systems

Article

Enhancing Intrusion Detection Systems Using a Deep Learning and Data Augmentation Approach

Rasheed Mohammad [1], Faisal Saeed [1,*], Abdulwahab Ali Almazroi [2], Faisal S. Alsubaei [3,*] and Abdulaleem Ali Almazroi [4]

[1] College of Computing and Digital Technology, Birmingham City University, Birmingham B4 7XG, UK; rasheed.mohammad@bcu.ac.uk
[2] Department of Information Technology, College of Computer Science and Engineering, University of Jeddah, Jeddah 23218, Saudi Arabia; aalmazroi@uj.edu.sa
[3] Department of Cybersecurity, College of Computer Science and Engineering, University of Jeddah, Jeddah 23218, Saudi Arabia
[4] Department of Information Technology, Faculty of Computing and Information Technology in Rabigh, King Abdulaziz University, Rabigh 21911, Saudi Arabia; aaealmazrouy@kau.edu.sa
* Correspondence: faisal.saeed@bcu.ac.uk (F.S.); fsalsubaei@uj.edu.sa (F.S.A.)

Abstract: Cybersecurity relies heavily on the effectiveness of intrusion detection systems (IDSs) in securing business communication because they play a pivotal role as the first line of defense against malicious activities. Despite the wide application of machine learning methods for intrusion detection, they have certain limitations that might be effectively addressed by leveraging different deep learning architectures. Furthermore, the evaluation of the proposed models is often hindered by imbalanced datasets, limiting a comprehensive assessment of model efficacy. Hence, this study aims to address these challenges by employing data augmentation methods on four prominent datasets, the UNSW-NB15, 5G-NIDD, FLNET2023, and CIC-IDS-2017, to enhance the performance of several deep learning architectures for intrusion detection systems. The experimental results underscored the capability of a simple CNN-based architecture to achieve highly accurate network attack detection, while more complex architectures showed only marginal improvements in performance. The findings highlight how the proposed methods of deep learning-based intrusion detection can be seamlessly integrated into cybersecurity frameworks, enhancing the ability to detect and mitigate sophisticated network attacks. The outcomes of this study have shown that the intrusion detection models have achieved high accuracy (up to 91% for the augmented CIC-IDS-2017 dataset) and are strongly influenced by the quality and quantity of the dataset used.

Keywords: data augmentation; deep learning; cyber-attack; intrusion detection; machine learning

check for updates

Citation: Mohammad, R.; Saeed, F.; Almazroi, A.A.; Alsubaei, F.S.; Almazroi, A.A. Enhancing Intrusion Detection Systems Using a Deep Learning and Data Augmentation Approach. *Systems* **2024**, *12*, 79. https://doi.org/10.3390/systems12030079

Academic Editors: Fernando De la Prieta Pintado, Nuno Lopes and Joaquim Gonçalves

Received: 5 December 2023
Revised: 4 February 2024
Accepted: 15 February 2024
Published: 1 March 2024

1. Introduction

Many firms rely heavily on online computing systems to run businesses and thus become targets for cyber-attacks, so they prioritize artificial intelligence-based intrusion detection systems (IDSs) among their network security procedures [1–3]. According to Markevych and Dawson [4], the high complexity of modern cyber-attacks requires more innovations when proposing and applying robust IDSs to protect critical assets and network data by identifying and classifying network traffic and detecting anomalous behavior. With the rapid increase in the amount of available online data, the possibility of being hacked by different intrusion attacks also increases [5]. The application of deep learning and machine learning architectures in IDSs showed interesting results in the accuracy and efficiency of detecting diverse cyber threats in network environments by learning from patterns and anomalies in network traffic.

Intelligent network protection systems have become vital for robust defense against malicious threats [1,6]. Models that predict based on time-series data are classically used for network intrusion detection [3]. However, data featured by time (time series) are known to be comprised of non-linear features due to numerous data entries changing over time as a result of irregular variations [3]. On the other hand, many machine learning methods, for example, k-nearest neighbors, support naïve Bayes, and vector machines, are utilized for network-based intrusion detection systems (NIDS) [3,6]. However, these machine learning methods commonly utilize feature engineering, which makes them less suitable candidates for real-time usage, with a low detection rate [3,6]. Recently, there has been a prevalent utilization of deep learning methods, for instance, convolutional neural networks (CNNs) and recurrent neural networks together with NIDS (Vigneswaran et al., 2018). However, deep learning-based models are still under investigation for practical utilization as a result of several issues, including high false-positive rates [3].

Different datasets have been assessed in several models; for instance, Vigneswaran et al. [6] used the KDDCup-'99′ dataset with a deep learning model and used the UNSW-NB15 dataset, released by Nour and Slay [7], for evaluating the proposed models in their work. This dataset is known to overcome the shortcomings of the KDD98 and KDD99 datasets [8].

This work, which assesses various deep learning models, mainly relies on a convolutional neural network (CNN) to combine temporal and spatial features. The architecture is designed to effectively capture and process the inherent sequential patterns within the input data. A prediction was first made for binary classification (whether there is an attack or not), and then the category of the attack was given as the second prediction. For multi-category attack prediction, 10 categories were utilized (related to the UNSW-NB15 dataset) as follows: normal, denial of service, exploits, reconnaissance, generic, worms, analysis, shellcode, backdoor, and fuzzers, while there are 15 classes in the CIC-IDS-2017 dataset.

It was noticed that the number of entries of attacks was quite low compared to the normal network flows; accordingly, the accuracies reported related to attack class prediction were also quite low. This work aims to apply data augmentation in order to fairly assess the performance of intrusion detection using several deep learning-based methods. Moreover, many other studies have used a part of the CIC-IDS-2017 dataset, whereas this work used the whole CIC-IDS-2017 dataset. Therefore, the key contributions of this work are the utilization of four datasets, UNSW-NB15, CIC-IDS-2017, 5G-NIDD, and FLNET2023, for intrusion detection systems, applying an oversampling technique to emphasize that every class (particularly attack classes) has an equivalent number of entries, applying data augmentation to fairly evaluate the performance of the models developed, and finally applying five deep learning architectures with augmented versions of the used datasets.

The rest of the paper is organized as follows: Section 2 discusses the related works on several intrusion detection systems for cybersecurity applications. Section 3 highlights the Materials and Methods of this study, while the Results and Discussion are presented in Sections 4 and 5, respectively. The paper is concluded in Section 6.

2. Related Works

Intrusion detection systems can be categorized into pattern-matching or artificial intelligence (AI) anomaly detection approaches [2,3,9]. However, both approaches have some weaknesses. For instance, pattern matching usually demonstrates high false-positive rates, while AI-based solutions rely on specifying features to foresee the likelihood of an attack [3]. Furthermore, machine learning methods, such as support vector machine (SVM) and KNN, suffer from an inadequate group of features and accuracy, as well as higher false-positive rates, as reported in [2,3]. However, the work of Rodríguez et al. [2] with the CIC-IDS-2017 dataset and machine learning methods (multilayer perceptron, naïve Bayes, logistic, sequential minimal optimization, adaptive boosting, k-nearest neighbors, J48, PART, OneR, and random forest) reported high accuracy. Similarly, the work of Vigneswaran et al. [6] reported the high accuracy of the machine learning methods tested with the UNSW-NB15 dataset. In contrast, machine learning methods with NIDS suffer from overfitting in addition to low accuracy when predicting attack categories, and a smaller proportion of the data is available for consideration as compared to the satisfactory quantities required in the categories [3]. The classic machine learning solutions prioritize feature availability, dimensionality reduction, and learning feature importance techniques for the sake of finding the optimum correlation between the data points that influence the final outcomes while entirely focusing on the significance of the association between the data features and paying attention to the time-steps for the sake of highly accurate prediction. Accordingly, deep learning-based solutions have been adopted to overcome the deficiencies of the proposed machine learning methods.

Vinayakumar et al. [10] used recurrent neural networks (RNNs) with the datasets NSL-KDD and UNSW-NB15 because they include time-step data, which conveniently justifies the utilization of RNNs—a logical choice—as RNN is classically employed to analyze data characterized as a time series to acquire a long-range temporal feature set [3,11]. However, RNNs are exposed to a vanishing gradient issue in which the feature set learned at the beginning of the model ends up having minimum impact on the end result of the model [3,12]. To overcome this issue, long short-term memory (LSTM) [13] was used. Vinayakumar et al. [10] used various CNN and GRU architectures for the multi-class prediction of NIDS with the UNSW-NB15 dataset; the achieved accuracy was 64.8% and 67.5% by GRU and LSTM, respectively. The work of Ieracitano et al. [1] used statistical analysis to prioritize the features that should be fed into a deep autoencoder (AE) for potential threat detection. The model recorded an 87% accuracy with NSL-KDD.

Khan et al. [14] reported that the 1D-CNN (3 layers) has the possibility to deliver better learning features if the TCP/IP packets are serialized in a specific range (of time) for operative classification. The experiment with the UNSW-NB15 database achieved a 91% accuracy. The prediction was binary classification. Wang et al. [15] used CNNs and RNNs, where CNNs were employed to process the learning of spatial features, while the RNN processed the temporal features existing in the long-range time-series data. The model designed by Wang et al. [15] was hierarchical and comprised of several CNN layers tailed by two LSTM layers. The model of Wang et al. [15] was evaluated with DARPA and ISCX2012 datasets (versions of KDD99, where attacks are recategorized into five classes), and accuracies of 50% and 97%, respectively, were achieved.

Su et al. [9] and Yang [16] built models based on BiLSTM, where Su et al. [9] used BiLSTM with an attention layer to filter the minimal-impact features. The model was built with various convolutional layers incorporating the attention and fully connected dense layer, BiLSTM, and tested on an NSL-KDD dataset (which has five network attack classes), where the accuracy was within the range of 82–84%. Yang [16] proposed a two-Bi-LSTM-layered architecture and tested it on the UNSW-NB15 dataset; the results showed 0.86 as the average F1 score. Finally, Ayantayo et al. [17] proposed three varied deep learning models, referred to as late fusion, early fusion, and late ensemble learning models, which incorporated feature fusion within fully connected deep networks. These models were evaluated using the NSL-KDD and UNSW-NB15 datasets, considering significant tuning parameters during the assessment. The accuracy of the best performance with UNSW-NB15 was approximately 84%.

Overall, deep learning and machine learning are both valuable approaches for time-series analysis, which can be applied to intrusion detection systems, but their relative effectiveness depends on several factors, including the complexity of the problem, the amount of data available, and the quality of feature engineering. Deep learning methods, specifically their variants, such as long short-term memory (LSTM) networks and recurrent neural networks (RNNs), have gained significant attention regarding time-series analysis, including network flow packets, due to their ability to capture complex temporal patterns automatically. However, they are not always superior to traditional machine learning methods, such as random forests, decision trees, or gradient boosting, which may outperform deep learning when dealing with time-series data characterized by simpler patterns or when the dataset is small. Feature engineering and domain knowledge play a vital role in this context, as they enable machine learning models to extract relevant information effectively [18]. Moreover, machine learning methods can be computationally less demanding, making them suitable for real-time or resource-constrained applications [19]. The choice between deep learning and machine learning for time-series analysis should be driven by the specific problem and available resources, and a hybrid approach that combines the strengths of both methods may often be the most effective [20].

3. Materials and Methods

The initial proposed model consists of the following key components: the input layer that accepts sequences of data, each comprising 196 data points along a single dimension, and then two successive 1D convolutional layers were utilized to extract the relevant features from the input sequences. The first convolutional layer utilized 128 filters with a kernel size of 5, tailed by a rectified linear activation function (ReLU). A max-pooling layer with a pool size of 2 was applied to minimize the spatial dimensions of the feature maps. The second convolutional layer, with 64 filters and the same kernel size and activation function, further refined the learned features. Another max-pooling layer was employed to down-sample the feature maps. The flattened layer, the outcome of the convolutional and pooling layers, was flattened into a one-dimensional vector, enabling seamless integration with the subsequent fully connected layers. The fully connected layers comprised a series of densely connected layers that followed the flattening process; the first fully connected layer comprised 128 neurons with a ReLU. To mitigate overfitting, a dropout layer with a rate of 0.5 was introduced, aiding the network's robustness. A second fully connected layer with 64 neurons and dropout was included, ensuring feature abstraction and generalization. The final layer, comprising 10 neurons, employed a SoftMax activation function to produce probability distributions over the classes. A custom learning rate scheduler was incorporated to dynamically adjust the learning rate during training. This allowed for efficient convergence and improved generalization. The model utilized the Adam optimizer with an initial learning rate of 0.001. During training, the model was subjected to the learning rate scheduler, which applied a decreasing factor to the learning rate after the initial epochs. The above model was modified to create other models with different deep learning architectures to measure the performances of those architectures on four large datasets. The other models tested the intrusion detection when more complicated CNN architecture was applied; for instance, GRU was combined with a CNN, and LSTM was combined with a CNN.

3.1. Datasets

We used four benchmark datasets that were used previously in several studies. The first dataset, UNSW-NB15, was developed and released by the University of New South Wales in 2015 and has been widely used since [3,21]. The UNSW-NB15′s entries comprise a wider diversity of attack types, feature quantity, and distinct IP addresses intended for recreation and data collection [7,8,22]. This dataset consists of a mixture of the real recent normal and current synthesized attack events of the traffic in the computer network [3,22], and this dataset is balanced [22]. Although UNSW-NB15 is smaller in size, it has less redundancy and is adequate to train models with high accuracy [22]. UNSW-NB15 has 10 categories: one is a normal network flow, and nine are attack-based flows (as shown in Table 1).

The second dataset is CIC-IDS-2017 [23], which was published by the Canadian Institute of Cybersecurity to support researchers with an alternative dataset that includes rich variations compared to the various shortcomings of the existing datasets [21]. It comprises bidirectional traffic flows (and thus captures the entire traffic) [2] and more than 2.3 million records, encompassing normal traffic, infiltration, DoS, web attacks, brute force, and scanning attacks [2,21], as shown in Table 1. While the CIC-IDS-2017 dataset provides a valuable resource for intrusion detection research, it is essential to consider its advantages and limitations when applying deep learning models. Accordingly, preprocessing methods to address class imbalance and other issues were conducted to harness the dataset's potential effectively. Class imbalance was found to significantly influence the accuracy and performance of the deep learning models [24].

Table 1. A list of features in the datasets.

Category (UNSW-NB15)	Quantity (UNSW-NB15)	Category (CIC-IDS-2017)	Quantity (CIC-IDS-2017)
		Heartbleed	11
		Infiltration	36
Normal	93,000	DoS Hulk	231,072
Analysis	2677	Web Attack—XSS	652
Fuzzers	24,246	Benign	2,359,087
Backdoor	2329	DoS GoldenEye	10,293
DoS	16,353	PortScan	158,930
Generic	58,871	Web Attack—SQL Injection	21
Exploit	44,525	Bot	1966
Reconnaissance	13,987	DDoS	41,835
Shellcode	1511	FTP-Patator	7938
Worms	174	Web Attack—Brute Force	1507
		SSH-Patator	5897
		DoS Slowhttptest	5499
		DoS Slowloris	5796
Overall	257,673		2,830,540

In order to not miss the important features, all were used due to the fact that different identification patterns, when there are different network attacks corresponding to a set of activities, can lead to network security hazards if left undetected [2].

For the aim of finding a generalizable perspective about employing deep learning models intended for intrusion detection, the applied deep learning architectures have been tested with two recent network intrusion datasets, namely 5G-NIDD [25] and FLNET2023 [26].

3.2. Data Preprocessing and Augmentation

To prepare the datasets for deep learning-based intrusion detection, two primary data preprocessing steps were undertaken as follows: normalization of numeric columns and one-hot encoding of the categorical features. The numeric columns were subject to min–max normalization to rescale the data within a domain of (0, 1). This normalization minimizes redundancy and reduces the training time for the models. For the categorical features, one-hot encoding was preferred over label encoding to avoid misinterpretation of the numerical values as ordinal, which could adversely impact classification [3].

The datasets initially presented a significant challenge due to the class imbalance, particularly in categories such as worms, analysis, infiltration, SQL injection, and shellcode. Such an imbalance negatively impacts the predictive accuracy of deep learning models for intrusion detection. To address this, oversampling techniques were employed. Oversampling ensures that each class, especially the minority attack classes, has an equivalent number of entries, enhancing the balancing of the dataset [3].

Intriguingly, the challenge of class imbalance extended to the testing phase. It was observed that the accuracy of predicting minority attack types was markedly low, nearly approaching zero. To counter this and to ensure a fair assessment of the model performance, data augmentation was applied to the testing portion of the dataset.

A key technique employed in this context was the synthetic minority oversampling technique (SMOTE) [27,28]. SMOTE is a sophisticated method for generating synthetic instances to balance datasets. It works by identifying instances of the minority class and synthesizing new and similar instances. This approach effectively mitigates the issue of class skewness, ensuring that minority classes are adequately represented [27,28]. The oversampling process of SMOTE involves creating synthetic samples that are a combination of the feature space similarities of existing minority instances. This allows for a more nuanced and representative sample of the minority class, avoiding the pitfalls of simple duplication. Although SMOTE has been criticized for lower performance in high-dimensional datasets [28], in this study, where the number of features in the two datasets was not considerably high, the performance of SMOTE was sufficient. By applying SMOTE, the augmented dataset attained a more balanced class distribution, which significantly contributed to enhancing the robustness and generalization capability of the developed models. This was particularly evident in the improved performance in detecting under-represented attack types, demonstrating the efficacy of SMOTE in handling imbalanced intrusion detection datasets.

The transformed and augmented datasets that were prepared by these preprocessing and data augmentation techniques established a solid foundation for the subsequent development and training of deep learning models for effective and accurate intrusion detection.

3.3. Deep Learning Architectures

Several deep learning architectures were applied in this study, including variations of convolutional neural networks (CNNs) combined with long short-term memory (LSTM) and gated recurrent unit (GRU) layers. These architectures were designed to address the specific challenges posed by the CIC-IDS-2017 dataset. Below, Tables 2–6 detail these architectures, highlighting their layer compositions and parameter configurations. Each table provides a detailed breakdown of the layers used in the respective architectures, along with their functions and importance in the context of intrusion detection. The inclusion of GRU and LSTM layers, particularly in a bidirectional configuration, is crucial for their ability to effectively process and learn from sequential data, a key aspect of network traffic patterns in intrusion detection systems.

Table 2. Layers and parameters of CNN Architecture 1.

Layer Type	Output Shape	Parameters	Explanation
Conv1D	(192, 128)	768	Convolutional layer processing 1D input with 128 filters.
MaxPooling1D	(96, 128)	0	Reduces dimensionality for computational efficiency.
Conv1D	(92, 64)	41,024	Second Conv1D layer with 64 filters for feature extraction.
MaxPooling1D	(46, 64)	0	Further dimensionality reduction.
Flatten	(2944)	0	Flattens input for the dense layer.
Dense	(64)	188,480	Fully connected layer for classification.
Dropout	(64)	0	Prevents overfitting by randomly dropping units.
Dense	(10)	650	Output layer with 10 units for class prediction.

Table 3. Layers and parameters of CNN Architecture 2.

Layer Type	Output Shape	Parameters	Explanation
Conv1D	(192, 128)	768	Convolutional layer for feature extraction with 128 filters
MaxPooling1D	(96, 128)	0	Reduces the spatial dimensions for efficiency.
Conv1D	(92, 64)	41,024	Second convolutional layer with 64 filters for deeper feature extraction.
MaxPooling1D	(46, 64)	0	Further reduces dimensions and helps prevent overfitting.
Flatten	(2944)	0	Flattens the 2D feature maps into a 1D vector for the dense layer.
Dense	(128)	376,960	Fully connected layer to learn non-linear combinations of features.
Dropout	(128)	0	Drops out units randomly to prevent overfitting.
Dense	(64)	8256	Another dense layer for further processing of features.
Dropout	(64)	0	Additional dropout layer for regularization.
Dense	(10)	650	Final output layer with 10 units for class prediction.

The output layer of this architecture was adapted for the CIC-IDS-2017 dataset to have 15 nodes, corresponding to 15 possible outcomes.

For the CIC-IDS-2017 dataset, the output layer was adapted to have 15 nodes, corresponding to 15 potential outcomes.

Table 4. Layers and parameters of CNN and GRU Architecture 3.

Layer Type	Output Shape	Parameters	Explanation
Conv1D	(196, 64)	4160	Initial convolutional layer with 64 filters for feature extraction.
MaxPooling1D	(19, 64)	0	Reduces dimensionality for computational efficiency.
BatchNormalization	(19, 64)	256	Normalizes the output of the previous layer to speed up training.
GRU	(64)	24,960	GRU layer for processing temporal sequence data.
Reshape	(64, 1)	0	Reshapes the output for compatibility with subsequent layers.
MaxPooling1D	(6, 1)	0	Further reduces sequence length for efficiency.
BatchNormalization	(6, 1)	4	Further normalization to stabilize learning.
GRU	(128)	50,304	Second GRU layer for more complex sequence modeling.
Dropout	(128)	0	Prevents overfitting by randomly omitting units.
Dense	(10)	0	Output layer for class prediction with 10 units.

Table 5. Layers and parameters of CNN, GRU, and LSTM Architecture 4.

Layer Type	Output Shape	Parameters	Explanation
Conv1D.	(196, 64)	4160	Convolutional layer for initial feature extraction.
MaxPooling1D	(19, 64)	0	Dimensionality reduction.
BatchNormalization	(19, 64)	256	Normalizes layer inputs to accelerate training.
GRU	(64)	24,960	GRU layer for temporal data processing.
Reshape	(64, 1)	0	Prepares data for subsequent bidirectional layer.
MaxPooling1D	(6, 1)	0	Reduces sequence length for computational efficiency.
BatchNormalization	(6, 1)	4	Further normalization.
Bidirectional	(256)	133,120	LSTM layer wrapped in a bidirectional layer for enhanced temporal feature extraction.
Dropout	(256)	0	Mitigates overfitting.
Dense	(10)	2570	Output layer for classification.
Activation	(10)	0	Output layer activation function.

Table 6. Layers and parameters of CNN and LSTM Architecture 5 with early dropout architecture.

Layer Type	Output Shape	Parameters	Explanation
Conv1D	(76, 64)	2112	Initial convolutional layer with 64 filters for extracting basic features from the input.
MaxPooling1D	(15, 64)	0	Reduces the dimensionality of the features, which helps in computational efficiency and mitigating overfitting.
BatchNormalization	(15, 64)	256	Normalizes the activations from the previous layer, stabilizing the learning process.
Bidirectional	(128)	66,048	LSTM layer wrapped in a bidirectional layer to process the sequential data in both forward and reverse directions for enhanced temporal understanding.
Reshape	(128, 1)	0	Reshapes the output for compatibility with subsequent layers.
MaxPooling1D	(25, 1)	0	Further reduces the length of the sequence, focusing on the most significant features.
BatchNormalization	(25, 1)	4	Additional normalization to ensure smooth training.
Bidirectional	(256)	133,120	Second bidirectional LSTM layer for deep and complex sequence modeling.
Dropout	(256)	0	Implemented early in the network to prevent overfitting by randomly dropping units.
Dense	(4)	1028	Dense layer for processing the extracted features before the final classification.
Activation	(10)	0	Applies the output layer activation function for final class prediction.

For the CIC-IDS-2017 dataset, modifications were made to the output layer, increasing it to 15 nodes for a comprehensive outcome classification.

Bidirectional, here, refers to a bidirectional wrapper applied to an LSTM layer, enhancing its ability to process the sequential data in both the forward and reverse directions. For the CIC-IDS-2017 dataset, the output layer was modified to have 15 nodes.

In this architecture, the bidirectional LSTM layers are key to processing time-series data effectively, capturing patterns from both the past and future contexts. For the CIC-IDS-2017 dataset, the output layer was modified to have 15 nodes to accommodate the various types of outcomes required for accurate intrusion detection.

3.4. Model Training

Stratified K-cross-fold validation was conducted to rearrange the data to emphasize that each fold was a good representation of all of the attack classes. Stratified K-cross-fold validation divides the dataset records into K sets, and the models use K-1 folds for training and are validated on the Kth fold. This process continues until all the folds have been employed to validate the model only once [3]. This drives parameter fine-tuning and supports models in categorizing the attacks effectively. Finally, the K-cross-fold method was selected, as it performs better than other methods in terms of computational power [3]. Additionally, when K-cross-fold validation was replaced by other split methods, insignificant changes were noticed with levels of accuracy.

4. Results

This section delves into the performance evaluation of various deep learning architectures on the following four datasets: UNSW-NB15, CIC-IDS-2017, 5G-NIDD, and FLNET2023, focusing on both binary and multiple classifications.

4.1. Binary Classification

For binary classification (attack vs. no attack), the simple CNN Architecture 1 (Table 2) exhibited high accuracy, achieving approximately 95% on the UNSW-NB15 dataset. This indicates the model's strong capability to distinguish between normal and anomalous network behaviors. Similarly, in binary classification (post-augmentation) for the CIC-IDS-2017 dataset, the accuracy was remarkably high at approximately 96%, suggesting that the models were more accurate at binary classification (where we have a generalized attack vs. no attack scenario) than the multi-class classifications.

4.2. Multiple Classifications

The results presented in Table 7 are the results of CNN Architecture 1 of all the folds before augmentation. In a more complex scenario involving multiple classes, the accuracy of CNN Architecture 1 on the UNSW-NB15 dataset without augmentation stood at 82% (as shown in Table 7). This drop in accuracy reflects the increased difficulty in distinguishing between multiple types of attacks compared to a binary classification.

Table 7. CNN Architecture 1 classification performance using UNSW-NB15 dataset before data augmentation.

Classes	Precision	Recall	F1 Score
Analysis	0.79	0.03	0.05
Backdoor	0.65	0.09	0.16
DoS	0.62	0.05	0.09
Exploits	0.60	0.05	0.09
Fuzzers	0.73	0.48	0.58
Generic	1.00	0.98	0.99
Normal	0.88	0.95	0.92
Reconnaissance	0.87	0.75	0.57
Shellcode	0.61	0.53	0.57
Worms	0.29	0.63	0.40
Accuracy	0.82		

However, for the UNSW-NB15 dataset, the data augmentation led to a decrease in the overall accuracy of CNN Architecture 1 to 74% (as shown in Table 8). However, the precisions of many minor classes (such as worms, shellcode, reconnaissance, fuzzers, and exploits) were enhanced, and the values of recall and the F1 score were improved considerably. This outcome could be attributed to the augmented dataset presenting a more challenging and varied set of samples, thus making classification more complex. It underscores the need for more sophisticated or tailored models to handle augmented datasets effectively. The other deep learning architectures showed a similar performance to the simple CNN architecture on this dataset, i.e., enhancing the classification performance for minor classes in terms of precision, recall, and the F1 score.

Table 8. CNN Architecture 1 classification performance using UNSW-NB15 dataset after data augmentation.

Classes	Precision	Recall	F1 Score
Analysis	0.65	0.30	0.41
Backdoor	0.40	0.75	0.52
DoS	0.44	0.53	0.48
Exploits	0.79	0.50	0.61
Fuzzers	0.77	0.80	0.79
Generic	1.00	0.97	0.99
Normal	0.93	0.76	0.84
Reconnaissance	0.88	0.81	0.85
Shellcode	0.90	0.97	0.93
Worms	0.98	0.99	0.99
Accuracy	0.74		

The deep learning models faced challenges with the CIC-IDS-2017 dataset, especially in accurately identifying the attacks with minority classes. As shown in Table 9, the precision values notably fell under 50% for a few classes, highlighting the difficulty in detecting these minority attack types. Similarly, Table 9 shows the low values of recall and the F1 score of minor classes (bot, heartbleed, infiltration, attack brute force, attack XSS, and attack SQL injection) before the data augmentations. A similar situation with more complicated deep learning architectures before data augmentation is shown in Table 10, which presents the results of deep learning Architecture 5. However, when the data augmentation took place on this dataset, the performance of the different deep learning architectures improved. For instance, CNN Architecture 1 (as shown in Table 11) scored 85% in terms of accuracy, while the lowest precision—scored with 'Web Attack_XSS'—was 53%, and the lowest recall and F1-score values, which were scored by 'Web Attack Brute Force', were 15% and 25%, respectively. This means there is an overall improvement in precision, recall, and the F1 score of minor classes (the bold values in Table 11). Similar results occurred with other architectures, and their results were almost identical (as shown in Table 12 for the results of Architecture 5 after data augmentation). Therefore, we have obtained more improvements in recognizing intrusion attacks that existed in the available dataset.

Testing the various deep learning architectures of 2–5 (Tables 3–6) on UNSW-NB15 revealed that none of them outperformed the accuracy achieved by the simple CNN model (Table 2). This observation suggests that increasing the model's complexity does not necessarily lead to a better performance in intrusion detection tasks on this dataset. The more intricate models might be overfitting or not capturing the nuances of network intrusion data as effectively as the simpler model. However, the advanced architectures (Tables 3–6) performed better than the simple CNN model on the CIC-IDS-2017 dataset (as shown in Table 12). On the other hand, this performance comes at the cost of time, computation power, and space. A closer examination of class-specific performance (Table 11) showed significant variations. For instance, the precision was only 56% and 53% in identifying SQL injection and XSS attacks, while the identification of other attacks had over 80% precision. This disparity indicates the need for class-specific optimizations or tailored approaches for certain types of attacks.

The ROC curves (Figures 1 and 2) demonstrated high recall/sensitivity for almost all classes using deep learning architectures on both datasets (the average values of all architectures were taken), indicating the model's ability to correctly classify the true-positive rates across different attack types. However, the F1 scores (Figures 3 and 4) were high only for a few classes (analysis, backdoor, and DoS in the UNSW-NB15 dataset), suggesting a variance in precision and recall balance across different classes.

Table 9. CNN Architecture 1 classification performance using CIC-IDS-2017 dataset before data augmentation.

Classes	Precision	Recall	F1 Score
BENIGN	1.00	0.98	0.99
Bot	0.25	0.27	0.24
DDoS	1.00	0.99	0.99
DoS GoldenEye	0.98	0.98	0.98
DoS Hulk	0.94	1.00	0.97
DoS Slowhttptest	0.87	0.98	0.92
DoS slowloris	0.98	0.95	0.97
FTP-Patator	0.99	0.99	0.99
Heartbleed	0.15	0.13	0.16
Infiltration	0.19	0.21	0.21
PortScan	0.88	0.97	0.92
SSH-Patator	0.88	0.98	0.93
Attack_Brute Force	0.36	0.40	0.45
Attack_Sql Injection	0.24	0.28	0.32
Attack_XSS	0.56	0.59	0.60
Accuracy	0.98		

Table 10. Deep learning Architecture 5 classification performance using CIC-IDS-2017 dataset before data augmentation.

Classes	Precision	Recall	F1 Score
BENIGN	0.99	0.99	0.99
Bot	0.99	0.34	0.51
DDoS	1.00	0.99	0.99
DoS GoldenEye	0.98	0.98	0.98
DoS Hulk	0.97	0.98	0.97
DoS Slowhttptest	0.89	0.99	0.94
DoS slowloris	0.98	0.99	0.98
FTP-Patator	1.00	0.64	0.78
Heartbleed	0.15	0.17	0.15
Infiltration	0.13	0.12	0.15
PortScan	0.90	0.95	0.92
SSH-Patator	1.00	0.49	0.66
Attack_Brute Force	0.91	0.05	0.09
Attack_Sql Injection	0.11	0.13	0.10
Attack_XSS	0.09	0.08	0.12
Accuracy	0.98		

Table 11. CNN Architecture 1 classification performance using CIC-IDS-2017 dataset after data augmentation.

Classes	Precision	Recall	F1 Score
BENIGN	0.67	0.91	0.77
Bot	0.97	0.99	0.98
DDoS	0.99	1.00	1.00
DoS GoldenEye	0.84	0.99	0.91
DoS Hulk	0.98	1.00	0.99

Table 11. *Cont.*

Classes	Precision	Recall	F1 Score
DoS Slowhttptest	0.99	0.99	0.99
DoS slowloris	0.99	0.99	0.99
FTP-Patator	1.00	0.99	0.99
Heartbleed	1.00	0.80	0.89
Infiltration	0.99	0.79	0.88
PortScan	0.99	1.00	0.99
SSH-Patator	0.91	0.49	0.64
Attack_Brute Force	0.90	0.15	0.25
Attack_Sql Injection	0.56	0.72	0.63
Attack_XSS	0.53	0.97	0.69
Accuracy	0.85		

Table 12. CNN Architecture 5 classification performance using CIC-IDS-2017 dataset after data augmentation.

Classes	Precision	Recall	F1 Score
BENIGN	0.85	0.96	0.90
Bot	0.99	1.00	0.99
DDoS	1.00	1.00	1.00
DoS GoldenEye	0.80	0.99	0.89
DoS Hulk	0.99	1.00	1.00
DoS Slowhttptest	0.99	0.99	0.99
DoS slowloris	0.99	0.99	0.99
FTP-Patator	1.00	1.00	1.00
Heartbleed	1.00	1.00	1.00
Infiltration	1.00	1.00	0.99
PortScan	0.99	1.00	0.99
SSH-Patator	0.94	1.00	0.97
Attack_Brute Force	0.66	0.50	0.57
Attack_Sql Injection	0.99	0.59	0.74
Attack_XSS	0.63	0.75	0.68
Accuracy	0.92		

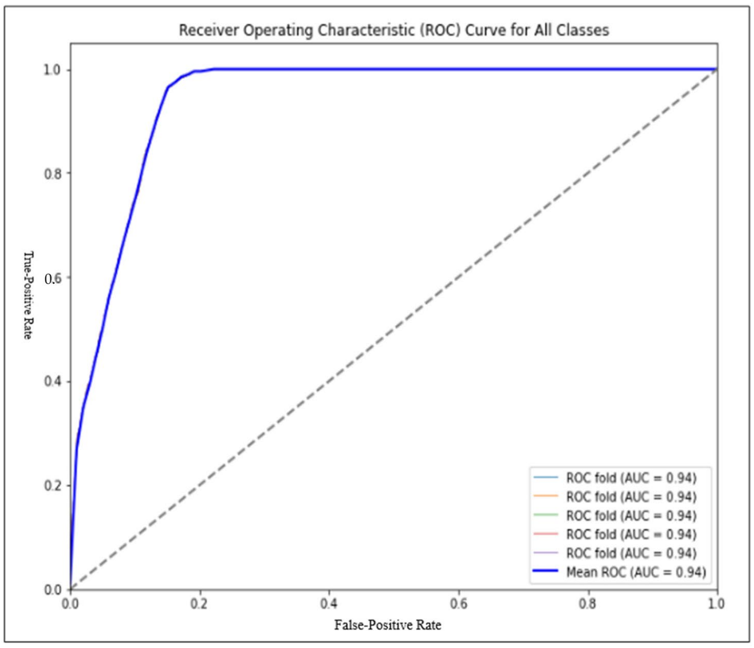

Figure 1. ROC with UNSW-NB15 dataset (after augmentation).

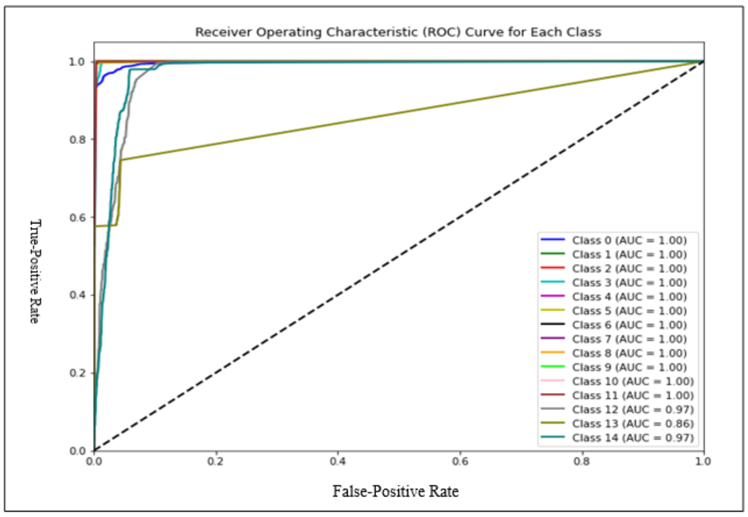

Figure 2. ROC with CIC-IDS-2017 (after augmentation).

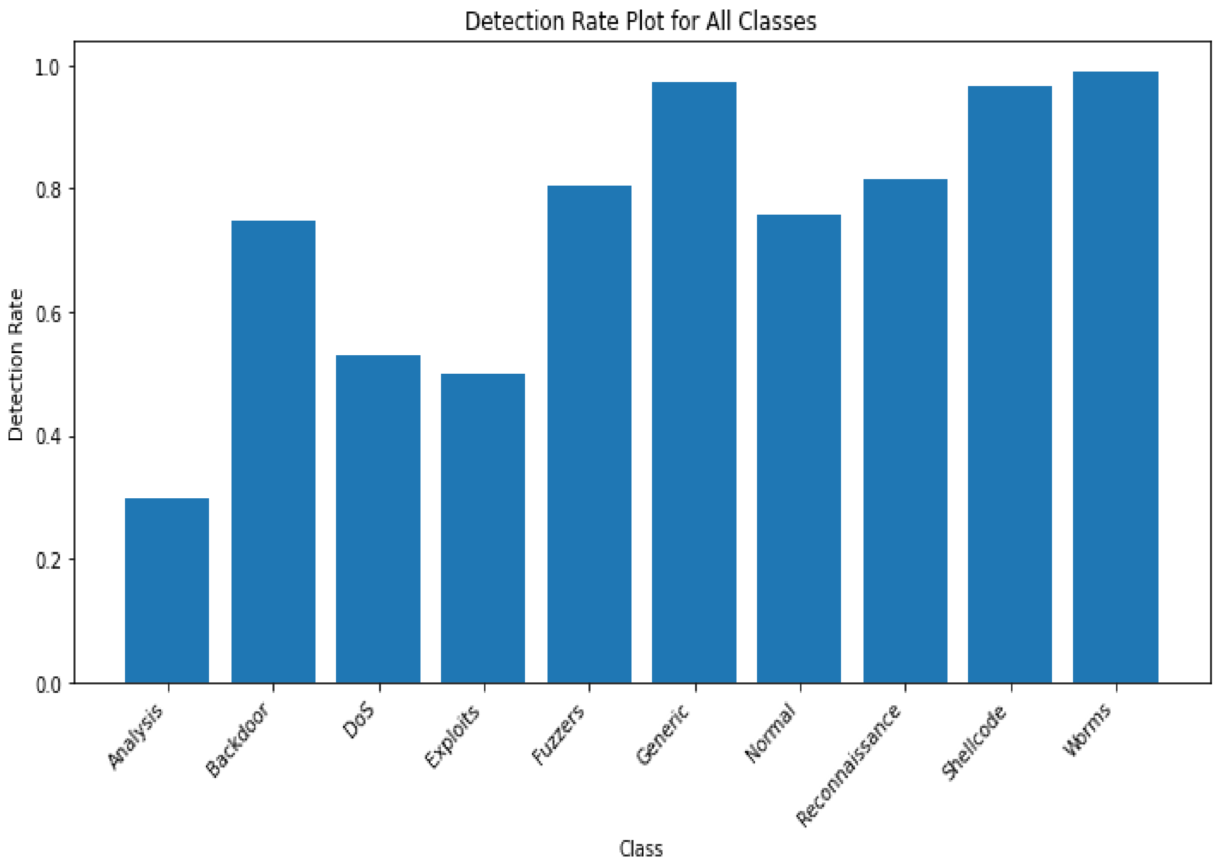

Figure 3. F1-score average for UNSW-NB-15 by all architectures (after augmentation).

The performance of the proposed deep learning architectures on the 5G-NIDD and FLNET2023 datasets are shown in Tables 13–16. Amarakoon et al. [25] reported high performances of the machine learning methods (KNN, random forest, and decision tree) on the 5G-NIDD dataset with all types of attacks. It was noticed that there are a reasonable number of entries for every class of attacks (except ICMPFLOOD, i.e., 366 entries), and this might be due to the fact that this dataset was generated experimentally in the lab. The above-mentioned architectures (1–5) were examined on the 5G-NIDD dataset (before and after augmentation) and showed similar results to those reported in [25]. As shown in Tables 13–16, it was found that the simple CNN Architecture 1 (Table 2) achieved high performances in both cases (before and after augmentation), which is similar to architectures 2–5 (Tables 3–6). This is due to the fact that both datasets (5G-NIDD and FLNET2023) have a good number of entries for minor classes of attack, which gives the deep learning methods sufficient data to identify patterns that feature those classes of attacks. Augmentation helps more in terms of improving the F1-score and recall values.

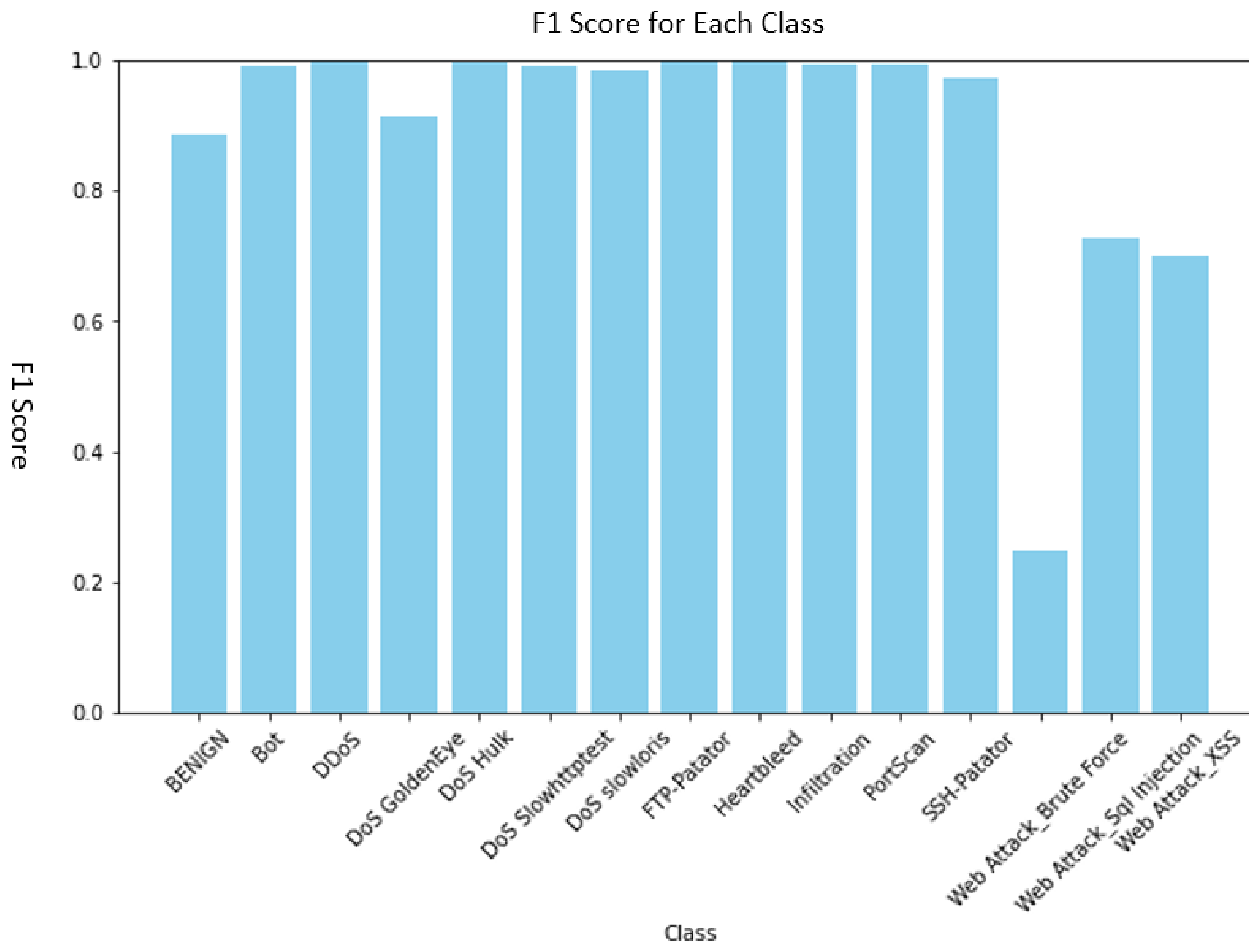

Figure 4. Average F1 score for CIC-IDS-2017 by all architectures (after augmentation).

Table 13. CNN Architecture 1 classification performance using 5G-NIDD dataset after data augmentation.

Classes	Precision	Recall	F1 Score
BENIGN	0.99	0.95	0.97
UDPFlood	0.95	0.99	0.97
HTTPFlood	0.99	1.00	0.99
SlowrateDoS	0.99	0.98	0.99
TCPConnectScan	1.00	0.99	1.00
SYNScan	1.00	1.00	1.00
UDPScan	1.00	1.00	1.00
SYNFlood	1.00	0.99	0.99
ICMPFlood	0.99	1.00	1.00
Accuracy	0.97		

Table 14. CNN Architecture 5 classification performance using 5G-NIDD dataset by architecture after data augmentation.

Classes	Precision	Recall	F1 Score
BENIGN	0.99	0.95	0.97
UDPFlood	0.95	0.99	0.97
HTTPFlood	0.99	1.00	1.00
SlowrateDoS	1.00	0.99	0.99
TCPConnectScan	1.00	1.00	1.00
SYNScan	1.00	1.00	1.00
UDPScan	1.00	1.00	1.00
SYNFlood	1.00	1.00	1.00
ICMPFlood	0.99	1.00	1.00
Accuracy	0.98		

Table 15. CNN Architecture 1 classification performance using FLNET2023 dataset after data augmentation.

Classes	Precision	Recall	F1 Score
DDoS-bot	1.00	0.99	0.99
DDoS-dyn	0.99	1.00	1.00
DDoS-stomp	1.00	1.00	1.00
DDos-tcp	1.00	1.00	1.00
DDoS-hulk	1.00	1.00	1.00
DoS-slowhttp	0.99	1.00	1.00
Infiltration-mitm	1.00	1.00	1.00
Normal	1.00	1.00	1.00
Web-command-injection	0.96	0.98	0.97
Web-sql-injection	1.00	0.89	0.94
Web-xss	0.99	1.00	0.99
Accuracy	1.00		

Table 16. CNN Architecture 5 classification performance using FLNET2023 dataset after data augmentation.

Classes	Precision	Recall	F1 Score
DDoS-bot	1.00	0.99	0.99
DDoS-dyn	0.99	1.00	1.00
DDoS-stomp	1.00	1.00	1.00
DDos-tcp	1.00	1.00	1.00
DDoS-hulk	1.00	1.00	1.00
DoS-slowhttp	0.99	1.00	1.00
Infiltration-mitm	1.00	1.00	1.00
Normal	1.00	1.00	1.00
Web-command-injection	0.97	0.99	0.98
Web-sql-injection	1.00	0.91	0.96
Web-xss	0.99	1.00	0.99
Accuracy	1.00		

5. Discussion

In contrast to the work of Sinha and Manollas [3], which incorporated bidirectional LSTM with CNN, this study tried to rely on CNN, mainly because it can perform better in terms of memory and CPU cycles. The results of [3] and this study are similar, but this work demonstrates that the combination of convolutional and fully connected layers, along with the learning rate scheduler, can provide an equally effective alternative for CNN+BiLSTM architecture. The reason for the preference for the combination of convolutional and fully connected layers is that it is generally less memory-intensive and computationally lighter compared to the LSTM layers. Therefore, for limited resources and a shorter time of training, this combination is preferable, particularly as it delivers results with the same accuracy as CNN+BiLSTM, at least with the UNSW-NB15 and CIC-IDS-2017 datasets or similar. This is in line with the conclusion arrived at by Vigneswaran et al. [4], although they tested their model with a small dataset (KDDCup-′99′).

On the other hand, Rodríguez et al. [2] reported high accuracy when machine learning methods were applied with intrusion detection; however, the dataset imbalance was not processed, and the extreme feature selection procedures (it considered 5 out of 77 features) to obtain high performance potentially underestimate important features. This study employed data augmentation to balance the dataset and assess the performances of deep learning methods in prediction intrusions. Moreover, all features were included except those related to IDs or those that were constants. As discussed in the literature, this enables the detection of intrusion that could be characterized by small changes in a very limited number of features, therefore including all features to ensure considering all possibilities. The conclusion drawn is that while more advanced models may offer slightly higher accuracy, a simple or shallow CNN-based model also remains capable of achieving high detection accuracy. This conclusion is drawn from the findings of this study and the research conducted by Ayantayo et al. [17], which indicated that, despite exploring numerous architectures and techniques to maintain feature quality, model accuracy did not surpass 85%. Hence, it is likely that accuracy is significantly influenced by the quality of the dataset rather than by being solely reliant on the depth of the deep learning architectures. This conclusion came from the findings of this study, where high-quality datasets (FLNET2023 and 5G-NIDD) were assessed, and the results of the simple deep learning model (in this study) and machine learning methods by Amarakoon et al. [25] delivered high accuracy with data augmentation.

Overall, the fact that several networks were assessed with four datasets and have delivered the same level of accuracy suggests that the datasets might have certain characteristics that make them conducive to deep learning models and that these models are effectively capturing the patterns within the data. This can be interpreted in several ways as follows: (1) Complexity of data, where deep learning models, particularly neural networks with multiple layers, excel at capturing intricate and hierarchical patterns within data. When different deep learning architectures yield similar accuracies, it indicates that the dataset may possess complex and non-linear relationships that deep models are adept at uncovering [29]. (2) High-dimensional data, where deep learning models are well-suited for highly dimensional data, such as images, audio, or text. The CIC-IDS-2017, 5G-NIDD, and FLNET2023 datasets contain a large number of features; accordingly, deep learning methods may have an advantage in extracting meaningful representations, leading to comparable accuracies across different architectures [30]. (3) Rich feature representations, where deep learning models can automatically learn rich feature representations from raw data. When multiple deep learning methods perform similarly, it suggests that the dataset contains sufficient information for these models to derive meaningful features without extensive manual feature engineering [31]. (4) Data variability, where if the dataset is diverse and exhibits variability in its patterns, deep learning models can adapt and generalize effectively. The fact that different architectures yield similar accuracies indicates that these models are robust to variations within the dataset [32]. (5) Data size, where deep learning models often involve large amounts of data to accomplish their best. If multiple deep learning methods achieve the same accuracy, it implies that the dataset may be sufficiently large enough to support the training of deep models without overfitting [33], which is the case with this work. A small number of entries of attack classes could cause researchers to consider a further preprocessing step, which is ensuring all attack classes are represented in the training and testing portions of the datasets, which makes augmentation a good choice to overcome the small number of entries of a specific group of attacks.

In conclusion, the provision of an increased volume of data, both in terms of features and entries, typically leads to a higher accuracy in outcomes, even when employing relatively simple deep learning architectures. However, this expectation does not consistently align with the reality of many publicly available datasets. Such datasets often face criticism on several fronts, including their representation of real intrusion attacks, as well as concerns regarding their quality and size. Conversely, datasets such as FLNET2023 and 5G-NIDD, which are generated in simulated environments, tend to yield typical results with both machine learning and deep learning techniques. Nevertheless, this raises critical questions regarding the extent to which the data from these datasets accurately mirror real-world intrusion scenarios and the feasibility of having all the represented features (columns) available during actual intrusion incidents.

6. Conclusions

The datasets UNSW-NB15 and CIC-IDS-2017 were selected for analysis due to their prevalent usage in recent scholarly publications. However, it is vital to note that both datasets exhibit a disproportionality, featuring a smaller number of attack entries in comparison to normal network flow entries. This characteristic was also observed in the FLNET2023 and 5G-NIDD datasets. Despite this, their content has been substantially enhanced relative to other datasets in the field of intrusion detection. To ensure a comprehensive assessment of the performance of several deep learning methods, five distinct architectures were evaluated using augmented versions of all datasets. The empirical findings suggest that for effective intrusion detection, complex and resource-intensive architectures may not be necessary. Instead, simple CNN-based architectures with data augmentation are capable of achieving high accuracy. This observation was further substantiated when these architectures were applied to the FLNET2023 and 5G-NIDD datasets. In summary, when multiple deep learning methods exhibit comparable levels of accuracy on a dataset, it is indicative of the dataset's complexity, high dimensionality, and richly informative nature, which align well with the strengths of deep learning architectures. This also implies that deep learning models are proficient in extracting and utilizing the relevant features for predictive purposes, demonstrating their versatility and ability to manage diverse and challenging datasets. Future research should explore the efficacy of other deep learning architectures across a broader spectrum of datasets, with the aim of advancing the capabilities of intrusion detection systems.

Author Contributions: Conceptualization, R.M. and F.S.; methodology, R.M. and F.S.; software, R.M.; validation, A.A.A. (Abdulwahab Ali Almazroi), F.S.A. and A.A.A. (Abdulaleem Ali Almazroi); formal analysis, R.M. and F.S.; investigation, A.A.A. (Abdulwahab Ali Almazroi), F.S.A. and A.A.A. (Abdulaleem Ali Almazroi); resources, A.A.A. (Abdulwahab Ali Almazroi), F.S.A. and A.A.A. (Abdulaleem Ali Almazroi); data curation, R.M.; writing—original draft preparation, R.M. and F.S.; writing—review and editing, R.M., F.S., A.A.A. (Abdulwahab Ali Almazroi), F.S.A. and A.A.A. (Abdulaleem Ali Almazroi); visualization, R.M.; project administration, F.S. and F.S.A.; funding acquisition, A.A.A. (Abdulwahab Ali Almazroi), F.S.A. and A.A.A. (Abdulaleem Ali Almazroi). All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the University of Jeddah, Jeddah, Saudi Arabia, under grant No. UJ-23-DR-212. Therefore, the authors thank the University of Jeddah for its technical and financial support.

Data Availability Statement: The datasets are available online upon request.

Acknowledgments: This work was funded by the University of Jeddah, Jeddah, Saudi Arabia, under grant No. UJ-23-DR-212. Therefore, the authors thank the University of Jeddah for its technical and financial support.

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

References

1. Ieracitano, C.; Adeel, A.; Gogate, M.; Dashtipour, K.; Morabito, C.F.; Larijani, H.; Raza, A.; Hussain, A. Statistical analysis driven optimized deep learning system for intrusion detection. In *Advances in Brain Inspired Cognitive Systems*; Springer Nature: Cham, Switzerland, 2018; Volume 10989, pp. 759–769. [CrossRef]

2. Rodríguez, M.; Alesanco, Á.; Mehavilla, L.; García, J. Evaluation of Machine Learning Techniques for Traffic Flow-Based Intrusion Detection. *Sensors* **2022**, *22*, 9326. [CrossRef] [PubMed]
3. Sinha, J.; Manollas, M. Efficient Deep CNN-BiLSTM Model for Network Intrusion Detection the 2020. In Proceedings of the 3rd International Conference on Artificial Intelligence and Pattern Recognition, Xiamen, China, 26–28 June 2020.
4. Markevych, M.; Dawson, M. A review of enhancing intrusion detection systems for cybersecurity using artificial intelligence (ai). In Proceedings of the International Conference Knowledge-Based Organization, Sibiu, Romania, 19 July 2023; Volume 29, pp. 30–37.
5. Dini, P.; Elhanashi, A.; Begni, A.; Saponara, S.; Zheng, Q.; Gasmi, K. Overview on Intrusion Detection Systems Design Exploiting Machine Learning for Networking Cybersecurity. *Appl. Sci.* **2023**, *13*, 7507. [CrossRef]
6. Vigneswaran, R.K.; Vinayakumar, R.; Soman, K.P.; Poornachandran, P. Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security. In Proceedings of the 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 10–12 July 2018.
7. Nour, M.; Slay, J. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset. *Inf. Secur. J. A Glob. Perspect.* **2016**, *25*, 18–31.
8. Nour, M.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 10–12 November 2015.
9. Su, T.; Huazhi, S.; Zhu, J.; Want, S.; Li, Y. BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset. *IEEE Access* **2020**, *8*, 29575–29585. [CrossRef]
10. Vinayakumar, R.; Soman, K.; Poornachandran, P. Evaluation of recurrent neural network and its variants for intrusion detection system (IDS). *Int. J. Inf. Syst. Model. Des. Clin. Res.* **2017**, *8*, 43–63. [CrossRef]
11. Balakrishnan, V.; Shi, Z.; Law, C.; Lim, R.; Teh, L.; Fan, Y. A deep learning approach in predicting products' sentiment ratings: A comparative analysis. *J. Supercomput.* **2022**, *78*, 7206–7226. [CrossRef] [PubMed]
12. Zheng, B.; Liu, B. A Scalable Purchase Intention Prediction System Using Extreme Gradient Boosting Machines with Browsing Content Entropy. In Proceedings of the 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 12–14 January 2018.
13. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
14. Khan, R.U.; Zhang, X.; Alazab, M.; Kumar, R. An improved convolutional neural network model for intrusion detection in networks. In Proceedings of the Cybersecurity and Cyberforensics Conference (CCC), Melbourne, VIC, Australia, 8–9 May 2019.
15. Wang, W.; Sheng, Y.; Wang, J.; Zeng, X.; Ye, X.; Huang, Y.; Zhu, M. HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access* **2018**, *6*, 1792–1806. [CrossRef]
16. Yang, S. Research on network behavior anomaly analysis based on bidirectional LSTM IEEE 3rd Information Technology, Networking. In Proceedings of the Electronic and Automation Control Conference (ITNEC), Chengdu, China, 15–17 March 2019.
17. Ayantayo, A.; Kaur, A.; Kour, A.; Schmoor, X.; Shah, F.; Vickers, I.; Kearney, P.; Abdelsamea, M.M. Network intrusion detection using feature fusion with deep learning. *J. Big Data* **2023**, *10*, 167. [CrossRef]
18. Pratap, P.C.; Pandey, P. Time Series Forecasting Using Machine Learning Models: A Survey. In Proceedings of the 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 10–12 July 2018.
19. Zhao, Z.; Chen, H.R.; Liu, R. Deep Learning in Time-Series Analysis: A Survey. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
20. Hajirahimi, Z.; Khashei, M. Hybrid structures in time series modeling and forecasting: A review. *Eng. Appl. Artif. Intell.* **2019**, *86*, 83–106. [CrossRef]
21. Bachl, M.; Hartl, A.; Fabini, J.; Zseby, T. Walling up Backdoors in Intrusion Detection Systems. In Proceedings of the the 3rd ACM CoNEXT Workshop on Big Data, Machine Learning and Artificial Intelligence for Data Communication Networks, Orlando, FL, USA, 9 December 2019.
22. Sharma, N.; Yadav, N.; Singh; Sharma, S. Classification of UNSW-NB15 dataset using Exploratory Data Analysis using Ensemble Learning. *Trans. Ind. Netw. Intell. Syst.* **2021**, *8*, e4. [CrossRef]
23. Sharafaldin, I.; Lashkari, A.; Habibi; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic. In Proceedings of the Characterization the 4th International Conference on Information Systems Security and Privacy, Funchal, Portugal, 22–24 January 2018.
24. Duan, L.; Xue, W.; Huang, J.; Zheng, X. Joint Sample Position Based Noise Filtering and Mean Shift Clustering for Imbalanced Classification Learning. *Tsinghua Sci. Technol.* **2024**, *29*, 216–231. [CrossRef]
25. Samarakoon, S.; Siriwardhana, Y.; Porambage, P.; Liyanage, M.; Chang, S.; Kim, J.; Kim, J.; Ylianttila, M. 5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network. *IEEE Dataport* **2022**. [CrossRef]
26. Kumar, P.; Liu, J.; Tayeen, A.S.M.; Misra, S.; Cao, H.; Harikumar, J.; Perez, O. FLNET2023: Realistic Network Intrusion Detection Dataset for Federated Learning. In Proceedings of the MILCOM 2023–2023 IEEE Military Communications Conference (MILCOM), Boston, MA, USA, 30 October–3 November 2023; pp. 345–350.
27. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]

28. Elreedy, D.; Atiya, A.F. A comprehensive analysis of synthetic minority oversampling technique (SMOTE) for handling class imbalance. *Inf. Sci.* **2019**, *505*, 32–64. [CrossRef]
29. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1.
30. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
31. Bengio, Y. Learning deep architectures for AI. *Found. Trends Mach. Learn.* **2009**, *2*, 1–127. [CrossRef]
32. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
33. Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv* **2017**, arXiv:1611.03530. [CrossRef]