated using the benchmark dataset “DDOS attack SDN Dataset” [32]. This dataset was specifically generated on a testbed comprising 10 Mininet topologies, where switches are connected to a single Ryu controller. The dataset was created through network simulations that included both benign traffic (TCP, UDP, ICMP) and various types of malicious traffic, such as TCP Syn, ICMP, and UDP Flood attacks.

Table 4 provides a detailed description of the benchmark dataset, highlighting its characteristics that closely mimic a real-world SDN network environment. To ensure the quality and effectiveness of the dataset, thorough preprocessing is essential before feeding it into the DL model.

Table 4. Benchmark Dataset Description.

Specification	Details
Total Number of Records	175,305
Number of Attack Records	43,446
Number of Normal Records	131,859
Type of Category	Attack and Normal.
Normal Classes	TCP, UDP, and ICMP
Abnormal Classes	ICMP, TCP Syn, and UDP Flooding Attacks
Total Number of Features in the Dataset	22 features
The Calculated Features	- Packet rate is the number of packets sent per second and is calculated by dividing the packet per flow by monitoring interval. - Packet per flow, which is the packet count during a single flow. - Total flow entries in the switch. - A number of Packet_ins messages. - Port Bandwidth is the sum of tx_kbps and rx_kbps. - Byte per flow is the byte count during a single flow. - tx_kbps, rx_kbps represent the data transfer and receiving rate.
The Features From the Switches	- Switch-id, Packet_count, byte_count, duration_sec, duration_nsec, which is the duration in nanoseconds. - rx_bytes is the number of bytes received on the switch port. - dt field shows the date and time, which has been converted into a number. - tx_bytes is the number of bytes transferred from the switch port. - Total duration is the sum of duration_sec and duration_nsec. - Source IP, Destination IP, Port number.

1. **Data scrubbing** or data cleaning refers to the process of correcting or eliminating data that is duplicated, corrupted, improperly formatted, or incomplete within a dataset.
2. **The data transformation** process involves converting data from one format or structure into another, for example, transforming string data into numerical data so that it applies to DL algorithms.
3. **Data normalization** is to minimize or even exclude duplicated data. The Mix–Max scaler is one of the common methods to normalize the input features or variables. It transforms all features into the range between 0 and 1. To avoid bias caused by features that have been measured at different scales and that do not equally contribute to model fitting, we implement a normalization approach. Specifically, we adopt feature-wise normalization, such as Min–Max scaling, to standardize feature vectors, as depicted in Equation (1).

$$x_{Scale} = \frac{x - x_{min}}{x_{min} - x_{min}} \tag{1}$$

4. **The data-balancing** approach is used to ensure that the two classes in the dataset (normal and attack) are distributed evenly. Table 4, for example, displays an imbalanced class distribution with 131,859 normal records and 43,446 attack records. We utilized the SMOTE oversampling method [33] to boost the number of normal

classes to 131,859 data points in order to address this issue. The accuracy of data categorization is greatly helped by this balancing process.

5.2. Cross-Feature Selection

The main purpose of performing cross-feature selection before utilizing the RNN model is to enhance feature representation, reduce redundancy and noise, capture complex feature interactions, and decrease dimensionality. By selecting the most relevant and informative features, we aim to improve the overall performance of the model in detecting DDoS attacks. Cross-feature selection helps to identify the features that have the highest predictive power and are most relevant to the task at hand. This process eliminates irrelevant or redundant features, which can introduce noise and negatively impact the model's performance. By focusing on the most informative features, the model can better capture the underlying patterns and relationships in the data.

The core component of the proposed approach is the mechanism for cross-feature selection, which identifies a set of important features from the dataset. This selection process plays a crucial role in DDoS attack detection on the SDN controller. Extracting, ranking, and selecting relevant and significant features from network traffic is essential for modeling network behaviors, particularly those exhibited by botnet-triggered DDoS attacks.

The cross-feature selection phase consists of three steps: (i) ranking features based on IGR, (ii) ranking features using the Chi-square method, and (iii) cross-features of the top-ranking features. By employing multiple feature selection algorithms, it is possible to achieve a consensus on the most important features that significantly contribute to the identification of DDoS attacks on the SDN controller. This approach effectively reduces the number of relevant features while preserving the detection accuracy. Let (fi) denote all features present in the dataset as follows:

$$\begin{gathered}(fi) = \{switch,\ dt,\ src,\ dst, bytecount,\ dur,\ pktcount\ dur\_nsec,\ tot\_dur, flows,\\ packetins, pktperflow,\ yteperflow, Pairflow,\ pktrate\ Protocol,\ port\_no,\ tx\_bytes,\\ rx\_bytes, \_kbpsrx\_kbps,\ tx\_kbpstot\}.\end{gathered} \tag{2}$$

Thereby, applying IGR and Chi-square on (fi) resulted on IGR(f') and CHI(f'): $IGR(f') = fi = \{byteperflow, bytecount,\ pktcount,\ pktperflow,\ pktrate,\ dt,\ tot\_dur, packetins, tx\_bytes, dur\}$ *and* $CHI(f') = fi = \{bytecount,\ pktcount,\ Protocol,\ flows,\ dt, dst,\ Pairflow, dur,\ tot\_dur,\ rx\_bytes\}$.

Then, the features intersection chooses the features that appear in both the feature ranking steps' outputs, IGR(f') and CHI(f'), and the remaining features are not included. The intersection methods output between CHI(f') and IGR(f') is presented as follows using Equation (2):

$$\text{CHI}(f') \bigcap \text{IGR}(f') = ff_{\text{inal}} = \{bytecount, pktcount,\ dt,\ tot\_dur,\ dur\} \tag{3}$$

In a nutshell, the ff_{inal} is used as input for the next phase. Figure 6 depicts the cross-feature selection mechanism process.

5.3. RNN-Based DDoS Attack Detection Model

In this phase, RNN is trained to build a detection model that can effectively identify DDoS attacks on the SDN controller using the selected cross-features f_{final}. The RNN-based DDoS attack detection process employs the target function $f(x) = y_i$, where y_i takes values from the set 0, 1. In this set, 0 represents the normal class, while 1 represents the attack class. The main process of the RNN is illustrated in Figure 7.

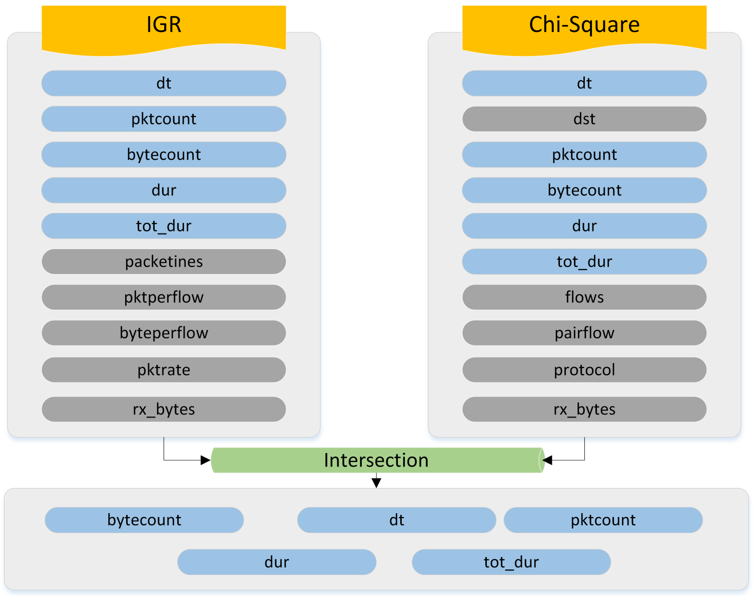


Figure 6. Cross-feature selection mechanism process.

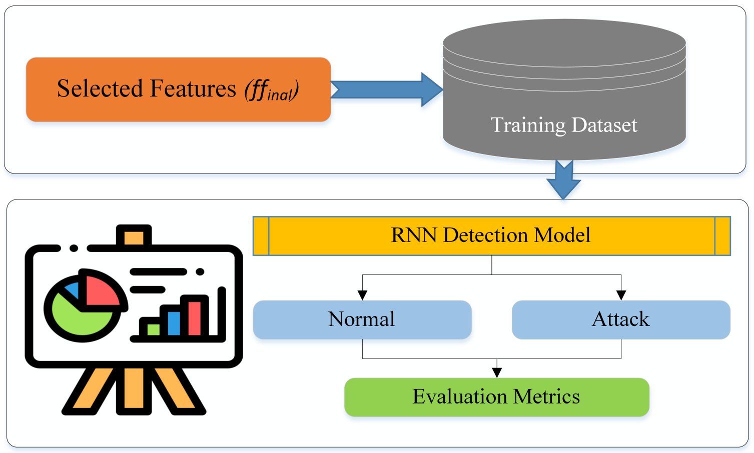


Figure 7. Process of RNN-based detection model.

6. Results of Experiment

Here, we explain the experimental outcomes and provide the metrics used to assess the proposed approach.

6.1. Evaluation Metrics

We evaluated the effectiveness of the proposed approach by measuring its detection accuracy, false positive rate (FPR), precision, and F1 measure. These metrics were calculated based on the confusion matrix as presented in Table 5.

Table 5. Matrix of Confusion.

	Class Prediction	
	Normal	**Attack**
Actual Class	False Negative (FN)	True Positive (TP)
	True Negative (TN)	False Positive (FP)

Where **True Positive (TP)** means that the classifier identified the attack accurately, **False Negative (FN)** represents occasions when an attack was mistakenly labeled to be normal by the classifier, **False Positive (FP)** means that the classifier has mistakenly labeled a benign instance as malicious, and **True Negative (TN)** represents instances in which the classifier appropriately classified normal instances. Additionally, other evaluation metrics, such as precision, false alarm rate (FAR), detection accuracy (DA), and F-measure, are also used by researchers in their studies as described below:

- **FAR (False Alarm Rate)**: This metric is used to determine how many attack samples were mistakenly predicted relative to the total number of normal samples. The following Equation (4) is used to calculate FAR:

$$\text{FAR} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{4}$$

- **Precision**: This metric represents the percentage of attacks successfully predicted from a set of test samples. Equation (5) can be used to calculate precision:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

- **F1-Measure**: This metric is the harmonic mean of precision and recall and is used to evaluate the accuracy of a system. The below Equation (6) is used to calculate the F1 measure:

$$\text{F1 Measure} = 2 \times (\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}) \tag{6}$$

- **DA (Detection Accuracy)**: The proportion of correctly labeled cases relative to the total number of instances is what this indicator measures. However, it is only useful when a balanced dataset is used. The accuracy can be calculated using Equation (7):

$$\text{DA} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{7}$$

The performance evaluation metrics utilized in this study are widely accepted in the field to assess IDS performance [34]. Hence, the proposed approach is evaluated using all these metrics.

6.2. Result of Cross-Feature Selection Mechanism

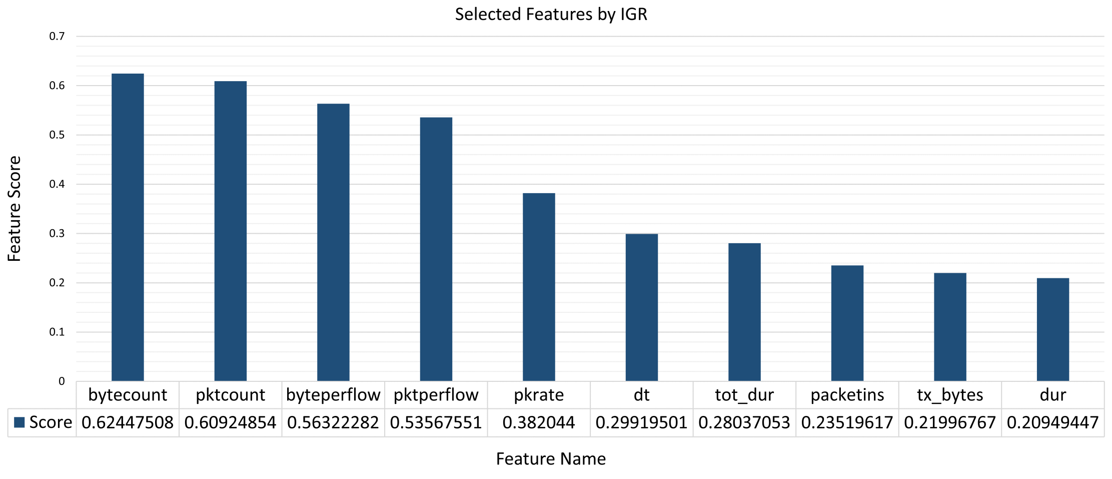
Using relevant features for training the detection model can have a significant impact on the effectiveness of the approach. Consequently, 22 features are selected for input into the IGR and Chi-square algorithms. The ranking of features based on IGR and Chi-square scores is presented in Table 6.

Table 6 shows that all 22 features are assigned two score values. The first score value is assigned by the IGR algorithm, while the Chi-square algorithm assigns the second score value. Then, 10 out of 22 features were selected based on the highest scores. Figure 8 presents the top 10 features selected by IGR, while Figure 9 presents the top 10 features ranked by Chi-square.

We can observe that the top 10 features selected by IGR are bytecount, pktcount, byteperflow, pktperflow, pkrate, dt, tot_dur, packetins, tx_bytes, and dur. The highest feature selected by IGR is bytecount with a feature score of 0.62447508. Regarding the features selected by Chi-square, the top 10 features are bytecount, pktcount, protocol, flows, dt, dst, pairflow, dur, tot_dur, and rx_bytes. The highest feature selected by Chi-square is bytecount with a feature score of 32.6835706.

Table 6. Ranking of features based on IGR and Chi-square scores.

Number	Features Name	IGR	Chi-Square
1	bytecount	0.0.62447508	32.6835706
2	byteperflow	0.56322282	30.8833236
3	dst	0.03520766	29.1430015
4	dt	0.29919501	6.82629011
5	dur	0.20949447	3.30824831
6	dur_nsec	0.12252036	2.13023575
7	flows	0.03502166	1.88301362
8	packetins	0.23519617	1.40931663
9	pairflow	0.01022723	1.40733114
10	pktcount	0.60924854	0.945133119
11	pktperflow	0.53567551	0.895645311
12	pktrate	0.382044	0.712864682
13	$port_no$	0.00218732	0.3840113
14	protocol	0.05085547	0.148766223
15	rx_bytes	0.20313028	0.053182109
16	rx_kbps	0.112327	0.033016546
16	src	0.09571495	0.026685311
18	switch	0.00341777	0.021747427
19	tot_dur	0.28037053	0.02169964
20	tot_kbps	0.14311258	0.016125104
21	tx_bytes	0.21996767	0.006544647
22	tx_kbps	0.09093322	0.000388616



	bytecount	pktcount	byteperflow	pktperflow	pkrate	dt	tot_dur	packetins	tx_bytes	dur
■ Score	0.62447508	0.60924854	0.56322282	0.53567551	0.382044	0.29919501	0.28037053	0.23519617	0.21996767	0.20949447

Figure 8. Top 10 Features Selected by IGR.

Based on Figures 8 and 9, $CHI(f') = \{bytecount,\ pktcount,\ byteperflow,\ pktperflow,\ pktrate, dt,\ tot\_dur,\ packetins, tx\_bytes,\ dur\}$ and $CHI(f') = \{bytecount,\ pktcount,\ Protocol, flows\, dt,\ dst,\ Pairflow,\ durtot\_durrx\_bytes\}$. As a result, $IGR(f') \bigcap CHI(f') = ff_{inal}$. Therefore, $ff_{inal} = \{bytecount,\ pktcount,\ dt, tot\_dur,\ dur\}$, as shown in Table 7.

Table 7. Feature Selection Results.

No.	Feature Name	IGR	Chi-Square
1	bytecount	0.624475	32.68357
2	dt	0.299195	3.308248
3	dur	0.209494	1.409317
4	pktcount	0.609249	30.88332
5	tot_dur	0.280371	1.407331

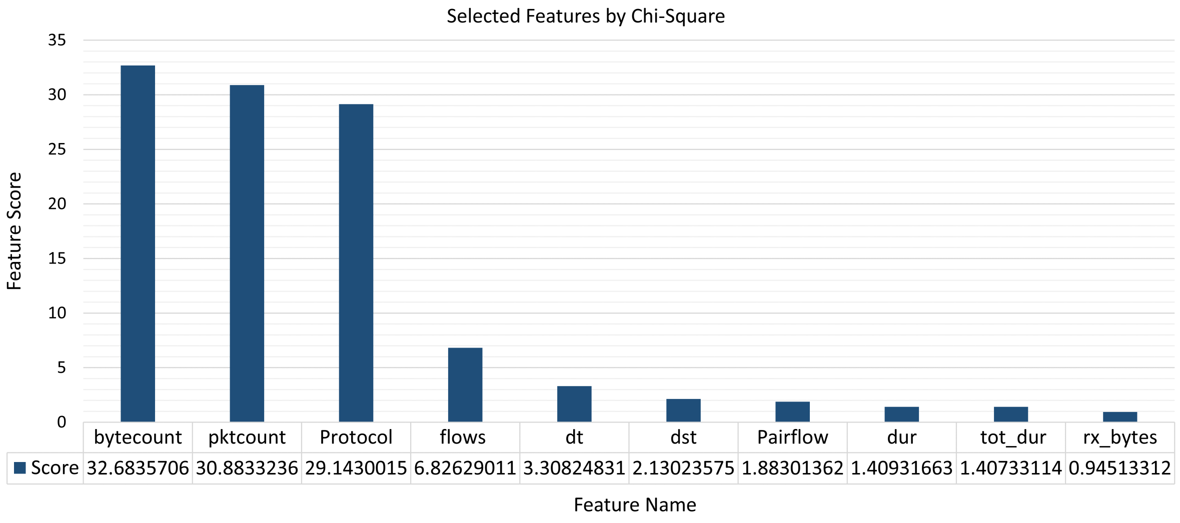


Figure 9. Top 10 Features Selected By Chi-Square.

As shown in Table 7, the proposed cross-feature selection selected the significant feature f_{final} that contributes to detecting DDoS attacks on SDN. Additionally, it reduces the number of features from 22 to 5, implying a reduction in the processing time required to train the DL model. The features f_{final} are then utilized as inputs for the RNN to generate the detection model.

6.3. RNN Model Setup

In this subsection, we provide detailed information about the learning model configuration and its architecture. RNN has been trained using the selected features f_{final}, and its architecture is displayed in Table 8 and Figure 10. To obtain optimal weight values, we utilized the Adam optimizer in conjunction with a sparse categorical cross-entropy loss function. The learning rate plays a crucial role in deep learning algorithms, as it determines the magnitude of the model's steps during each iteration. We conducted a series of experiments using learning rates of 0.01, 0.001, 0.0001, and 0.00001 to determine the optimal value for the Adam optimizer. Among these options, the learning rate of 0.01 yielded the highest detection rate, making it the preferred choice. To mitigate the risk of overfitting, we implemented an early stopping method. If the validation loss does not decrease after a certain number of iterations, the training process is stopped. To achieve the best results during testing, it is crucial to set the number of epochs in such a way that the network's accuracy no longer improves. Since the RNN model converged in under 100 iterations, we consider it to be the optimal value. The hyperparameter values are determined through experimentation. Furthermore, these hyperparameters are commonly used in existing research, as demonstrated in [35].

Table 8. The performance metrics of the DLADSC.

Run No.	Detection Accuracy %.	False Positive Rate %.	Precision %.	F1-Measure %.
1	94.66	8.76	91.70	94.8
2	94.36	7.79	92.44	94.44
3	94.05	8.40	91.89	94.15
4	93.56	8.18	91.99	93.63
5	94.30	7.44	92.71	94.36

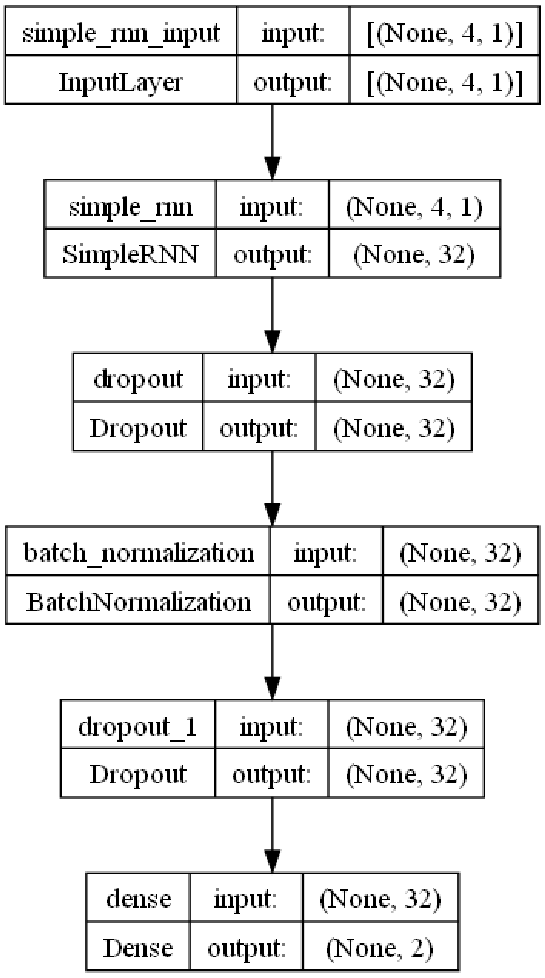


Figure 10. Architecture of RNN Algorithm.

7. Results and Discussion

In this stage, an RNN-based training model is generated using the architectures listed in Table 9, with the f_{final} features. Due to the long training and inference times required by RNN, the model is trained offline to address this issue. Once generated, the detection model is operated online to detect DDoS attacks on the SDN controller. To ensure consistency across all run times, the experiments were repeated five times, as they were based on simulated datasets. Additionally, a cross-validation testing technique is employed. Eighty percent of the data is used for training, while twenty percent is reserved for testing. The 80/20 rule, often referred to as the Pareto principle, is a widely adopted methodology for focusing on the most crucial aspects of a given topic. This approach ensures an efficient and improved research method [36]. In ML classification, a confusion matrix is a valuable tool for performance measurement. It presents a table that defines the performance of a classification algorithm. The confusion matrix provides a visual representation and summary of the classification algorithm's performance, as illustrated in Table 10.

Based on Table 10, the detection accuracy, false positive rate, precision, and F1-measure are calculated for five-run times using Equations (3)–(6), respectively. The results of precision, false positive rate, detection accuracy, and F1-measure after being calculated five times are presented in Table 8.

Table 9. The Architecture of the RNN Algorithm.

Type of Layer	Output	Parameter #
simple_RNN (SimpleRNN)	(None, 32)	1088
dropout (Dropout)	(None, 32)	0
batch_normalization (Batch Normalization)	(None, 32)	128
dropout1(Dropout)	(None, 32)	0
flatten (Flatten)	(None, 32)	0
dense (Dense)	(None, 2)	66
Total params: 1282 Trainable params: 1218 Non-trainable params: 64.0		

Table 10. RNN Confusion Matrix.

Run No.	Classes	Attack	Normal
Run 1	Actual class	12,384	237
		1121	11,683
Run 2	Actual class	12,184	437
		997	11,807
Run 3	Actual class	12,184	437
		1076	11,728
Run 4	Actual class	12,031	590
		1047	11,757
Run 5	Actual class	12,125	496
		953	11,851

Furthermore, Figure 11 demonstrates the efficient detection of DDoS attacks on SDN controllers by the RNN using the f_{final} features, achieving an average detection accuracy of 94.186%, average precision of 92.146%, average false positive rate (FPR) of 8.114%, and average F1-measure of 94.276%.

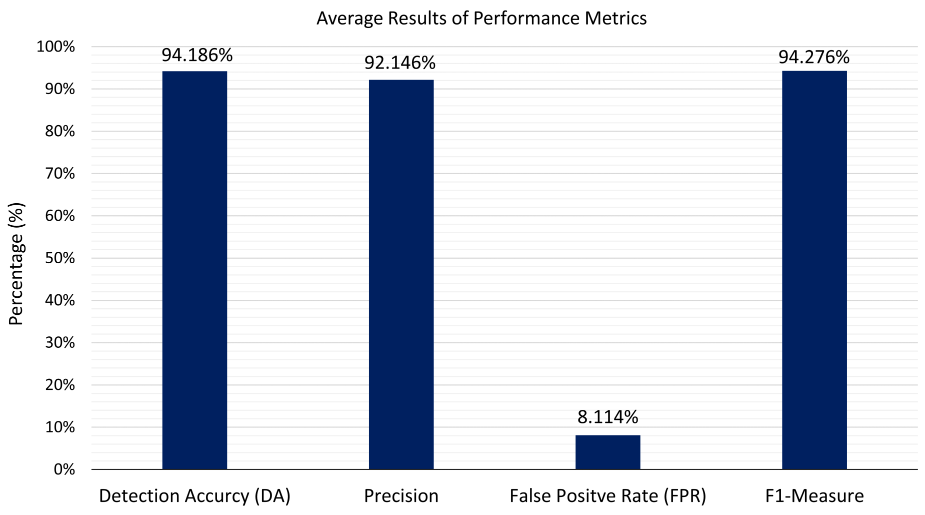


Figure 11. Average Result of Performance Metrics.

Comparison Analysis of DLADSC with Existing Approaches

This subsection aims to present a fair comparison between the proposed approach and existing approaches by evaluating them using the same dataset and testing environment. Table 11 provides a comprehensive comparison of the proposed approach and other approaches based on various metrics, including: (i) the proposed approach, (ii) the type of dataset used, (iii) average detection accuracy, (iv) precision, (v) false positive rate (FPR), (vi) F-measure, and (vii) detection time.

Table 11. Comparison of DLADSC with existing DL approaches in SDN networks.

Metrics	**Proposed Approach**	**Hsieh et al. [26]**	**Boukria et al. [28]**
Approach	DLADSC (RNN)	SPIFFY (CNN)	DL logarithm
Type of Dataset	DDoS attack SDN Dataset	Synthetic LFA	Non-SDN Dataset
Detection Accuracy	94.186%	91.976%	93.203%
Precision	92.146%	91.772%	88.472%
FPR	8.114%	8.138%	12.746%
F-Measure	94.276%	91.932%	93.546%
Detection Time (s)	1.627	1.618	1.681

Based on the comparison tabulated in Table 11, the proposed DLADSC approach achieved remarkable performance with a detection accuracy of 94.186%, precision of 92.146%, FPR of 8.114%, F-measure of 94.276%, and a detection time of 1.627 s. In contrast, Hsieh et al.'s SPIFFY approach achieved slightly lower results with a detection accuracy of 91.976%, precision of 91.772%, FPR of 8.138%, F-measure of 91.932%, and a detection time of 1.618 s. Furthermore, Boukria et al.'s DL logarithm function approach showed comparatively lower performance with a detection accuracy of 93.203%, precision of 88.472%, FPR of 12.746%, F-measure of 93.546%, and a detection time of 1.681 s.

The superior performance of the proposed DLADSC approach can be attributed to the effective cross-feature selection mechanism, which aims to identify the most relevant features (f) for detecting DDoS flooding attacks on the SDN controller. This approach outperformed both the SPIFFY approach and the DL logarithm function approach in terms of detection accuracy, precision, FPR, and F-measure. Additionally, the proposed approach demonstrated a relatively lower detection time, indicating its efficiency and effectiveness in detecting DDoS attacks.

8. Conclusions, Limitations and Future Works

Security vulnerabilities pose significant concerns in SDN networks, as attackers can exploit weaknesses and launch DDoS attacks on SDN controllers. To address this issue, this research paper proposes a new DL-based approach called DLADSC for detecting DDoS attacks on SDN controllers. The proposed approach consists of three stages and effectively detects DDoS attempts. In the first stage, a cross-feature selection method is employed to rank the features based on scores obtained from the Chi-square and IGR algorithms. The top-scoring features are then selected to train the RNN model using the feature intersection mechanism.

The effectiveness of the proposed DLADSC approach is evaluated using four standard metrics: average F1-measure, average detection accuracy, average FPR, and average detection time. The experimental results show that the proposed approach achieves 94.186% for average detection accuracy, 92.146% for precision, 8.114% for average FPR, and 94.276% for average F1-measure. These findings demonstrate that DLADSC can accurately identify DDoS attacks targeting SDN controllers. Furthermore, when compared to other DL-based approaches, DLADSC exhibits significant improvements in performance.

Although the proposed DLADSC approach opens up an exciting avenue for future research in DDoS detection, there are a few limitations and areas for future work. Firstly, it would be beneficial to explore different DL classifiers such as LSTM, GRU, and autoencoders to assess their performance in detecting DDoS attacks on SDN controllers. Secondly, employing various feature selection algorithms could help identify features that enhance the SDN controller's ability to detect DDoS attacks. Lastly, optimizing the hyperparameter values of the RNN model using bio-inspired algorithms such as gray wolf and whale could lead to further improvements in performance.

Author Contributions: Conceptualization, designing, A.M., M.A. and A.A.B.; Methodology, M.A.; Writing the original draft, A.M., M.A. and A.A.B.; Writing—review and editing, A.M., M.A., A.A.B., B.A.A. and S.D.A.R.; Supervision, M.A.; Funding acquisition, B.A.A. and S.D.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: The authors are thankful to the Deanship of Scientific Research at Najran University for funding this work under the Research Groups Funding program grant code (NU/RG/SERC/12/50).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Aladaileh, M.A.; Anbar, M.; Hasbullah, I.H.; Bahashwan, A.A.; Al-Sarawn, S. Dynamic Threshold-Based Approach to Detect Low-Rate DDoS Attacks on Software-Defined Networking Controller. *Comput. Mater. Contin.* **2022**, *73*, 1403–1416. [CrossRef]
2. Zubaydi, H.D.; Anbar, M.; Wey, C.Y. Review on detection techniques against DDoS attacks on a software-defined networking controller. In Proceedings of the 2017 Palestinian International Conference on Information and Communication Technology (PICICT), Gaza, Palestine, 8–9 May 2017; pp. 10–16. [CrossRef]
3. Alanazi, S.T.; Anbar, M.; Karuppayah, S.; Al-Ani, A.K.; Sanjalawe, Y.K. Detection Techniques for DDoS Attacks in Cloud Environment: Review Paper. In Proceedings of the Intelligent and Interactive Computing, Melaka, Malaysia, 17 May 2019; Lecture Notes in Networks and Systems; Springer: Singapore, 2019; pp. 337–354. [CrossRef]
4. Deeb Al-Mo, A.A.; Wan, T.C.; Al-Saedi, K.; Altaher, A.; Ramadass, S.; Manasrah, A.; Melhiml, L.B.; Anbar, M. An online model on evolving phishing e-mail detection and classification method. *J. Appl. Sci.* **2011**, *11*, 3301–3307. [CrossRef]
5. Bahashwan, A.A.; Anbar, M.; Abdullah, N. New architecture design of cloud computing using software defined networking and network function virtualization technology. In Proceedings of the International Conference of Reliable Information and Communication Technology, 2019 (IRICT 2019), Johor, Malaysia, 22–23 September 2019; pp. 705–713. [CrossRef]
6. Mousavi, S.M.; St-Hilaire, M. Early detection of DDoS attacks against SDN controllers. In Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC), Garden Grove, CA, USA, 16–19 February 2015; pp. 77–81. [CrossRef]
7. Aladaileh, M.A.; Anbar, M.; Hintaw, A.J.; Hasbullah, I.H.; Bahashwan, A.A.; Al-Sarawi, S. Renyi Joint Entropy-Based Dynamic Threshold Approach to Detect DDoS Attacks against SDN Controller with Various Traffic Rates. *Appl. Sci.* **2022**, *12*, 6127. [CrossRef]
8. Sudar, K.M.; Beulah, M.; Deepalakshmi, P.; Nagaraj, P.; Chinnasamy, P. Detection of Distributed Denial of Service Attacks in SDN using Machine learning techniques. In Proceedings of the 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 27–29 January 2021; pp. 1–5. [CrossRef]
9. Wan, L.; Wang, Q.; Zheng, S. Deep SSAE-BiLSTM Model for DDoS Detection In SDN. In Proceedings of the 2nd International Conference on Computer Communication and Network Security (CCNS), Xining, China, 30 July–1 August 2021; pp. 1–4. [CrossRef]
10. Alanazi, F.; Jambi, K.; Eassa, F.; Khemakhem, M.; Basuhail, A.; Alsubhi, K. Ensemble Deep Learning Models for Mitigating DDoS Attack in Software-Defined Network. *Intell. Autom. Soft Comput.* **2022**, *33*, 923–938. [CrossRef]
11. Deepa, V.; Sudar, K.M.; Deepalakshmi, P. Detection of DDoS attack on SDN control plane using hybrid machine learning techniques. In Proceedings of the 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 13–14 December 2018; pp. 299–303. [CrossRef]
12. Santos, R.; Souza, D.; Santo, W.; Ribeiro, A.; Moreno, E. Machine learning algorithms to detect DDoS attacks in SDN. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5402. [CrossRef]
13. Goransson, P.; Black, C.; Culver, T. *Software Defined Networks: A Comprehensive Approach*; Morgan Kaufmann: Burlington, MA, USA, 2015.
14. Bahashwan, A.A.; Anbar, M.; Manickam, S.; Al-Amiedy, T.A.; Aladaileh, M.A.; Hasbullah, I.H. A Systematic Literature Review on Machine Learning and Deep Learning Approaches for Detecting DDoS Attacks in Software-Defined Networking. *Sensors* **2023**, *23*, 4441. [CrossRef] [PubMed]

15. Nadeau, T.D.; Gray, K. *SDN: Software Defined Networks: An Authoritative Review of Network Programmability Technologies*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2013.
16. Stallings, W. Software-defined networks and openflow. *Internet Protoc. J.* **2013**, *16*, 2–14.
17. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
18. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
19. Elejla, O.E.; Anbar, M.; Hamouda, S.; Faisal, S.; Bahashwan, A.A.; Hasbullah, I.H. Deep-Learning-Based Approach to Detect ICMPv6 Flooding DDoS Attacks on IPv6 Networks. *Appl. Sci.* **2022**, *12*, 6150. [CrossRef]
20. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. *Handb. Brain Theory Neural Netw.* **1995**, *3361*, 1995.
21. Pan, X.; Yang, Y.; Xia, C.Q.; Mirza, A.H.; Shen, H.B. Recent methodology progress of deep learning for RNA–protein interaction prediction. *Wiley Interdiscip. Rev. RNA* **2019**, *10*, e1544. [CrossRef] [PubMed]
22. Dongare, A.; Kharde, R.; Kachare, A.D. Introduction to artificial neural network. *Int. J. Eng. Innov. Technol. (IJEIT)* **2012**, *2*, 189–194.
23. Karhunen, J.; Raiko, T.; Cho, K. Chapter 7—Unsupervised deep learning: A short review. In *Advances in Independent Component Analysis and Learning Machines*; Bingham, E., Kaski, S., Laaksonen, J., Lampinen, J., Eds.; Academic Press: Cambridge, MA, USA, 2015; pp. 125–142. [CrossRef]
24. Khashab, F.; Moubarak, J.; Feghali, A.; Bassil, C. DDoS attack detection and mitigation in SDN using machine learning. In Proceedings of the 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), Tokyo, Japan, 28 June–2 July 2021; pp. 395–401. [CrossRef]
25. Celesova, B.; Val'ko, J.; Grezo, R.; Helebrandt, P. Enhancing security of SDN focusing on control plane and data plane. In Proceedings of the 2019 7th International Symposium on Digital Forensics and Security (ISDFS), Barcelos, Portugal, 10–12 June 2019; pp. 1–6. [CrossRef]
26. Hsieh, C.H.; Wang, W.K.; Wang, C.X.; Tsai, S.C.; Lin, Y.B. Efficient Detection of Link-Flooding Attacks with Deep Learning. *Sustainability* **2021**, *13*, 12514. [CrossRef]
27. Lee, T.H.; Chang, L.H.; Syu, C.W. Deep learning enabled intrusion detection and prevention system over SDN networks. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
28. Boukria, S.; Guerroumi, M. Intrusion detection system for SDN network using deep learning approach. In Proceedings of the 2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS), Skikda, Algeria, 15–16 December 2019; Volume 1, pp. 1–6. [CrossRef]
29. Akhunzada, A.; Ahmed, E.; Gani, A.; Khan, M.K.; Imran, M.; Guizani, S. Securing software defined networks: Taxonomy, requirements, and open issues. *IEEE Commun. Mag.* **2015**, *53*, 36–44. [CrossRef]
30. Pradhan, A.; Mathew, R. Solutions to Vulnerabilities and Threats in Software Defined Networking (SDN). *Procedia Comput. Sci.* **2020**, *171*, 2581–2589. [CrossRef]
31. Amidi, A.; Amidi, S. *Vip Cheatsheet: Recurrent Neural Networks*; Stanford University: Stanford, CA, USA, 2018 .
32. Ahuja, N.; Singal, G.; Mukhopadhyay, D. DDOS attack SDN dataset. *Mendeley Data* **2020**, *1*. [CrossRef]
33. Fernández, A.; Garcia, S.; Herrera, F.; Chawla, N.V. SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *J. Artif. Intell. Res.* **2018**, *61*, 863–905. [CrossRef]
34. Anbar, M.; Abdullah, R.; Al-Tamimi, B.N.; Hussain, A. A machine learning approach to detect router advertisement flooding attacks in next-generation IPv6 networks. *Cogn. Comput.* **2018**, *10*, 201–214. [CrossRef]
35. Ullah, I.; Mahmoud, Q.H. Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks. *IEEE Access* **2021**, *9*, 103906–103926. [CrossRef]
36. Iqbal, M.; Rizwan, M. Application of 80/20 rule in software engineering Waterfall Model. In Proceedings of the 2009 International Conference on Information and Communication Technologies, Karachi, Pakistan, 15–16 August 2009; pp. 223–228. [CrossRef]