electronics

MDPI

Article

Internet of Drones Intrusion Detection Using Deep Learning

Rabie A. Ramadan [1,2,*], Abdel-Hamid Emara [3,4], Mohammed Al-Sarem [4,5] and Mohamed Elhamahmy [6]

[1] Computer Engineering Departmental, Faculty of Engineering, Cairo University, Giza 12613, Egypt
[2] College of Computer Science and Engineering, University of Hail, Hail 55473, Saudi Arabia
[3] Department of Computers and Systems Engineering, Faculty of Engineering, Al-Azhar University, Cairo 11884, Egypt; abdemara@gmail.com
[4] College of Computer Science and Engineering, Taibah University, Medina 41477, Saudi Arabia; mohsarem@gmail.com
[5] Department of Computer Science, Saba'a Region University, Mareb, Yemen
[6] Higher Institute of Computer Science and Information Systems, Fifth Settlement, Cairo 11477, Egypt; mezzat1967@yahoo.com
* Correspondence: rabie@rabieramadan.org

Abstract: Flying Ad Hoc Network (FANET) or drones' technologies have gained much attraction in the last few years due to their critical applications. Therefore, various studies have been conducted on facilitating FANET applications in different fields. In fact, civil airspaces have gradually adopted FANET technology in their systems. However, FANET's special roles made it complex to support emerging security threats, especially intrusion detection. This paper is a step forward towards the advances in FANET intrusion detection techniques. It investigates FANET intrusion detection threats by introducing a real-time data analytics framework based on deep learning. The framework consists of Recurrent Neural Networks (RNN) as a base. It also involves collecting data from the network and analyzing it using big data analytics for anomaly detection. The data collection is performed through an agent working inside each FANET. The agent is assumed to log the FANET real-time information. In addition, it involves a stream processing module that collects the drones' communication information, including intrusion detection-related information. This information is fed into two RNN modules for data analysis, trained for this purpose. One of the RNN modules resides inside the FANET itself, and the second module resides at the base station. An extensive set of experiments were conducted based on various datasets to examine the efficiency of the proposed framework. The results showed that the proposed framework is superior to other recent approaches.

Keywords: intrusion detection; FANET; RNN; LSTM; deep learning

Citation: Ramadan, R.A.; Emara, A.-H.; Al-Sarem, M.; Elhamahmy, M. Internet of Drones Intrusion Detection Using Deep Learning. *Electronics* **2021**, *10*, 2633. https://doi.org/10.3390/electronics10212633

Academic Editors: Hung-Yu Chien and Khaled Elleithy

Received: 11 August 2021
Accepted: 12 October 2021
Published: 28 October 2021

1. Introduction

Today, the Internet has become an important and correlative part of our life with a progressive increase in the number of devices connected to it, especially Internet of Things (IoT) devices. Internet of Things (IoT) is one of the most recent technologies that connect devices through the Internet resulting in progressive development and supporting human lives, professions, and education [1,2]. IoT can deal with all types of connected devices in daily life, also called the Internet of Everything (IoE) [3]. At the beginning of 2021, the IoT, one of the quickest developing online areas, was connected with 50 billion gadgets [4]. However, as Yuan et al. reported in [5], 17 million denial-of-service attacks on networks occurred in 2020.

Internet of Drones (IoD) is constructed from IoT by exchanging "Things" with "Drones" while holding incomparable properties. IoD, which is defined by Gharibi et al. [6] as a "layered network control architecture," plays an essential role in the development of Unmanned Aerial Vehicles (UAVs). In IoD, several drones can join and form a network while sending and receiving data from each other. This paper uses the terms FANET, drones, and UAV interchangeably to refer to the same meaning.

Recently, network security has become a critical research domain, particularly after developing Internet and communication techniques. It uses different tools such as firewall and Network Intrusion Detection Systems (NIDSs) for network and assets security [7]. NIDSs are used to permanently monitor the network traffic for bad and suspicious behavior [8–10]. The first idea of IDS was proposed in 1980, followed by several IDS products developed to fulfill network security requirements [11]. However, over the last decade, enormous developments in communication and network technologies have occurred, increasing the network size, the number of applications, and the amount of data generated and shared across the network. Subsequently, the number of new attacks has increased, and their detection has become a challenging task. For example, the data at a certain node are critical for an organization, where any exposure to that node may negatively seriously affect the organization.

In addition, the US Department of Transportation published a study in 2001 evaluating the vulnerability of the US transportation infrastructure to GPS disruption drones [12]. The study emphasized the dangers presented by civil GPS spoofing attacks. Another drone attacks analysis is reported in [13], where many of the drones' security challenges are described while the authors focus on WiFi attacks and GPS spoofing while drones Denial of Service (DoS) attacks and other threats are reported in [14,15].

IDSs could be classified as anomaly-based, signature-based, or specification-based techniques. Anomaly-based techniques could be further classified as statistical-based, knowledge-based, or machine learning-based. The statistical-based anomaly IDS compares traffic statistics to a generated stochastic model of normal operation. The attack is the deviation between the two statistical patterns: the normal memorized and the current captured. In knowledge-based anomaly detection, experts offer many rules in the form of an expert system or a fuzzy-based system to describe the behavior of normal connections and attacks. The rule-based system is connected to inputs in fuzzy-based anomaly detection. Based on the input data, a subset of the rules is activated. In a machine learning-based anomaly IDS, an explicit or implicit model of the observed patterns is created. Based on previous results, these models are updated on a regular basis to improve intrusion detection efficiency.

Signature-based detection is also known as knowledge-based detection. It compares the current traffic to the attack signature and reports an attack if the two match; otherwise, it does not report an attack. Unlike previous methods, this one does not generate a large number of false alarms. However, it requires constant signature updating. Moreover, a specification-based method employs specifications or constraints to define the operation of a particular program and alerts the user to any violation of such specifications or constraints based on comparison to previously established and memorized specifications and constraints.

The focus of this paper falls in the first category of IDSs, anomaly based IDSs, more specifically on machine learning approaches. Machine learning algorithms showed a better performance than nonmachine learning approaches due to learning and training modules [16]. Thus, a cost effective as well as efficient IDS is highly mandatory to keep the drones' network secure. Many of the researchers suggested artificial intelligence (AI) techniques, including machine learning (ML), deep learning (DL), and other techniques. These techniques aim at learning valuable information from a huge amount of data [17]. Both ML and DL techniques have become the most popular techniques in network security because of their ability to learn useful characteristics from the network traffic and detect normal and abnormal activities that rely on the learning styles [18]. ML techniques learn useful information from the network traffic depending mainly on feature engineering [19]. At the same time, DL techniques do not depend on feature engineering and are perfect at automatically learning complex features from pure data due to their deep framework [20].

Over the past years, researchers have suggested different ML and DL bases for efficient NIDS to detect malicious attacks. However, the progressive increase in network traffic and its security threats has demonstrated several challenges for the NIDS techniques

to detect attacks efficiently. The IDSs aim at detecting intruders. In an IoT field, these intruders act as hosts attempting to access other nodes without a license. A NIDS consists of three basic features: an agent, an analysis engine, and a response module. The main function of the agent is to collect information from the network by monitoring events. In contrast, the analysis engine and a response module are responsible for tracing the signs of intrusion, generating alerts, and working on the results received from the analysis engine, respectively. NIDSs have become more useful and efficient year after year, but invaders have also developed various attack methods to overcome these detection technologies. In addition, traditional NIDSs are not suitable for the complex network layers in IoT [21]. The complexity increases when UAV is involved where drone operations are more complex and require more security than other networks due to their critical applications. In fact, successful drone attacks may lead to exposing people's life to danger. Moreover, the nature of the drones' networks requires different security arrangements, especially intrusion detection.

This study aims to provide an innovative distributed framework for drone intrusion detection based on DL techniques, namely, LSTM-RNN architecture. The main idea is to have distributed modules of RNN where each drone will have one module that tries to detect any attack on the drone itself. Another centralized LSTM-RNN module resides at the base station that confirms the attack and makes the decision, notifying other drones of certain attacks. The drones' LSTM-RNN module handles only the traffic coming from its communication device. However, the LSTM-RNN base station module examines all of the drones' traffic.

The contributions of this current work are:

- A real-time data analytics framework for investigating FANET intrusion detection threads has been proposed.
- Utilizing deep learning algorithms for drone networks intrusion detection.
- An extensive set of experiments are conducted to examine the efficiency of the proposed framework.
- Examining the proposed framework on various datasets.

The paper is organized as follows: Section 2 is a review of the most related work while Section 3 is a description of the proposed framework; the conducted experiments are detailed in Section 4 along with the used evaluation metrics; the simulation results are deliberated and evaluated in Section 5. Finally, the paper concludes and future work is provided in Section 6.

2. Related Work

2.1. Intrusion Detection System

NIDS is the most reliable technique to detect internal and external attacks with high accuracy [22–24]. NIDSs can be categorized into signature-based NIDSs and anomaly-based NIDSs [25]. Recently, several approaches have been proposed, among which the machine learning-based ones are the promising ones [22,26,27]. However, most NIDSs produce a large number of false positives and low detection accuracy. Thus, researchers investigated the ability of hybridization of several machine learning techniques and statistical methods. The following section presents a brief review of machine learning-based techniques that are used for intrusion detection.

2.2. Machine Learning Methods

Several researchers have used machine learning to discover complex patterns and behaviors from collected data. Unlike rule-based NIDSs [28,29] and expert system-based NIDSs [30], machine learning-based NIDSs can extract distinct features from new arriving messages and intrusions. One of the machine learning algorithms proposed in the literature is random forest (RF) [31], where automated intrusion patterns are extracted. Intruders are assessed by comparing network activity to certain patterns. The authors based their conclusions on the KDD'99 dataset assessment evaluating the proposed model performance.

They used 10% as a training set to train the RF classifier and 90% as a test set. For unbalanced data, both down-sampling and oversampling were applied. The experiment was then run in WEKA with 66% training samples and 34% testing samples. The suggested method obtained a 94.7% detection rate with a 2% false-positive rate.

According to Zhang et al. [26], RF classifier can detect intrusions in high-speed traffic data. Therefore, they developed RF on top of Apache Spark's distributed processing system. This framework can identify real-time network intrusions at high volume and speed. According to Al-Jarrah et al. [32], feature selection affects Random Forest performance. The authors used RF with forward and backward features selection methods for the same purpose. They utilized the original KDD'99 dataset after cleaning out redundancy. Some of the processing operations were done using normalization, discretization, and balancing. For the KDD'99 dataset, the most important features in the KDD'99 dataset were identified using FSR and BER with RF classifier method from among the 41 features. The results indicate that the proposed RF-FSR technique is appropriate for large-scale NIDSs. Another machine learning technique was suggested in [33], where the authors utilize evolutionary algorithms as an intrusion detection method. Instead of the KDD'99 standard for evaluating the model, they utilized a hand-made dataset that involved 35 packet-based features. There was a balance (600 common cases with 600 attacks) in the training dataset, whereas the test set for the two-class label included 100 instances. The first 30 populations were randomly generated. After the model training, all 35 features were fed into the KNN algorithm. The results showed that 97.42% of the most common attacks were verified for only the top 19 features and 78% for the top 28 features. In addition to RF and KNN, intrusion of the malicious network was examined using SVM [34], incremental SVM [35], Decision Trees [36], and Naïve Bayes.

The most relevant work to our proposal in this paper is the algorithm proposed in [1], where the authors utilized two deep learning algorithms. However, the paper is designed for regular networks, not for drone networks. In addition, the contribution in [1] was very limited and did not show the implementation details of the used deep learning algorithms. Moreover, the authors examined their proposal based on only one dataset, CICIDS2017, focusing on DDoS attacks. Another work was proposed in [37] where the authors tried to develop a mathematical model with optimal hyperparameters tuned for high-performance sparse autoencoders to optimize features and classify normal and abnormal traffic patterns. Their approach is different from the proposed approach in this paper, where again, the proposed approach in [37] is mainly designed for regular networks, not for drone networks. In addition, the paper experiments focused on the CICIDS2017 dataset, where some other attacks that could be found in other datasets were not examined. The third related study in the field of NIDSs was proposed by Razan et al. [38], where the main focus of the paper was dimensionality reduction to enhance intrusion detection using machine learning approaches as classifiers. Thus, its focus is different from the one proposed in this paper, where here, the focus is on drone networks. Our main proposal in this paper involves a framework with two different components, one is embedded in the drone and another is at the base station. Both have a deep learning module for local intrusion detection, and they also help each other confirm the intrusion and notify other drones for further communication with the intruder drone. In addition, the proposed framework is examined with different datasets to cover most of the possible attacks and use various algorithms such as LSTM, RNN, and LR. For more details on the data mining methods used in NIDSs, the reader is directed to [39,40]. The summarization of this section is presented in Table 1.

Table 1. Feature description for phishing websites.

Paper	Findings	Settings/ Environment	Feature Set	Technique Applied	Dataset
[31]	Detection rate 94.7%, False positive rate 2%	Down-sampling/ Oversampling 66% training set/ 34% testing set WEKA	41 features from the TCP dump data	Random Forest	KDD'99
[26]	Execution time of the proposed time is less than other classifiers, GBDT achieved (98.2 Pre. 97.4 Recall, 97.8 F1)	70% training set/30% testing set Spark2.2.0, kafka2.11	13 features	Distributed Random Forest, Gradient boosting decision tree (GBDT), multiclass SVM and Adaboost	CICIDS2017
[32]	RF-FSR: Accuracy 99.9%	Normalization/ Discretization/ Downsamplig Balancing, 10-fold cross validation WEKA	41 features as the original KDD'99	RF-FSR/RF-BER	Filtered version of KDD'99
[33]	Among of 35 features, 28 features yielded 78% accuracy for unknown attacks, 19 features yielded 97.42% accuracy for known attacks	Normalization/30 chromo-somes/Microsoft Visual C++	35 features which are derived from packet headers	K-nearest Neigh-bour/Genetic Algorithm	Handmade dataset
[34]	Detection rate: 88.5%, False alarm ratio: 11.5%; Best achieved accuracy 96%	LIBSVM (MATLAB)	-	C4.5/one-class SVM	NSL-KDD
[35]	Detection rate: (91.83–97.731), False alarm ratio: (0.0375–0.188)	Resampling/ Normalization Visual C++ and MATLAB	All features of the KDD'99 dataset	Improved incremental SVM (RS-ISVM)	KDD'99
[36]	Decisions trees are slightly better than Naïve Bayes, Accuracy of DT:93.02%/Accuracy of NB:91.45%	Dataset granularities	All features of the KDD'99 dataset	Naïve Bayes/Decision Trees	KDD'99

2.3. Hybridization of IDSs

Below is a short overview of several noteworthy studies integrating various classification systems for intrusion detection, as given in Table 2. These systems used both machine learning and statistical models along with the two types of intrusion detection (signature-based IDSs and anomaly-based IDSs). For instance, Thaseen and Kumar [41] selected Chi-square features to reduce the data dimension and find the optimal subset of all data attributes. The selected attributes were then used for multiclass SVM training. In addition, the authors of [42] introduced a two-stage hybrid IDS method integrating a SIDS with AIDS models in a hierarchical fashion. The C4.5 decision tree classifier was built on top of a SIDS. SVM was then used to split the subsets. Similarly, the authors of [43] proposed another hybrid method for detecting known and new attacks based on SVM. They tried to produce small and representative training datasets using a modified k-means method. The proposed approach reduces overall training time and improves the NIDS

performance. Moreover, in [44], the k-means and decision trees model was utilized to detect abnormal and normal computer activity.

Table 2. Detection model: hybrid machine learning approach.

Paper	Machine Learning Classifier		Intrusion Detection System Techniques			Technique Applied
	Ensemble	Hybrid	Hybrid IDS	AIDS	SIDS	
[41]	x	√	x	x	√	Chi-square /multiclass SVM
[42]	x	√	√	x	x	C4.5/One-class SVM
[44]	x	√	√	√	x	K-means/C4.5
[25]	√	√	√	√	√	Stacking Ensemble of C5.0/One-class SVM
[31]	√	x	x	√	√	Random Forest
[45]	x	√	x	x	√	Random Forest /Average One-Dependence Estimator
[32]	√	x	x	√	x	Random Forest with Feature Selection

Furthermore, the authors of [25] proposed a hybrid IDS combining C5.0 and one-class SVM. The model was evaluated using NSL-KDD and ADFA datasets. The findings revealed higher detection improvement and lower false alarm rates. With regard to basic classifiers (Naïve Bayes, decision trees (C4.5), decision trees, and KNN approach), it is shown that the stacking group technique alone was able to reduce false alarm rates. However, it requires a long execution time. In [45], the authors combined the Average One-Dependence Estimator with a Random Forest ensemble classifier (AODE). The AODE addressed the problem of independence of the attribute due to the naive Bayes deficiency.

3. The Proposed Drone Intrusion Detection Framework

The nature of the UAV network is different from other traditional networks since they have to communicate with the ground base station(s). Moreover, drones move from one place to another, depending on the application. That is why the proposed framework consists of two components, the drone and the base station. In addition, there is no need for different RNN modules where a centralized module is more than enough in traditional networks. On the other hand, the drones' networks might need more than one RNN module, one on the drone itself and another at the base station. The base station module handles all drone traffic and confirms the drone's RNN module's decision.

This section introduces the proposed distributed framework for drone intrusion detection. This paper uses the LSTM-RNN version of RNN. The main idea is to have distributed modules of LSTM-RNN where each drone will have one module that tries to detect any attack on the drone itself. Another centralized module, known as the base station, resides on the ground with another LSTM-RNN, to confirm the attack and makes the decision notifying other drones. The base station is assumed to be able to communicate with all of the drones in the sky. It could be a single station or distributed stations. In this paper, we assumed that the base station is only one station, and it is able to communicate directly with all the drones.

Drone intrusion detection could work in two different ways, batch or stream, depending on the used technology. For instance, if MapReduce is used as a centralized component for the decision-making process, the processing has to be run in batches, which takes some time in order to form the batch. On the other hand, if other frameworks, such as Storm, Spark, Flink, or Apache Kafka, are utilized, the detection process could be done at the runtime. Apache Kafka is preferred in this paper due to its distribution characteristics, especially when the big data stream is coming on. The paper emulates the real-time analysis

by feeding the data into a form of a stream to the RNN modules. The framework, shown in Figure 1, has two main components, drones and base station.

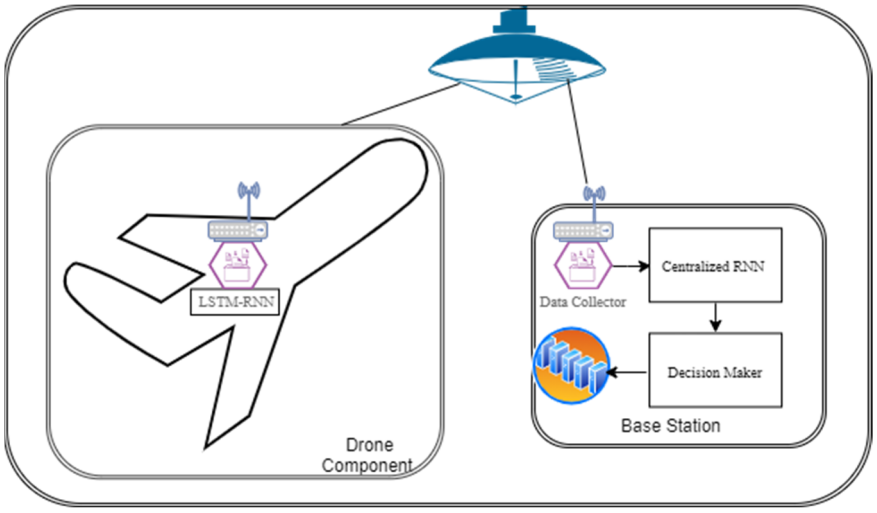


Figure 1. Drone intrusion detection framework.

As shown in the figure, the drone is assumed to have a simplified version of LSTM-RNN. In this context, the simplified version means a network with less traffic in terms of training and runtime traffic. In other words, the proposed framework is a generic framework that is designed to work on physical drones as we plan to do in our future work. Therefore, it is assumed that each drone will have light communication and less traffic than the base station. Thus, the drone LSTM-RNN module does not require the same amount of data analysis as the base station LSTM-RNN module, where the base station LSTM-RNN module is supposed to receive all of the drone traffic for analysis.

3.1. Drone RNN

Signature-based IDSs depend on comprehensive knowledge regarding previously detected attacks. Regular and statistical methods do not operate well in a drone system, which is vulnerable to new attacks. Unsupervised learning strategies are advanced techniques used to identify attacks based on the device parameters and produce alerts about abnormal attacks. This way, the device would be able to identify anomalies and will take measures to deter attacks. Alarms would be generated in the device if the protection system cannot avoid the attack, alerting the system administrator. This is considered the main difference between signature-based and learning-based intrusion detection systems. However, in most cases, the attack will not be detected if there is no prior knowledge about it. Moreover, the data noise may affect the detection process. Therefore, there is a need for real-time algorithms that handle the data as a series of events, not as separate events.

The advances in deep neural networks have made the supervised and unsupervised tools more effective. One of the deep learning algorithms is Recurrent Neural Networks (RNN); RNN comprises nodes linked to each other. These nodes may store information sequentially, but the data elements are processed one at a time, and they can handle input and output separately. RNNs are used effectively for different applications such as speech synthesis, time series prediction, video processing, and natural language processing. RNN uses a multi-layer perceptron design, as shown in Figure 2. It also has a looping structure that serves as the main path to enable knowledge transfer from one stage to the next. Figure 3 shows the folded RNN layers where the RNN loops are extracted.

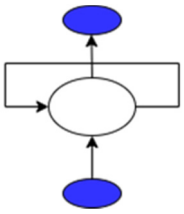


Figure 2. Unfolded RNN.

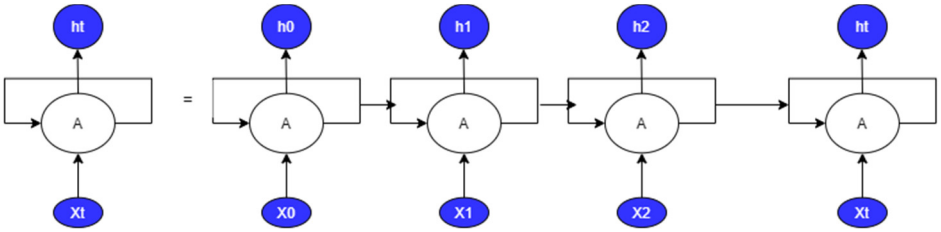


Figure 3. Folded RNN.

Due to many reasons, RNN-LSTM is used in this paper for drone intrusion detection. The acronym LSTM means "Long Short-Term Memory". In RNN-LSTM, the gradient may approach zero when multiplying values between zero and one or exceeding one (in the case of multiplying large values). This implies that the network might not connect previous inputs to their outputs. Looking at the RNN-LSTM structure, four layers are considered as the main components of the network. Those layers interact or communicate with one another. Looking at Figure 4, when the input is received, it passes through the four layers one after another. The first layer is the sigmoid layer, which determines whether to retain or discard the output from the previous layer. One more sigmoid layer determines which values to be updated. A tanh layer is used to generate some values that are possibly included in the state. The tanh and the sigmoid layers are combined to update the state. In the end, one final sigmoid layer generates the suggested output from the cell state. In the context of our drone intrusion detection system, the RNN-LSTM is utilized to detect any deviation from abnormal behavior. RNN-LSTM will be operating on the drone. However, the network has to learn the normal behavior to detect the deviation. The algorithm receives the parameters extracted from the drone traffic. These parameters are passed to the prediction function that tries to find the traffic anomaly. Once there is abnormal behavior, the system will alarm the controller for decision making.

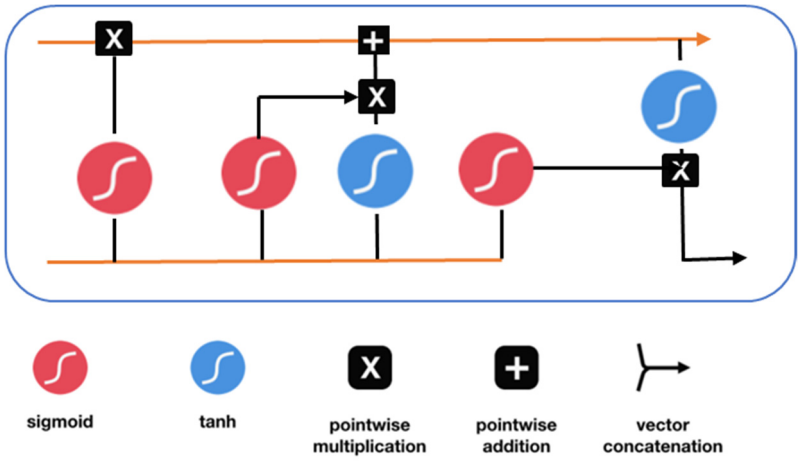


Figure 4. LSTM internal structure.

3.2. Data Collector

The data collectors' primary functions are to handle the data before feeding it to the RNN-LSTM module. This module also has the responsibility to stripe the data packets and extract the necessary features, such as transmission rate, reception rate, transmission to reception ratio, source IP, destination IP, transmission mode, and activity duration. Since our framework is designed to work in both data modes, batch and stream, as mentioned earlier, the data collector is assigned this task. Therefore, as seen in Figure 1, two collector modules are proposed in our framework; one of the data collector modules exists in each drone's component, and another resides at the base station component. Therefore, in batch data processing, the collector is configured to keep the data in a buffer before passing it to the RNN-LSTM module. The buffer size depends on the batch size to be processed, keeping in mind the drone limitations. In the case of the data stream mode, which is the case of our work in this paper, the data collector will be responsible for feeding the data as stream to the RNN-LSTM module. In other words, since we are simulating the drone's activities, the data collector emulates the real-time data processing and prepares it accordingly. However, in physical drones, which is not the case in this paper, the data collector will be responsible for intercepting the communication module's data and preparing it to fit the RNN-LSTM module requirements. In addition, the drone's data collector module is also responsible for transferring the collected data to the base station collector module along with the drone's RNN-LSTM module decision.

The base station data collector module receives all of the drones' data and their decisions. It processes all received data and passes them to the centralized RNN-LSTM module on the base station for decision verification. Then, it will forward the final decision to the decision-maker module for further processing.

3.3. Centralized RNN

Here, another RNN-LSTM is deployed on the base station. Once more, this module could work on batches or streams. It receives drone traffic from the data collector module either in batches or a stream based on the configured mode. The centralized RNN-LSTM will make a global decision based on the overall collected data to check which drone is compromised. The centralized RNN-LSTM module sends its decision to the decision-maker module. The centralized RNN will be more trained than the drones' RNN due to the traffic coming from the different drones.

3.4. Decision Maker

The decision-maker module's primary function is to analyze the drones received and the centralized RNN module decisions. The analysis is done through a voting process where three warnings are generated:

- Red: The red alarm means that the intrusion is certain and is identified by the two RNN modules. In this case, all of the networks are notified, and the exposed drone is isolated.
- Yellow: The yellow alarm means that one of the two modules identified the intrusion while the other module did not recognize the intrusion. They only allowed forwarding the received messages to the centralized RNN module. In such a case, the overall network is notified to stop dealing with the exposed drone for a certain period of time. This situation continues until further notification comes from the decision-making module.
- Green: This means that there is no need for any alarm where the drones are supposed to be safe. In this case, both the RNN modules confirm no intrusion on any of the drones.

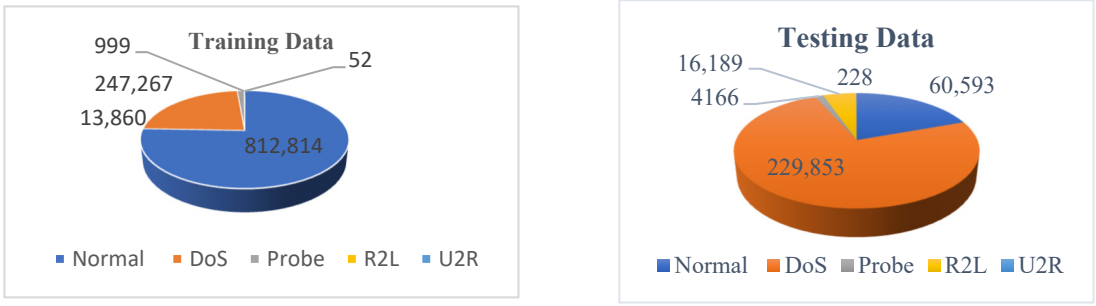
4. Results

In this section, the evaluation environment is explained, and different sets of experiments are introduced to examine the efficiency of the proposed approach. However, although the used datasets are the most used in the literature, to our knowledge, there are no drone datasets that can be used for intrusion detection examination in this paper. In addition, a practical collection of drone information is out of the scope of this paper, and it has been scheduled as future work in this article. However, we emulated the drone's activities by proposing the "Data Collector" module to stream the data to the RNN-LSTM module. RNN-LSTM is compared to Linear Regression (LR) and K-Nearest Neighbors (KNN) algorithms. Another set of experiments are conducted in this section to compare between our proposed machine learning approach and nonmachine learning approaches, signature-based approach.

4.1. The Used Datasets

The problem of intrusion detection in drones is still new, and to our knowledge, there is no specific dataset related to drone attacks. In addition, developing a practical drone network is out of the scope of this paper, and it has been added to the agenda for future work. Therefore, different standards datasets are utilized to examine the proposed framework's performance: KDDCup 99, NSL-KDD, UNSW-NB15, Kyoto, CICIDS2017, and TON_IoT. Those datasets are carefully chosen to cover a large number of attacks and examine the proposed algorithm using different attack WSN-DS procedures. Each dataset has special merit and a different number of records; therefore, selecting the exact percentage for all of the datasets will not be appropriate in most cases. Therefore, we had to adjust the percentages according to the number of records in the dataset as well as the nature of the attacks and their percentage in the datasets. Moreover, the IDS training is mostly supervised and done offline before putting the system into operation; therefore, the training percentages are controllable.

- KDDCup 99: This was collected based on tcpdump data DARPA intrusion detection challenge in 1998. The data were extracted at MIT Lincon laboratory through a setup of thousands of UNIX machines and hundreds of users accessing those machines. The data were collected for 10 weeks; the first 7 weeks' data were used for training, and the rest were used for testing. The 10 weeks' data are available to be used, but their size is huge to be used for the algorithm testing; 10% of the overall data has been made available for researchers and will be used in our experiments. The data, as shown in Figure 5, have 41 features to be examined and contain 5 classes of attacks ('Normal', 'DoS', 'Probe', 'R2L', 'U2R'). The detailed statistics of KDDCup are shown in Figure 2, including the training and testing data.



(**a**) (**b**)

Figure 5. KDDCup 99 (**a**) training data and (**b**) testing data.

- NSL-KDD: NSL-KDD is the distilled variant of KDD Cup 99 filters are used to eliminate duplicate records that are not required during data gathering. The discarded records were 136,489 and 136,497. The NSL-KDD dataset protects machine learning systems from overfitting. Using NSL-KDD in our implementation gave us further results over the KDDCup 99 dataset. However, the dataset suffers from not correctly reflecting the real-time traffic characteristics. NSL-DD statistical details, including the training and testing data per type of attack, are shown in Figure 6.

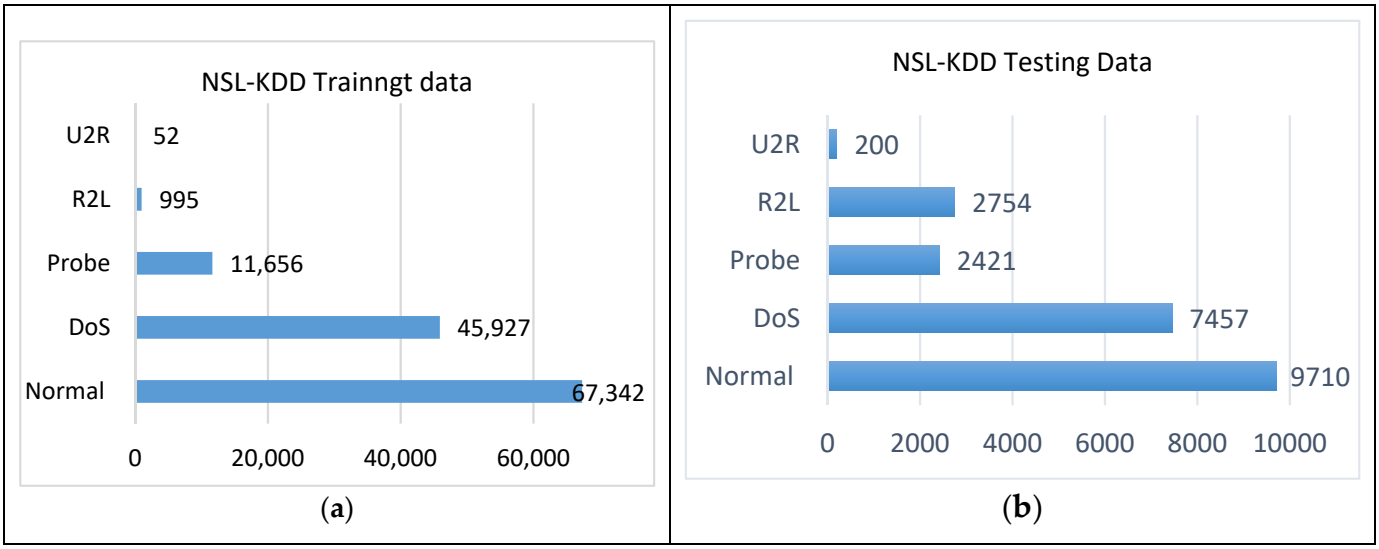


Figure 6. NSL-KDD (**a**) training data and (**b**) testing data.

- UNSW-NB15: This is a new dataset that addresses the KDDCup 99 and NSL-KDD datasets' problems. It was created by the Computer Security Research Team of the Australian Centre for Cyber Security (ACCS). It has the regular and attacks activities extracted by the IXIA Perfect Storm tool from a live network. Two servers were used to generate normal activities on a network, while another server was used to generate malicious activities. Tcpdump was used to collect network packet traces that compiled the network data from 100 GBs into 1000 MBs of pcaps. The pcap files were evaluated using Argus and Bro-IDS under Linux Ubuntu 14.0.3. In addition to the aforementioned methods, twelve algorithms were used to analyze the data in depth. The dataset comes in two forms: (i) full records with 2 million entries, and (ii) the data are partitioned into training and testing data. The training data consist of 82,332 records, whereas the testing data comprise 175,341 records, with 10 attacks included. The partitioned dataset includes 42 features and their classes are labeled as Normal and 9 different attacks. The detailed dataset statistics are given in Figure 7.

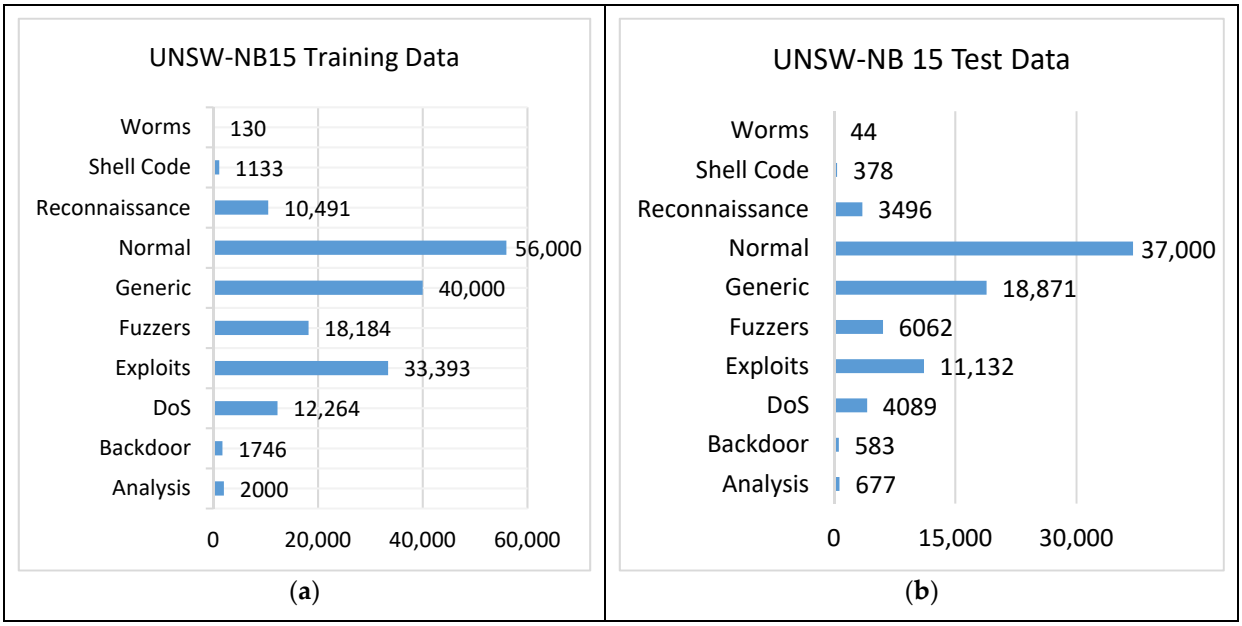


Figure 7. UNSW-NB15 (**a**) training data and (**b**) testing data.

- WSN-DS: This is an IDS dataset for WSNs which includes four types of DoS attacks: Blackhole, Grayhole, Flooding, and Scheduling. The data were collected using the LEACH technique, a low-energy adaptive clustering hierarchy protocol, then processed to generate 23 features. The dataset statistics are presented in Figure 8.

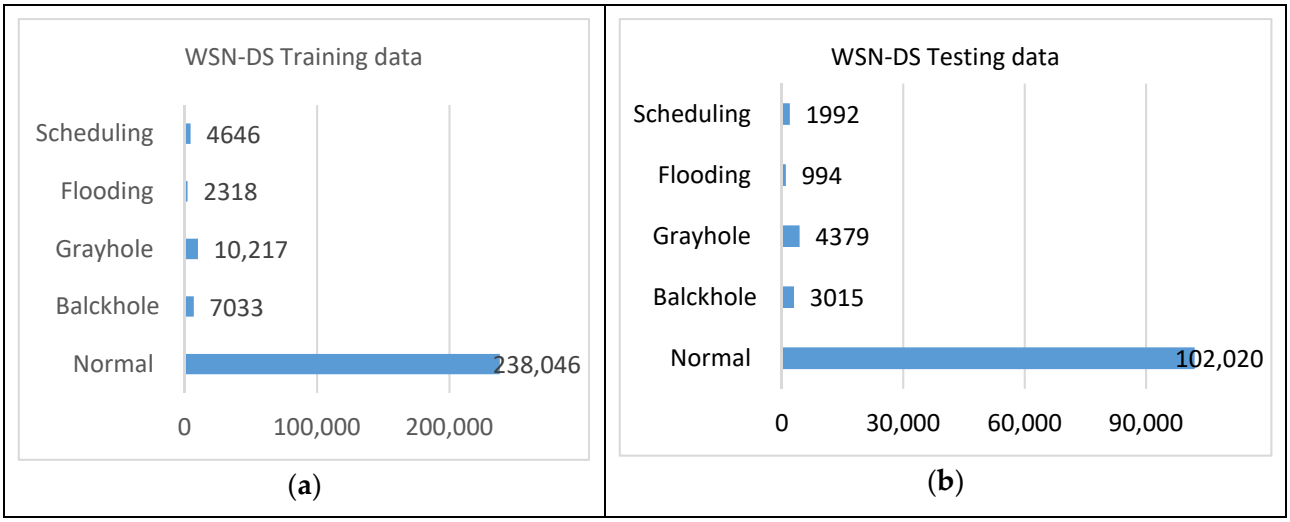


Figure 8. WSN-DS (**a**) training data and (**b**) testing data.

- CICIDS2017: This dataset contains real-time network traffic details for with attacks. The key interest is focused on real-time monitoring of the background traffic. The B-profile system is used to collect benign background traffic. This traffic includes 26 user characteristics focused on HTTP, HTTPS, FTP, SSH, and email protocols. Attacks like BruteForce FTP, Brute Force SSH, DOS, Heartbleed, Botnet, and DDoS were injected into the data. The statistics of CICIDS2017 are given in Figure 9.

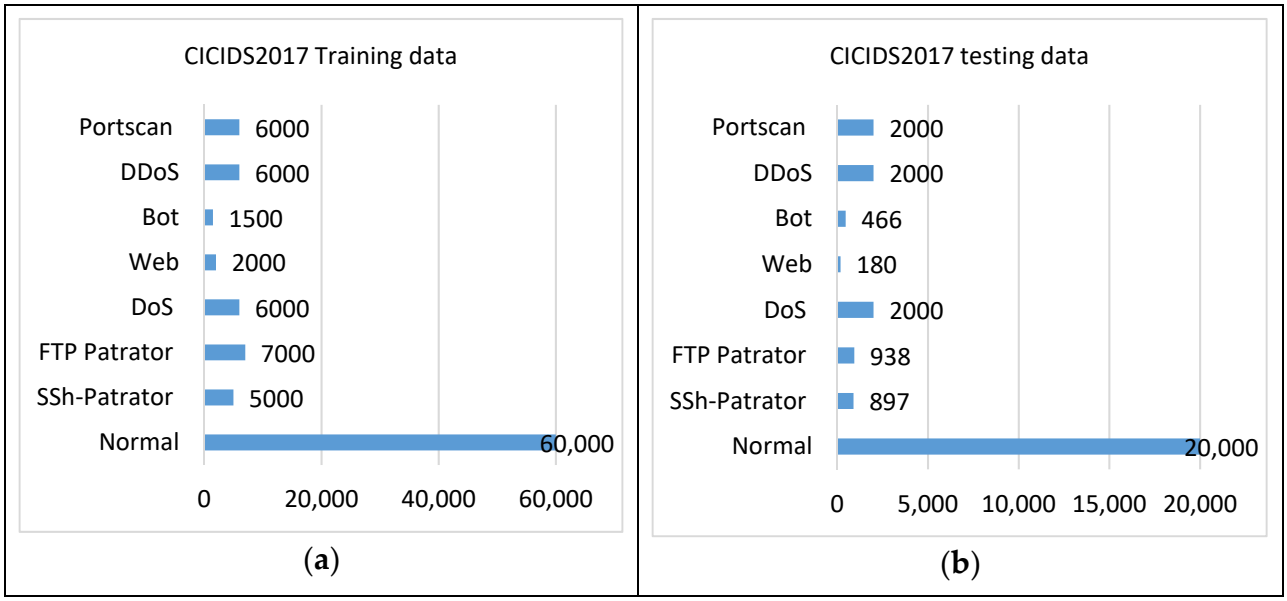


Figure 9. CICIDS2017 (**a**) training data and (**b**) testing data.

- TON_IoT [46]. TON_IoT uses a unique coordinated architecture that connects edge, fog, and cloud layers. The datasets include data from four sources, including operating systems, IoT/IIoT services, and network systems. The statistics were gathered via a realistic and large-scale testbed network built at UNSW Canberra's IoT Lab, part of the School of Engineering and Information Technology (SEIT). Table 3 contains the attacks categories of TON_IoT dataset:

Table 3. TON_IoT attacks categories.

Attack Type	TON_IoT No. of Records
Backdoor	508,116
DDoS	6,165,008
DoS	3,375,328
Injection	452,659
MITM	1052
Password	1,718,568
Ransomware	72,805
Scanning	7,140,161
XSS	2,108,944
Benign	796,380

4.2. Performance Measures

For the proposed framework's performance, some well-known metrics are used in similar situations. The records in the dataset are classified into normal and attacks. Therefore, the following parameters are considered for the appropriate measure to the performance of the proposed solutions:

- True Positive (TP): the number of correctly classified records toward the Normal class.
- True Negative (TN): the number of correctly classified records toward the Attack class.
- False Positive (FP): the number of Normal records wrongly classified toward the Attack class.
- False Negative (FN): the number of Attack records wrongly classified toward the Normal class.

Therefore, based on the previous parameters, the following methods were considered for the solutions performance evaluation in this paper:

- Accuracy: as can be seen in Equation (1), the accuracy is computed as the ratio of the correctly identified records to all of the records in the dataset. Certainly, as long as the accuracy is high, the model is considered performing better ($Accuracy \in [0; 1]$). The accuracy test helps calculate the balanced classes of the data we have.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- Precision: This is an attack measurement method used to calculate the ratio of the number of correctly classified attack records to the number of all identified attack records. Again, the precision is high, the model is considered to be performing better ($Precision \in [0; 1]$). The precision can be computed by Equation (2).

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

- True Positive Rate (TPR), Recall, or Sensitivity: This is the probability that an actual positive classification will test positive. Again, the higher the TPR, the higher the performance ($TPR \in [0; 1]$). The TPR can be computed using Equation (3).

$$TPR = \frac{TP}{TP + FN} \quad (3)$$

- F1-Score: This is another valuable performance measure, and it is also called F1-measure. The F-score is the relation between the harmonic mean of the precision and recall measures. If the *F1-score* is high, the proposed model performance ($F1-Score \in [0; 1]$) would be considered high. The *F1-score* is defined as given in Equation (4).

$$F1 - Score = 2\left(\frac{Precision\ x\ Recall}{Precision + \ Recall}\right) \quad (4)$$

5. Simulation Results

This section is dedicated to the experimental results to examine the performance of the proposed framework. The section starts by describing the simulation environment, followed by a description of the results. The proposed algorithm is compared to classical machine learning, such as Logistic Regression (LR) [47] and k-nearest Neighbors (KNN) [48].

5.1. Simulation Environment

TensorFlow was used to develop the deep learning algorithms on the drone and base station components to perform the experiments on the proposed framework. The full simulation was developed using pythons on an Intel Core i7-4790T CPU at 2.70GHz with 16MB RAM. The drone and base station LSTM_RNN modules were trained separately, assuming that the drone is an autonomous robot that should make its own decision and notify the base station of any threat that it may face through the collector component. Drones were assumed to run the LSTM_RNN IDS at the runtime while the collector collected the traffic and sent it in batches to the base station LSTM_RNN module. The data are configured to be fed into the drone LSTM_RNN module as a stream. The base station LSTM_RNN model receives all of the data from all of the drones in batches. Therefore, the base station cannot run on the same machine due to the high traffic to be tested. A cloud machine was prepared to rum Spark LSTM_RNN module for traffic drones traffic analysis as shown in Figure 10. Each of the previously described datasets was tested individually.

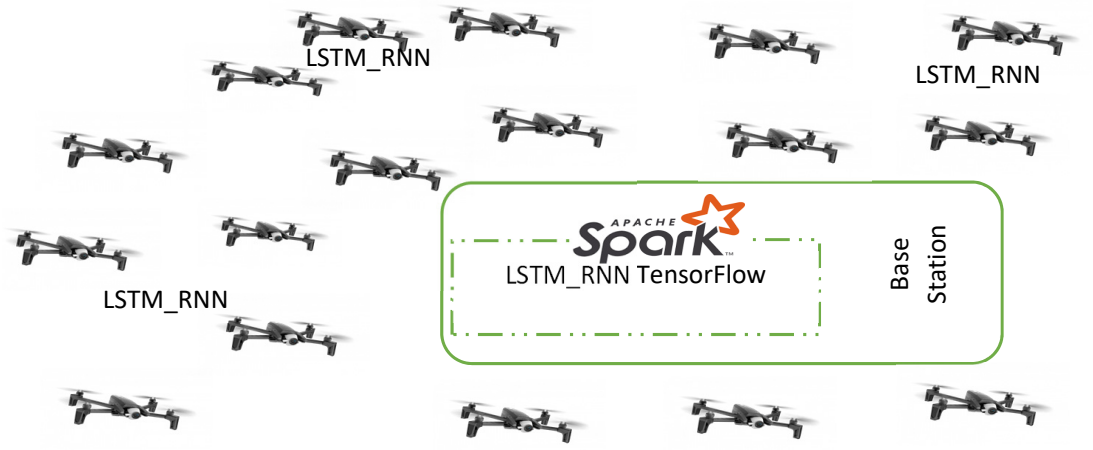


Figure 10. Base station Spark LSTM_RNN.

Some of the fields were normalized for the data to fit the LSTM_RNN, and some of the insignificant fields were discarded. Furthermore, the training and testing do not use all of the data once but in samples with different sample sizes. The sample size for both training and testing was chosen to be the same.

5.2. Performance Analysis

Though the first set of experiments examined the accuracy of the LSTM_RNN with different sample sizes, this was done using these datasets due to a large number of records and fields. The dataset was adopted to fit the proposed framework for drones. Figure 11 shows the detection accuracy versus sample size. As shown in the figure, LSTM_RNN accuracy increased with the increase of the sample size. It became more stable with the increase in the sample size. In addition, the average accuracy was 95%, and in some cases, it reached 97%. This is a good indicator of the performance of the proposed methodology.

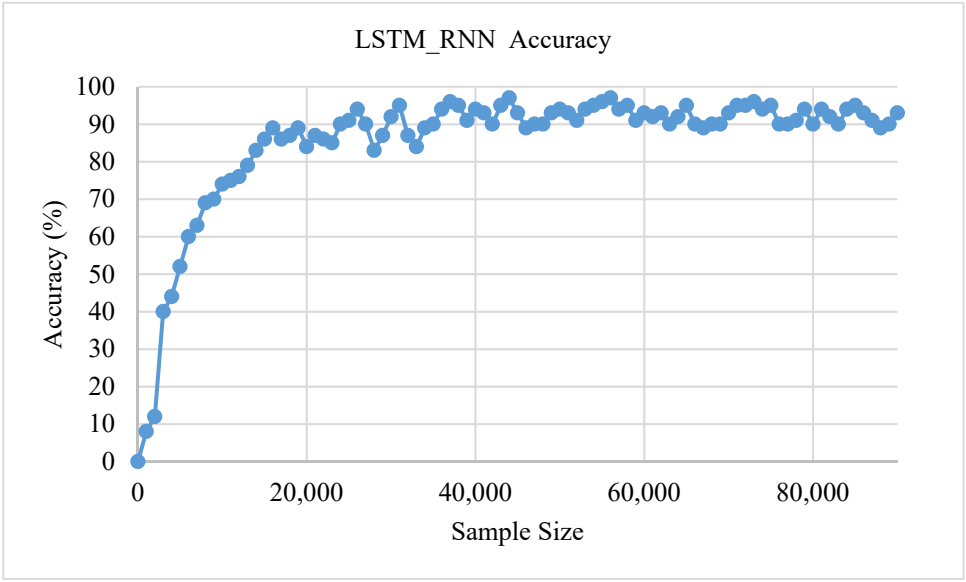


Figure 11. LSTM_RNN accuracy with the sample size.

5.2.1. LSTM_RNN Performance over UNSW-NB15 Dataset

Though the first set of experiments examined the accuracy of the LSTM_RNN with different sample sizes, this was done using these datasets due to a large number of records and fields. The dataset was adopted to fit the proposed framework for drones. Figure 11 shows the detection accuracy versus sample size. As shown in the figure, LSTM_RNN

accuracy increased with the increase of the sample size. It became more stable with the increase in the sample size. Moreover, the average accuracy was 95%, and in some cases, it reached 97%. This is a good indicator of the performance of the proposed methodology.

Figure 12 shows the comparison between the LSTM_RNN, LR, and KNN in binary classification. In this experiment, 10% of the dataset was used for training. As shown in the figure, LSTM_RNN performance showed a much better performance than LR and KNN algorithms. The accuracy of the LSTM_RNN reached 97%, while the Precision, Recall, and F-score were 94%, 98%, and 96%, respectively. On the other hand, KNN performed well with, on average, 91% Accuracy, Recall, and F-score followed by LR with almost 89%.

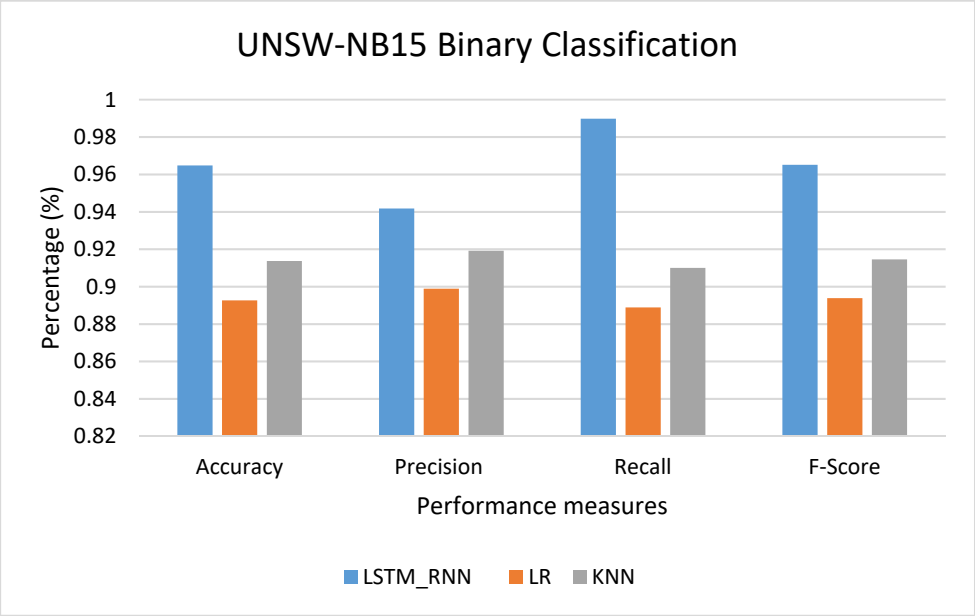


Figure 12. UNSW-NB15 Binary classification performance comparison.

Figure 13 shows another classification performance based on multiclass attacks. As can be seen, the LSTM_RNN and KNN had relative performance in terms of Accuracy and Precision. However, the Recall of LSTM_RNN, LR was much better than that of KNN, where the true positive value was much better than KNN. On the other side, LR still has lower precision, accuracy, Recall, and F-score than LSTM_RNN.

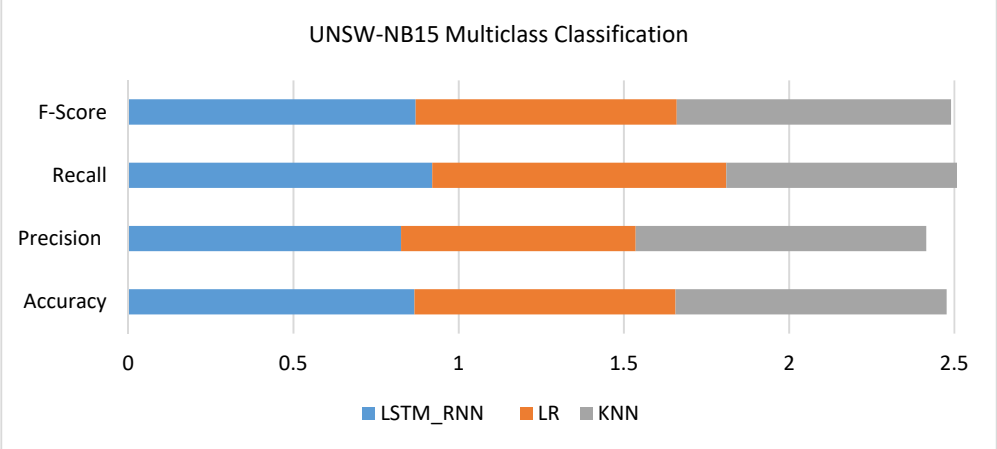


Figure 13. UNSW-NB15 Multiclass classification performance comparison.

Here is another set of experiments targeting the three algorithms' performance based on the UNSW-NB15 detailed attacks. As shown in Figure 14, LSTM_RNN showed a great

overall performance of the attacks presented in the UNSW-NB15 dataset. On average, LSTM_RNN outperformed KNN by 2% accuracy while it outperformed LR, on average, by 5% accuracy.

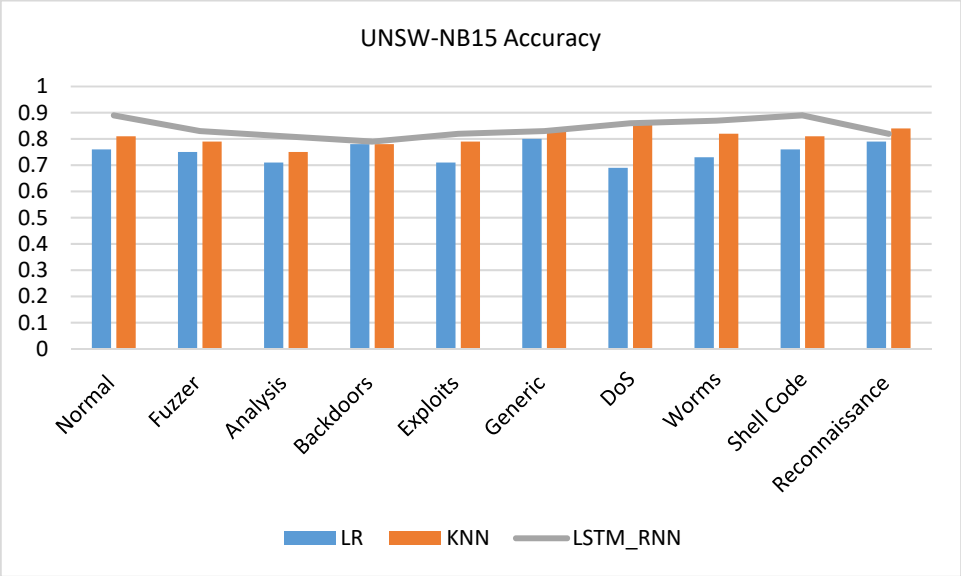


Figure 14. UNSW-NB15 Multiclass classification details accuracy comparison.

5.2.2. LSTM_RNN Performance over KDDCup 99 Dataset

Another set of experiments conducted over the KDDCup 99 Dataset to examine the performance of LSTM_RNN was compared to KNN and LR. Again, three sets of experiments were conducted, including the binary, multiclass, and detailed classifications. Figure 15 shows the algorithms' performance-based binary classification. As can be seen, LSTM_RNN shows a performance increase, on average, of 5% and 8% accuracy over KNN and LR, respectively. At the same time, LSTM_RNN outperformed both algorithms, KNN and LR, in precision, Recall, and F-score. It is worth mentioning that LSTM_RNN produced almost 94% and 95% True Positive and True Negative, respectively.

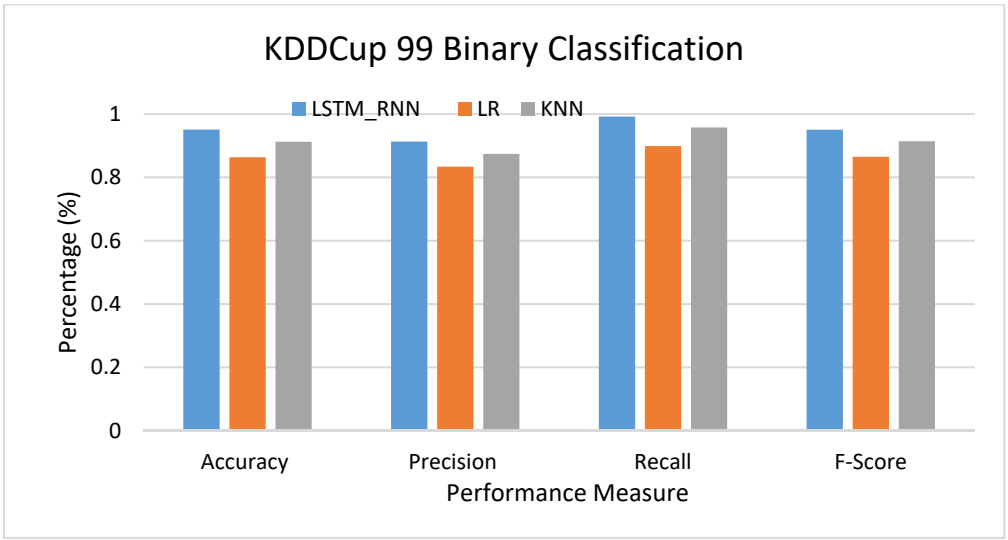


Figure 15. KDDCup 99 Binary classification performance comparison.

Figure 16 shows that LSTM_RNN produced almost 91% accuracy, 83% precision, and 99% recall. On the other hand, the KNN result shows 88% accuracy, 81% precision, and 96% recall. Similarly, LR gave 88% accuracy, 87% precision, and 99% recall. Therefore, it seems that LR outperforms the KNN due to the high false-negative rate of KNN. Applying the three algorithms for detailed classification over the KDDCup 99 Dataset, Figure 17 shows that LSTM_RNN and KNN performed well, on average, 100%, 100%, and 97% classifying U2L, U2R, and Probe attacks classification, respectively. Simultaneously, LSTM_RNN accuracy was 90% and 98% in classifying DoS and Normal attacks, respectively. Compared to KNN and LR, LSTM_RNN performed much better than both algorithms. However, it was noticed that KNN had 63% and 65% accuracy performance for DoS and Normal attacks, which are the lowest accuracy performance.

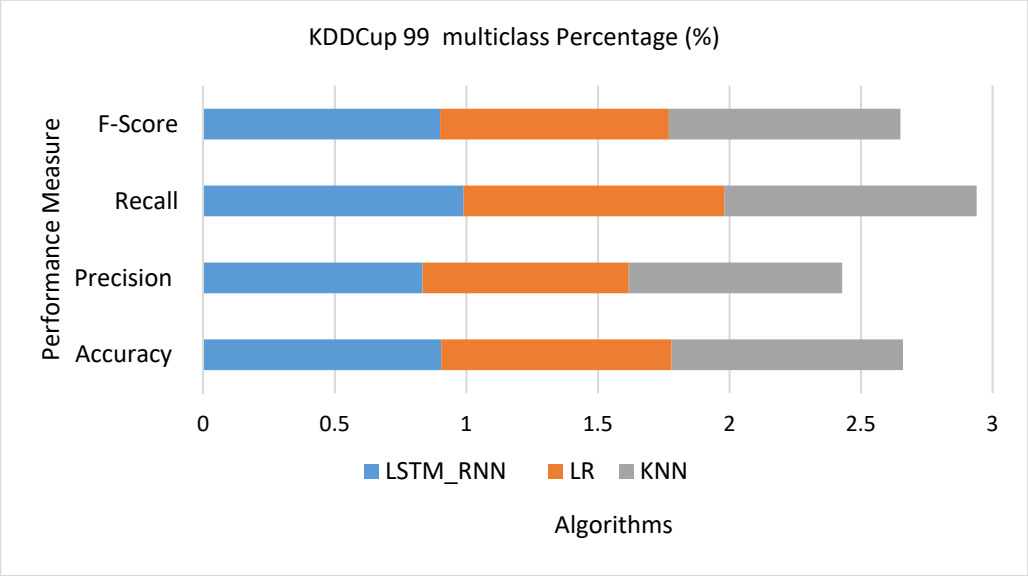


Figure 16. KDDCup 99 multiclass classification performance comparison.

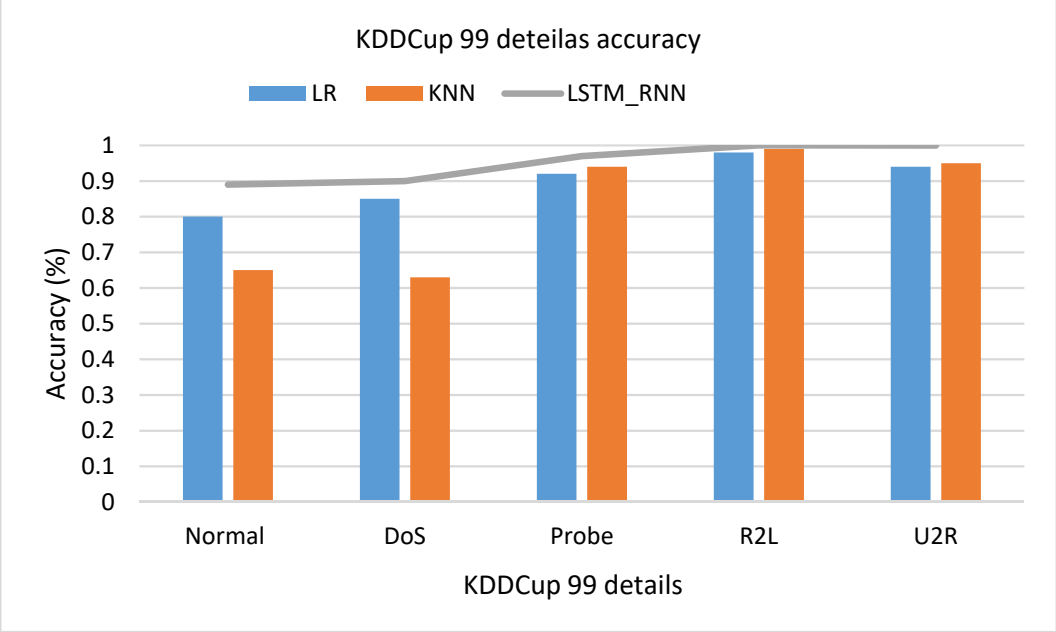


Figure 17. KDDCup 99 detailed classification details accuracy comparison.

5.2.3. LSTM_RNN Performance over NSL-KDD Dataset

NSL-KDD was another dataset used to examine the performance of the proposed LSTM_RNN algorithm. Again, the algorithm was examined on binary, multiclass, and detailed attack classification. As shown in Figure 18, LSTM-RNN outperformed LR and KNN algorithms for all performance measures except the precision where LR precision increased by 1% over LSTM_RNN. The false positive rate of LR was much better than that of LSTM_RNN, in this case.

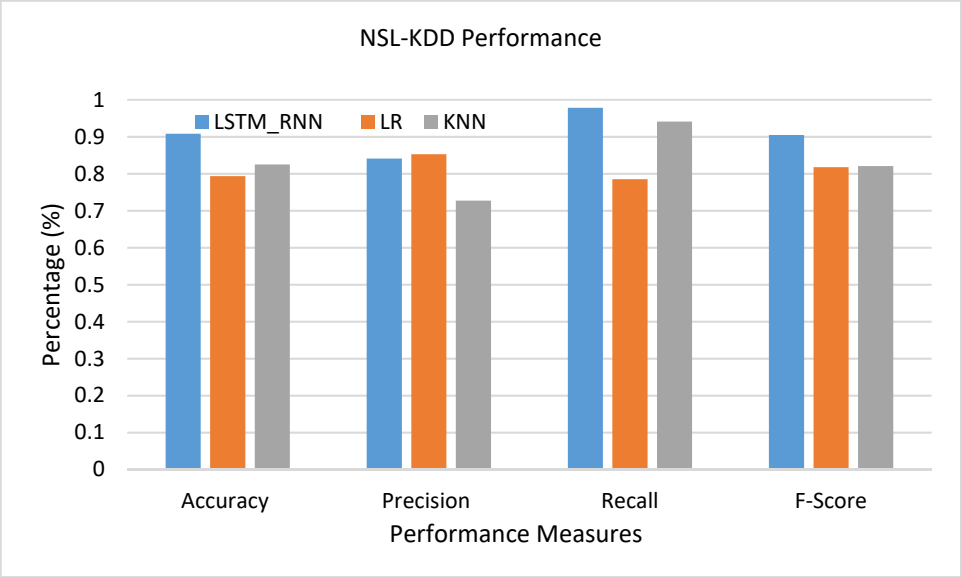


Figure 18. NSL-KDD Binary classification performance comparison.

LSTM_RNN in Figure 19 shows a better performance than LR and KNN algorithms with 88% accuracy, 80% precision, 98% recall, and 88% F-score. On the other hand, KNN showed good accuracy, Recall, and F-score but lower precision than LR, where the false positive rate of LR was lower than KNN by almost 1%.

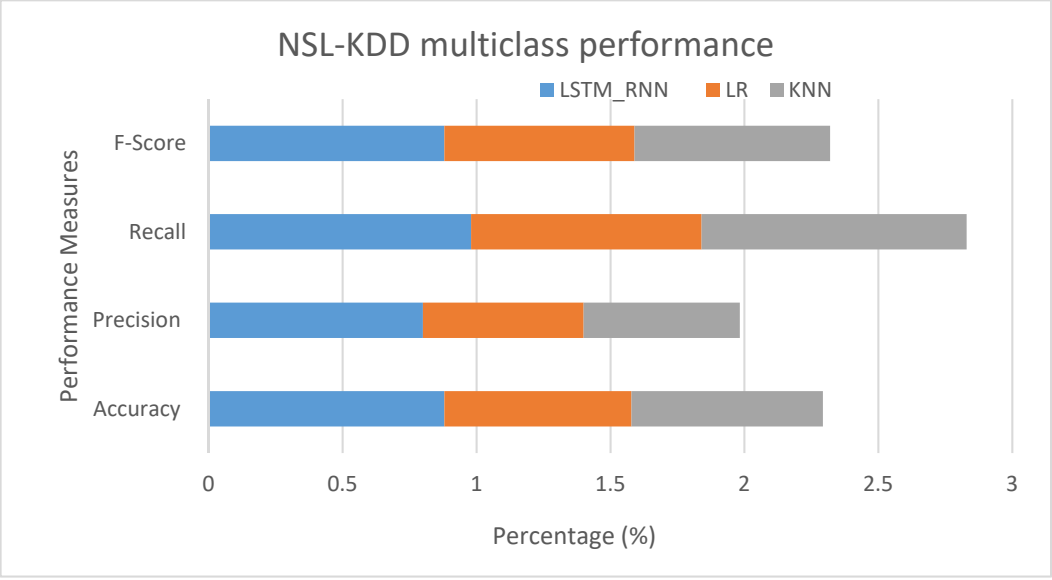


Figure 19. NSL-KDD Multiclass classification performance comparison.

In detailed class classification, Figure 20, LSTM_RNN resulted in 99% accuracy over R2L classification and 97% accuracy classifying U2R, while it had the lowest performance in classifying the Normal attacks with 85%. On the other hand, KNN and LR had a close accuracy but were still lower than LSTM_RNN.

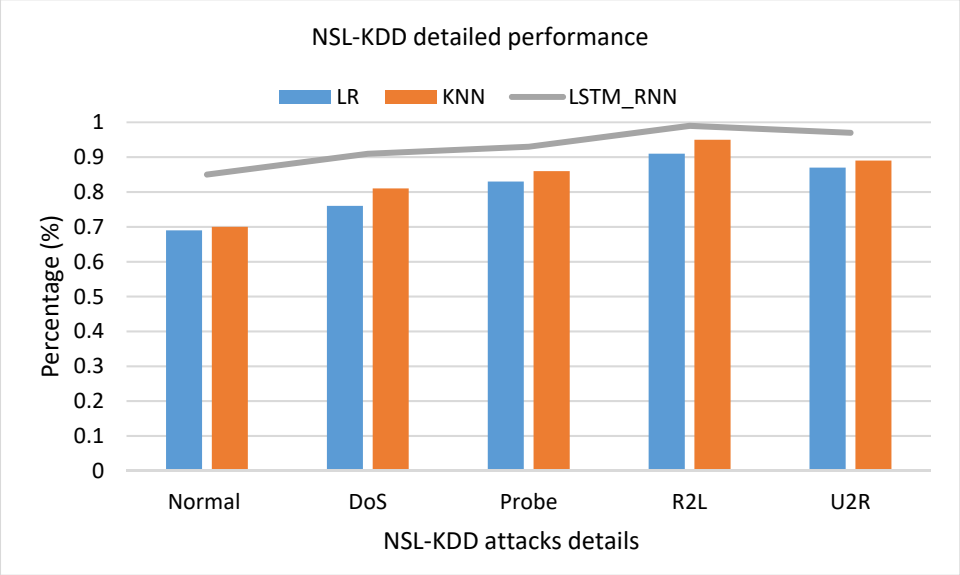


Figure 20. NSL-KDD detailed attacks classification accuracy comparison.

5.2.4. LSTM_RNN Performance over WSN-DS Dataset

Another set of experiments were conducted over WSN-DS Dataset. As shown in Figure 21, it seems that LR had a much better performance than LSTM_RNN and KNN, where the True Positive of LR was higher than LSTM_RNN and KNN by almost 1%. Almost the same results were confirmed over multiclass classification presented in Figure 22. However, looking at the detailed classification in Figure 23, it seems that LSTM_RNN still had better accuracy on classifying Scheduling, Flooding, and Normal attacks than the LR and KNN approaches.

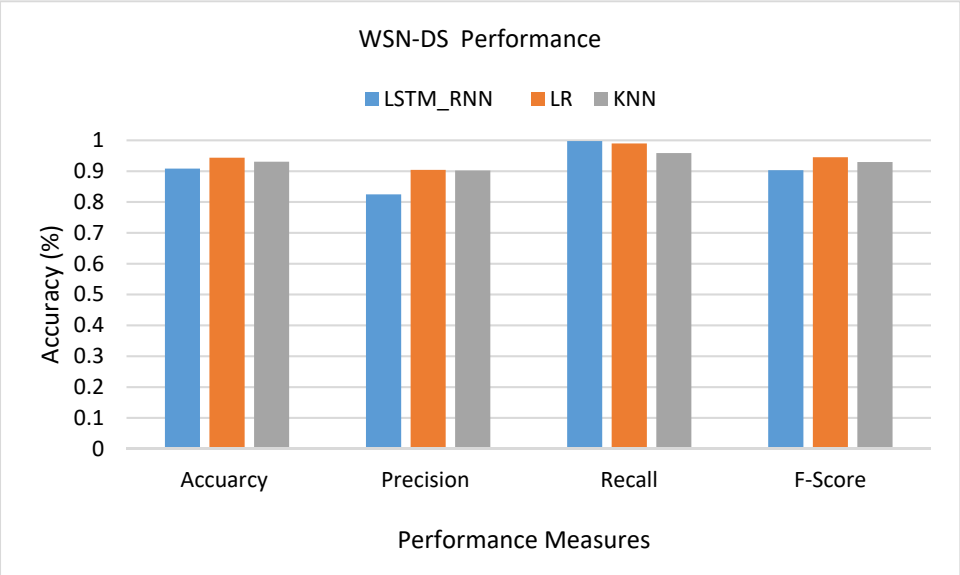


Figure 21. WSN-DS Binary classification performance comparison.

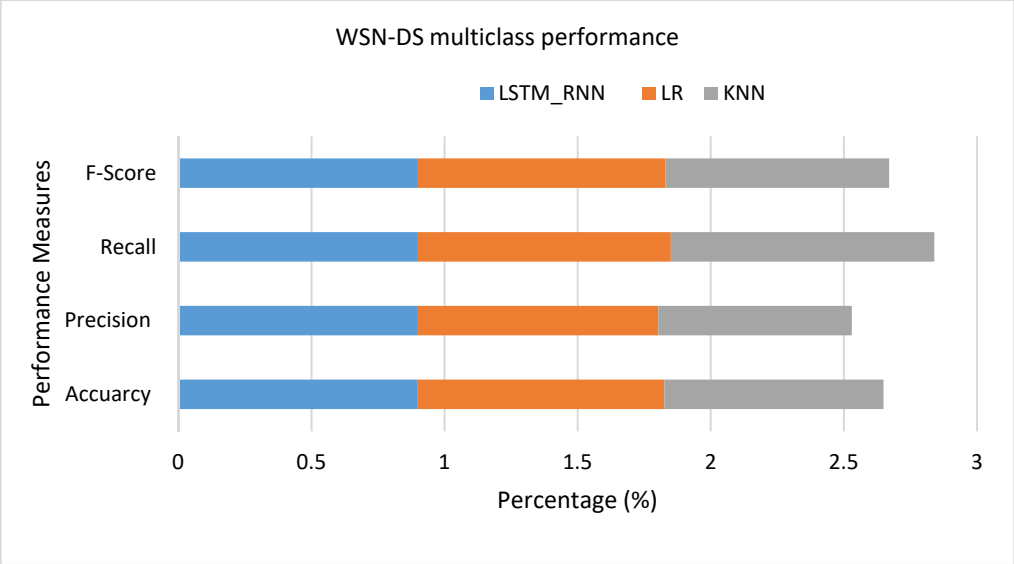


Figure 22. WSN-DS Multiclass classification performance.

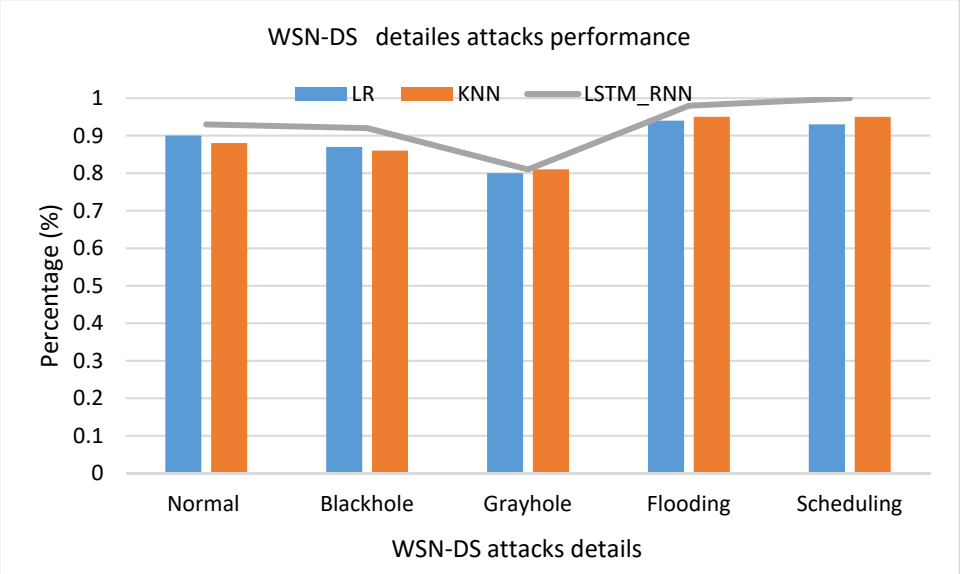


Figure 23. WSN-DS detailed attacks classification accuracy comparison.

5.2.5. LSTM_RNN Performance over CICIDS2017 Dataset

Under the CICIDS2017 dataset presented in Figure 24, LSTM_RNN was back to the front with higher performance than LR and KNN algorithms with 7% accuracy, 9% precision, 5% recall, and 7% accuracy F-score more than KNN. KNN showed a better performance than LR in most cases, but the Recall due to the false negative rate in KNN was slightly higher than in LR. The same results were confirmed in multiclass classification, Figure 25, with slightly lower Recall due to slightly high false negative. Investigating the classification of the detailed attack, shown in Figure 26, it seems that LR and KNN had almost the same performance based on all performance measures. However, LSTM_RNN shows the best performance. Simultaneously, all three algorithms could not perform well on classifying Grayhole attacks where, on average, they had 81% accuracy.

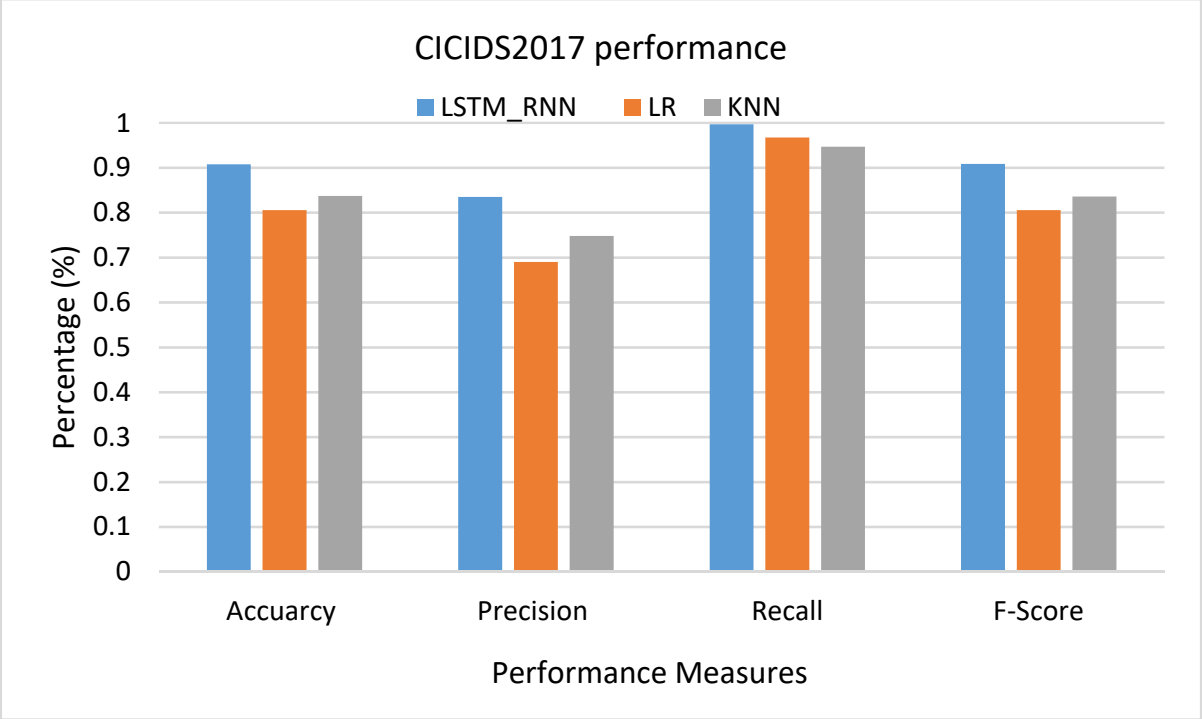


Figure 24. CICIDS2017 Binary classification performance.

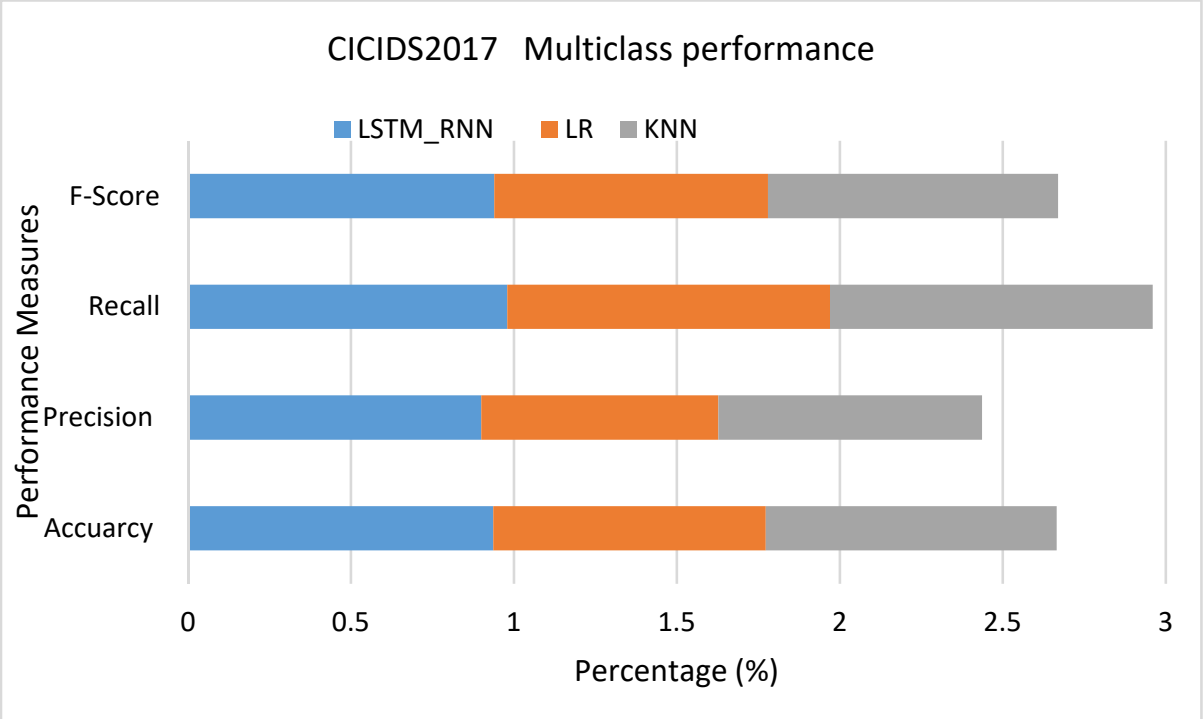


Figure 25. CICIDS2017 Multiclass classification performance comparison.

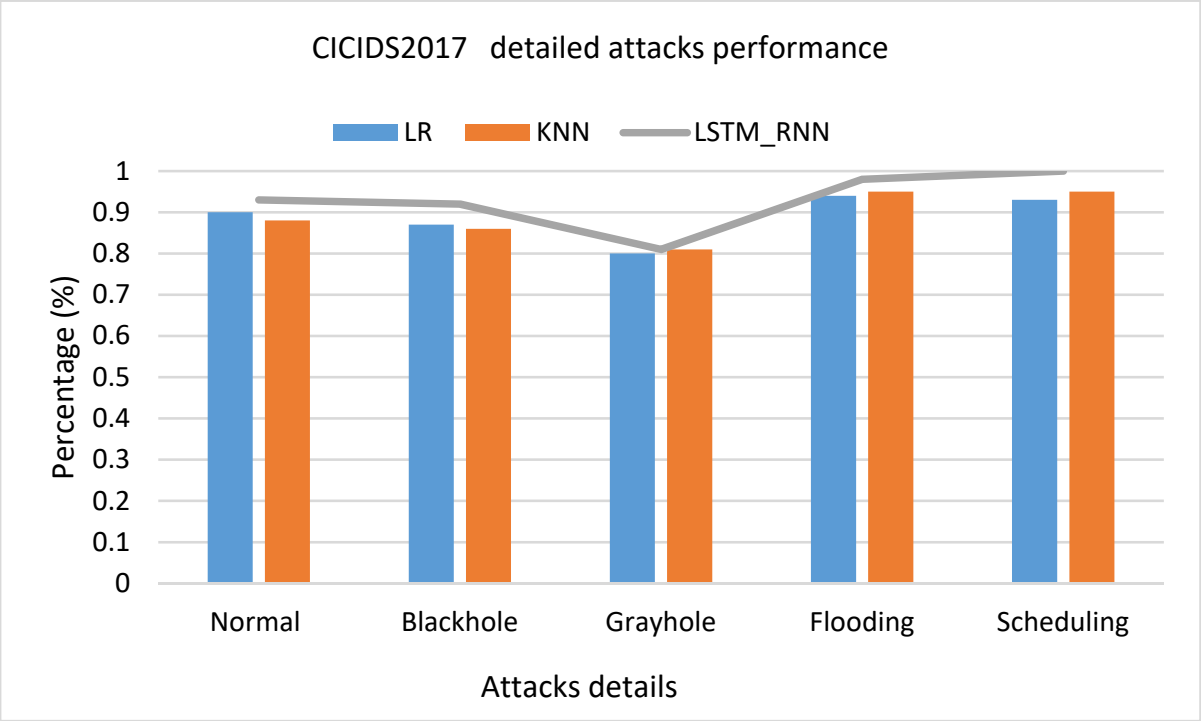


Figure 26. CICIDS2017 detailed attacks classification accuracy comparison.

5.2.6. LSTM_RNN Performance over ToN_IoT Dataset

Here is another set of experiments to examine LSTM_RNN performance. Table 4 shows the training sets and testing sets motivated by the work done in [46].

Table 4. Training set and testing set used in [46].

Attack	Training Set	Testing Set
Backdoor	12,000	8000
DDoS	12,000	8000
DoS	12,000	8000
Injection	12,000	8000
Mitm	625	418
Password	12,000	8000
Ransomware	12,000	8000
Scanning	12,000	8000
XSS	12,000	8000
Benign	180,000	120,000

Figure 27 shows the average performance of the LSTM_RNN, LR, and KNN-based ToN_IoT Dataset. As can be seen in the figure, the accuracy of LSTM_RNN seemed to outperform KNN and LR with almost 99%. However, LR showed the worst performance. At the same time, the recall of the three algorithms were close to each other.

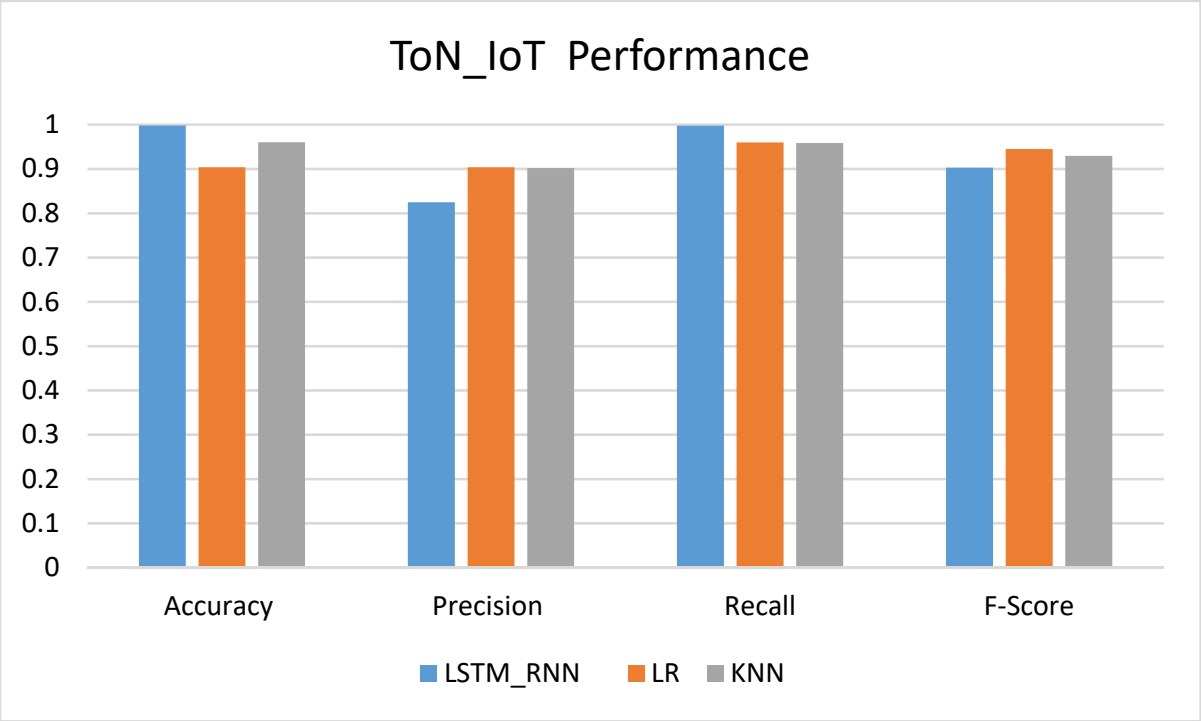


Figure 27. LSTM_RNN Performance Over ToN_IoT Dataset.

5.2.7. Signature-Based Approach vs. LSTM_RNN

Although the paper's main concern is machine learning approaches, this subsection examines one of the nonmachine learning approaches, specifically signature-based intrusion detection. We based our results on Snort [49] version 2.9.18 with the latest rules 23 September 2021. Snort is an open-source network intrusion prevention system that can analyze and record IP traffic in real-time. It can detect buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and many more attacks and probes. With the help of TcpReplay, we fed Snort with the dataset records. We selected UNSW-NB15 for this set of experiments because it is more recent than DDCup99 and NSL-KDD, and it handled their problems as well. A preprocessing step was conducted to remove all of the unnecessary information from the dataset. We also based our results in this section on binary classifications of the attacks.

Figure 28 shows the performance of both Snort and LSTM_RNN in terms of accuracy, precision, recall, and F-score. As expected, the LSTM_RNN performed much better than the Snort in every aspect. LSTM_RNN accurately detected the intrusion with 36% better than the Snort. In addition, it was clearly noticed that Snort had a low precision value of almost 1%. Therefore, we recommend that although machine learning algorithms require more time for training, they give more accurate results than nonmachine learning algorithms.

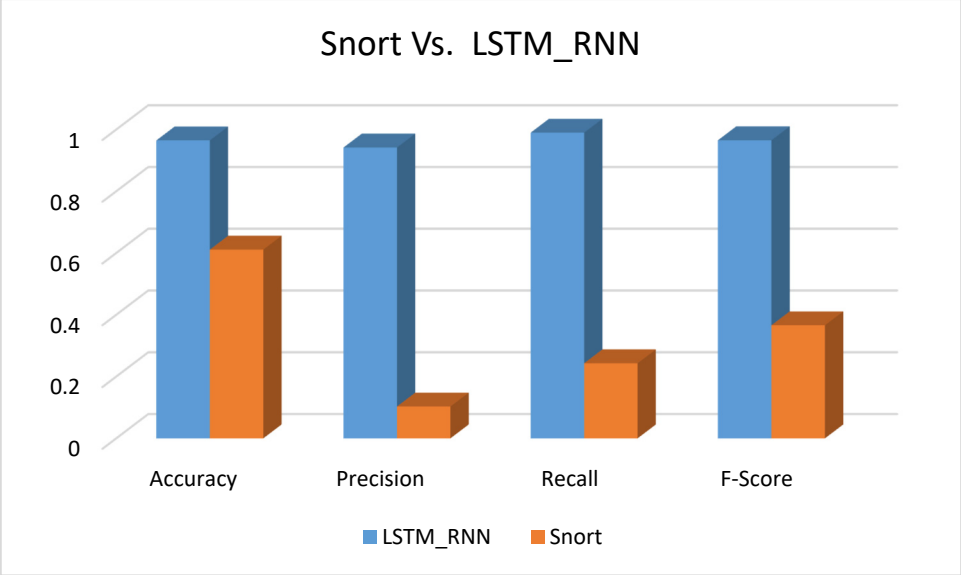


Figure 28. Signature-based performance Vs. LSTM_RNN.

5.2.8. Average Results and Discussion

Another possible view of the previous figures' data is the average results taken over all of the binary and multiclass classifications. The Critical Distance Diagrams (CDDs) from the post-hoc Nemenyi test were applied to the average results to better judge the algorithm's overall performance on the used datasets. To do so, the paper followed the same approach and guidelines presented in [50] to implement the test in R language. The four performance measures, Accuracy, Precision, Recall, and F-Score, were used to draw the diagram. As can be seen in Figures 29 and 30, the top line in the diagram represents the axis along which the average rank of each Spectro-temporal feature is plotted, starting from the lowest ranks (most important) on the left to the highest ranks (least important) on the right. The diagram works by grouping the algorithms with similar statistical values; those algorithms are connected. As shown in Figure 29, in binary classification, LSTM_RNN performed much better in all cases, which is why it is leftmost with a large distance between it and the other algorithms (LR and KNN). In addition, the figure shows that KNN is still ahead of the LR, with a small performance difference between them. Moreover, Figure 30 confirms the performance of the LSTM_RNN algorithm over the other algorithms. At the same time, it shows that KNN is in the middle between LR and LSTM_RNN with almost the same distance in between them. However, LR seems to fall behind both KNN and LSTM_RNN in the multiclass classification.

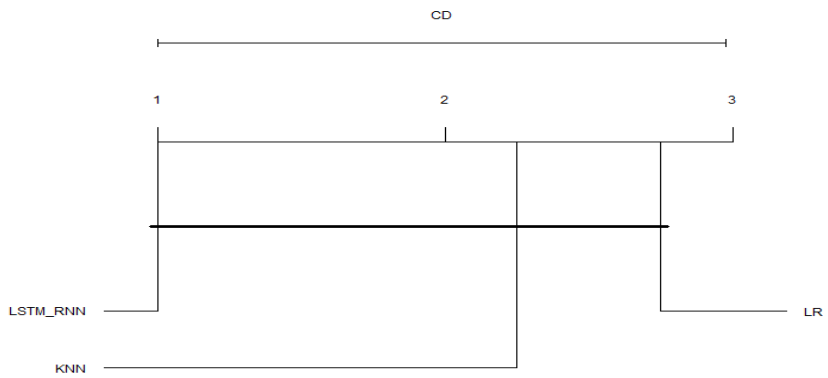


Figure 29. CDD graph for average binary classification.

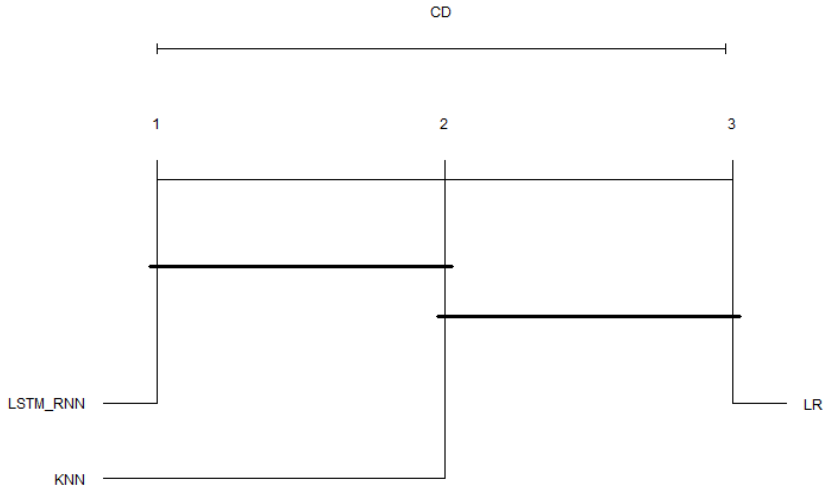


Figure 30. CDD graph for average multiclass classification.

6. Conclusions

The field of FANET has gained huge attention in recent years. It has been used in many civilian and military applications. However, there is still a gap in their securities. Current cryptography techniques might not be appropriate due to their complexities and power cost. Moreover, intrusion into such flying robots could be more dangerous than jamming or message interception. Therefore, this paper is considered one step towards an intelligent intrusion detection framework for Drones. The framework is based on two separate components: the drone and the Base station. It utilizes a particular type of RNN for both components, which is RNN-LSTM. The results showed superior performance with average performance, almost 15 on average, over some of the recent algorithms. One of the main restrictions of this paper is applying the proposed framework over real FANET. This is one of the future works following this paper. Another future direction is secure communication between the drones and the base station as well as between the drones themselves. In our future work, we intend to extend the proposed framework in this paper by implementing a lightweight algorithm based on a linear hyperbolic chaotic system of partial differential equations. The chaotic system is mainly applied to drones and satellite communication. Moreover, examining the cyberattacks against the application-layer communication of drones using different methods could be another interesting work to be conducted in a separate paper.

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

Funding: The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication.

Data Availability Statement: KDDCup 99 (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html); NSL-KDD (https://www.unb.ca/cic/datasets/nsl.html, accessed on 11 October 2021); UNSW-NB15 (https://research.unsw.edu.au/projects/unsw-nb15-dataset, accessed on 11 October 2021); Kyoto (http://www.takakura.com/Kyoto_data/, accessed on 11 October 2021); CICIDS2017 (http://www.unb.ca/cic/datasets/ids-2017.html, accessed on 11 October 2021); TON_IoT (https://ieee-dataport.org/documents/toniot-datasets, accessed on 11 October 2021).

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

References

1. Roopak, M.; Tian, G.Y.; Chambers, J. Deep Learning Models for Cyber Security in IoT Networks. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 452–457.
2. Liang, C.; Shanmugam, B.; Azam, S.; Karim, A.; Islam, A.; Zamani, M.; Kavianpour, S.; Idris, N.B. Intrusion detection system for the internet of things based on blockchain and multi-agent systems. *Electronics* **2020**, *9*, 1120. [CrossRef]
3. Statista Research Department. IoT: Number of Connected Devices Worldwide 2012–2025. Available online: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ (accessed on 20 May 2020).
4. Evans, D. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*; Cisco Internet Business Solutions Group (IBSG): San Jose, CA, USA, 2011.
5. Yuan, X.; Li, C.; Li, X. DeepDefense: Identifying DDoS Attack via Deep Learning. In Proceedings of the 2017 IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, China, 29–31 May 2017; pp. 1–8.
6. Gharibi, M.; Boutaba, R.; Waslander, S.L. Internet of Drones. *IEEE Access* **2016**, *4*, 1148–1162. [CrossRef]
7. Tarter, A. Importance of cyber security. In *Community Policing-A European Perspective: Strategies, Best Practices and Guidelines*; Springer: New York, NY, USA, 2017; pp. 213–230.
8. Li, J.; Qu, Y.; Chao, F.; Shum, H.P.; Ho, E.S.; Yang, L. Machine learning algorithms for network intrusion detection. In *AI in Cybersecurity*; Springer: New York, NY, USA, 2019; pp. 151–179.
9. Lunt, T.F. A survey of intrusion detection techniques. *Comput. Sec.* **1993**, *12*, 405–418. [CrossRef]
10. Ahmad, Z.; Khan, A.S.; Shiang, C.W.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4150. [CrossRef]
11. Debar, H.; Dacier, M.; Wespi, A. Towards a taxonomy of intrusion-detection systems. *Comput. Netw.* **1999**, *31*, 805–822. [CrossRef]
12. Humphreys, T. *Statement on the Vulnerability of Civil Unmanned Aerial Vehicles and Other Systems to Civil GPS Spoofing*; University of Texas at Austin: Austin, TX, USA, 2012; pp. 1–16.
13. He, D.; Chan, S.; Guizani, M. Drone-assisted public safety networks: The security aspect. *IEEE Commun. Mag.* **2017**, *55*, 218–223. [CrossRef]
14. Gudla, C.; Rana, M.S.; Sung, A.H. Defense techniques against cyber attacks on unmanned aerial vehicles. In Proceedings of the International Conference on Embedded Systems, Cyber-Physical Systems, and Applications (ESCS), Athens, Greece, 30 July–2 August 2018; pp. 110–116.
15. Shashok, N. Analysis of vulnerabilities in modern unmanned aircraft systems. *Tuft Univ.* **2017**, 1–10. Available online: http://www.cs.tufts.edu/comp/116/archive/fall2017/nshashok.pdf (accessed on 1 May 2021).
16. Hoque, M.S.; Mukit, M.; Bikas, M.; Naser, A. An implementation of intrusion detection system using genetic algorithm. *Int. J. Netw. Secur. Its Appl. (IJNSA)* **2012**, *4*. [CrossRef]
17. Prasad, R.; Rohokale, V. Artificial intelligence and machine learning in cyber security. In *Cyber Security: The Lifeline of Information and Communication Technology*; Springer: New York, NY, USA, 2020; pp. 231–247.
18. Lew, J.; Shah, D.A.; Pati, S.; Cattell, S.; Zhang, M.; Sandhupatla, A.; Ng, C.; Goli, N.; Sinclair, M.D.; Rogers, T.G.; et al. Analyzing machine learning workloads using a detailed GPU simulator. In Proceedings of the 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Madison, WI, USA, 24–26 March 2019; pp. 151–152.
19. Najafabadi, M.M.; Villanustre, F.; Khoshgoftaar, T.M.; Seliya, N.; Wald, R.; Muharemagic, E. Deep learning applications and challenges in big data analytics. *J. Big Data* **2015**, *2*, 1. [CrossRef]
20. Dong, B.; Xue, W. Comparison deep learning method to traditional methods using for network intrusion detection. In Proceedings of the 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), Beijing, China, 4–6 June 2016. [CrossRef]
21. Alnaghes, M.S.; Fayez, G. A Survey on Some Currently Existing Intrusion Detection Systems for Mobile Ad Hoc Networks. In Proceedings of the Second International Conference on Electrical and Electronics Engineering, Clean Energy and Green Computing (EEECEGC2015), Antalya, Turkey, 26–28 May 2015; Volume 12.
22. Sedjelmaci, H.; Senouci, S.M. An accurate and efficient collaborative intrusion detection framework to secure vehicular networks. *Comput. Electr. Eng.* **2015**, *43*, 33–47. [CrossRef]
23. Daeinabi, A.; Rahbar, A.G.P.; Khademzadeh, A. VWCA: An efficient clustering algorithm in vehicular ad hoc networks. *J. Netw. Comput. Appl.* **2011**, *34*, 207–222. [CrossRef]
24. Kumar, N.; Chilamkurti, N. Collaborative trust aware intelligent intrusion detection in VANETs. *Comput. Elect. Eng.* **2014**, *40*, 1981–1996. [CrossRef]
25. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, A. Hybrid intrusion detection system based on the stacking ensemble of c5 decision tree classifier and one class support vector machine. *Electronics* **2020**, *9*, 173. [CrossRef]
26. Zhang, H.; Dai, S.; Li, Y.; Zhang, W. Real-time distributed-random-forest-based network intrusion detection system using Apache spark. In Proceedings of the IEEE 37th International Performance Computing and Communications Conference (IPCCC), Orlando, FL, USA, 17–19 November 2018; pp. 1–7.
27. Maglaras, L.A. A novel distributed intrusion detection system for vehicular ad hoc networks. *Int. J. Adv. Comput. Sci. Appl.* **2015**, *6*, 101–106.

28. Parameshwarappa, P.; Chen, Z.; Gangopadhyay, A. Analyzing attack strategies against rule-based intrusion detection systems. In Proceedings of the Workshop Program of the 19th International Conference on Distributed Computing and Networking, Varanasi, India, 4–7 January 2018; pp. 1–4.
29. Patel, S.K.; Sonker, A. Rule-based network intrusion detection system for port scanning with efficient port scan detection rules using snort. *Int. J. Future Gener. Commun. Netw.* **2016**, *9*, 339–350. [CrossRef]
30. Cam, H.; Ozdemir, S.; Nair, P.; Muthuavinashiappan, D.; Sanli, H.O. Energy-efficient secure pattern based data aggregation for wireless sensor networks. *Com. Commun.* **2006**, *29*, 446–455. [CrossRef]
31. Zhang, J.; Zulkernine, M.; Haque, A. Random-forests-based network intrusion detection systems. *IEEE Trans. Syst. Man Cybern. C Appl. Rev.* **2008**, *38*, 649–659. [CrossRef]
32. Al-Jarrah, O.Y.; Siddiqui, A.; Elsalamouny, M.; Yoo, P.D.; Muhaidat, S.; Kim, K. Machine-learning-based feature selection techniques for large-scale network intrusion detection. In Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems Workshops, Madrid, Spain, 30 June 2014; pp. 177–181.
33. Su, M.Y. Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers. *Expert Syst. Appl.* **2011**, *38*, 3492–3498. [CrossRef]
34. Rani, M.S.; Xavier, S.B. A hybrid intrusion detection system based on C5.0 decision tree and one-class SVM. *Int. J. Curr. Eng. Technol.* **2015**, *5*, 2001–2007.
35. Yi, Y.; Wu, J.; Xu, W. Incremental SVM based on reserved set for network intrusion detection. *Expert Syst. Appl.* **2011**, *38*, 7698–7707. [CrossRef]
36. Amor, N.B.; Benferhat, S.; Elouedi, Z. Naive bayes vs decision trees in intrusion detection systems. In Proceedings of the 2004 ACM symposium on Applied computing, Nicosia, Cyprus, 14–17 March 2004; pp. 420–424.
37. Musafer, H.; Abuzneid, A.; Faezipour, M.; Mahmood, A. An enhanced design of sparse autoencoder for latent features extraction based on trigonometric simplexes for network intrusion detection systems. *Electronics* **2020**, *9*, 259. [CrossRef]
38. Abdulhammed, R.; Musafer, H.; Alessa, A.; Faezipour, M.; Abuzneid, A. Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection. *Electronics* **2019**, *8*, 322. [CrossRef]
39. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
40. Salo, F.; Injadat, M.; Nassif, A.B.; Shami, A.; Essex, A. Data mining techniques in intrusion detection systems: A systematic literature review. *IEEE Access* **2018**, *6*, 56046–56058. [CrossRef]
41. Thaseen, I.S.; Kumar, C.A. Intrusion detection model using fusion of chi-square feature selection and multi class SVM. *J. King Saud Univ. Comput. Inf. Sci.* **2017**, *29*, 462–472.
42. Kim, G.; Lee, S.; Kim, S. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Syst. Appl.* **2014**, *41*, 1690–1700. [CrossRef]
43. Al-Yaseen, W.L.; Othman, Z.A.; Nazri, M.Z.A. Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Syst. Appl.* **2017**, *67*, 296–303. [CrossRef]
44. Muniyandi, A.P.; Rajeswari, R.; Rajaram, R. Network anomaly detection by cascading k-Means clustering and C4. 5 decision tree algorithm. *Procedia Eng.* **2012**, *30*, 174–182. [CrossRef]
45. Jabbar, M.A.; Aluvalu, R. RFAODE: A novel ensemble intrusion detection system. *Procedia Comput. Sci.* **2017**, *115*, 226–234. [CrossRef]
46. Moustafa, N. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustain. Cities Soc.* **2021**, *72*, 102994. [CrossRef]
47. Besharati, E.; Naderan, M.; Namjoo, E. LR-HIDS: Logistic regression host-based intrusion detection system for cloud environments. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 3669–3692. [CrossRef]
48. Awad, N.A. Enhancing Network Intrusion Detection Model Using Machine Learning Algorithms. *CMC-Comput. Mater. Contin.* **2021**, *67*, 979–990. [CrossRef]
49. Snort Tool. Available online: https://www.snort.org/faq/what-is-snort (accessed on 23 September 2021).
50. Calvo, B.; Santafé, R.G. scmamp: Statistical comparison of multiple algorithms in multiple problems. *R J.* **2016**, *8/1*, 248–256. [CrossRef]