Article

Intelligent Cyber-Attack Detection and Classification for Network-based Intrusion Detection Systems

Nuno Oliveira , Isabel Praça , Eva Maia and Orlando Sousa

Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development (GECAD), Porto School of Engineering (ISEP), 4200-072 Porto, Portugal
Correspondence: {nunal; icp; egm; oms}@isep.ipp.pt

Received: date; Accepted: date; Published: date

Abstract: With the latest advances in information and communication technologies, greater amounts of sensitive user and corporate information are constantly shared across the network making it susceptible to an attack that can compromise data confidentiality, integrity and availability. Intrusion Detection Systems (IDS) are important security mechanisms that can perform a timely detection of malicious events through the inspection of network traffic or host-based logs. Throughout the years, many machine learning techniques have proven to be successful at conducting anomaly detection but only a few considered the sequential nature of data. This work proposes a sequential approach and evaluates the performance of a Random Forest (RF), a Multi-Layer Perceptron (MLP) and a Long-Short Term Memory (LSTM) on the CIDDS-001 dataset. The resulting performance measures of this particular approach are compared with the ones obtained from a more traditional one, that only considers individual flow information, in order to determine which methodology best suits the concerned scenario. The experimental outcomes lead to believe that anomaly detection can be better addressed from a sequential perspective and that the LSTM is a very reliable model for acquiring sequential patterns in network traffic data, achieving an accuracy of 99.94% and a f1-score of 91.66%.

Keywords: Intrusion Detection Systems; Anomaly detection; Sequential analysis; Random Forest; Multi-Layer Perceptron; Long-Short Term Memory

1. Introduction

Nowadays, information and communication technology plays a vital role in the life of modern organizations. Its development have opened a wide range of new communication methods that allow faster and cheaper ways to access and share information. Modern companies increasingly rely on these technologies to provide greater availability for its clients and for managing their businesses. This dependency on information systems causes a substantial amount of sensible user and corporate information to be shared across the network making it more susceptible to cyber-attacks that can compromise data confidentiality, integrity and availability.

An Intrusion Detection System (IDS) dynamically monitors actions of a specific environment, for example, network traffic, syslog records or system-calls of a given operating system, in order to determine if those actions are a legitimate use or a symptom related to a given attack [1]. These systems are usually classified into Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS). A NIDS works on feature vectors that comprise summarized information related to network traffic within a specified time interval while a HIDS is located on a specific host and monitors information related to the system [2].

Intrusion detection can be performed in several ways and normally they are described as Anomaly-based Intrusion Detection or Signature-based Intrusion Detection. The last, also known as misuse detection, attempts to detect and classify attacks by matching predefined patterns. This technique although maintaining reasonable levels of false alarms rates it is only good for well-known attacks [3]. In order to overcome this disadvantage some researchers have developed flexible signature-based NIDS [4,5]. On the other hand, anomaly detection aims to elaborate heuristic rules or statistical models based on the analysis of normal and abnormal activities that can further be used to classify some behaviour as benign or malicious. These systems are able to detect novel attacks and can be combined with artificial intelligence methods in order to increase their detection performance.

The advances in hardware, software and network topologies, like Internet of Things (IoT), causes cyber attacks to be more complex, sophisticated and, therefore, harder to detect [6]. This dynamic nature of cyber attacks makes anomaly detection an interesting area to employ artificial intelligence algorithms like Neural Networks or more traditional classifiers, like Random Forests (RF), k-Nearest Neighbors (kNN) or Support Vector Machines (SVM). The learning process of these algorithms can be classified as either Supervised, Unsupervised or Semi-supervised. A Supervised approach implies the presence of labeled data from which the algorithm will learn broad patterns that can later be used to assign unseen data to its correspondent class. This classification can either be Binary, for example, when classifying a given network traffic feature vector as either benign or malicious, or it can consider multiple classes. In a Multi-class situation a malicious event can be labeled as a specific attack. This way, artificial intelligence methods can be used not only to detect abnormal behaviour but also to determine the nature of that behaviour in order to understand what type of attack has been performed.

Significant and realistic data from emulated network environments is required for testing and comparing different anomaly detection approaches and methods. The absence of reliable datasets have been appointed in literature as one of the main obstacles for intrusion detection research [7]. However, in the last few years some intrusion detection datasets have been published, namely CIDDS-001 [8] , CICIDS2017 [9] and UNSW-NB15 [10]. For a detailed comparison between several NIDS datasets readers can be redirected to [11].

In our research, anomaly detection for the CIDDS-001 dataset was addressed from two distinct viewpoints. The first considers only individual flows while the second performs a deeper analysis on flow sequences. For each viewpoint three state of the art artificial intelligence models are employed - RF, MLP and LSTM. By analysing these methods results through multiple evaluation metrics its possible to better understand which approach best suits the dataset and which model achieves the best performance. The objective of these techniques is not only to detect malicious behaviour but also to specify the nature of that behaviour by identifying the respective attack type though a multi-class classification process.

As far as we are aware, multi-class classification was never addressed in order to detect distinct attack types for the CIDDS-001 dataset. On the other hand, only in [12], LSTM's were used to address intrusion detection from a sequential perspective, combining patterns from both individual flows and sequence of flows. However, the experimental results were obtained in the context of the UNSWNB-15 dataset, which is older than CIDDS-001 and regards distinct types of attacks. With our work, we hope to highlight the benefits of using machine learning for processing a stream of network flows to accurately detect malicious behaviour on network traffic.

This work was developed in the context of the SATIE - Security of Air Transport Infrastructures of Europe, as a research for a novel approach to investigate about temporal correlation between cyber and physical alerts. The SATIE project will build a holistic, interoperable and modular security toolkit to be exploited by the next generation of Airport Operation Centre and Security Operation Centre in order to protect critical air transport infrastructures against combined cyber-physical threats [13].

The paper is organized in multiple sections that can be detailed as follows. Section 2 provides a comparison between the current research and previous works on the CIDDS-001 dataset as well as the description of other implementations of the same methods for other relevant datasets. Section 3 briefly explains the machine learning models that were used, the problem's viewpoints and the considered evaluation metrics. In Section 4 the obtained results are presented and discussed. Section 5 provides a summary of the main conclusions that can be drawn from this research and appoints further research topics that can be addressed.

2. Related Work

In the last few years, several works have tested different machine and deep learning models on the CIDDS-001 dataset. These experiments addressed the target variable *Class* and, in general, attempted to classify each flow as either *Suspicious*, *Unknown*, *Normal*, *Attacker* or *Victim*. This particular work, although being related to previous researches, considers a different target variable and, therefore, performs a substantially different analysis on the dataset. The *AttackType* attribute was used as label in order to train the selected models to not only identify the occurrence of a given attack but also to detect the type of attack that has been performed. Nevertheless, it is very important to deeply understand what conclusions have been drawn from prior and related work.

In [14], Tama *et al.* evaluated the performance of a Deep Neural Network (DNN) for conducting anomaly detection in IoT environment. To assure a reliable performance analysis, several validation methods, such as cross-validation and repeated cross-validation, were performed on different datasets, namely, CIDDS-001, UNSW-NB15 and GPRS [15]. A grid-search was conducted in order to find the appropriate hyper-parameters of the DNN for each dataset. The model achieved nearly 100% accuracy for the the CIDDS-001 dataset (using Class as target variable).

In [16], Verma and Ranga performed an analysis on the CIDDS-001 dataset from the machine learning point of view. Both k-NN Classifier and k-Means Clustering methods performed well. A comparative analysis between CIDDS-001 and other existing benchmarking datasets is appointed as a future research topic in order to measure the complexity of this dataset over others.

In [17], Althubiti *et al.* used the CIDDS-001 External Server data to test a Long-Short Term Memory (LSTM) model. The data was split into 67% for training and 33% for testing. The LSTM achieved greater performance when compared with other methods, namely SVM, Naïve Bayes (NB) and Multi-Layer Perceptron (MLP). Although sequence size is not mentioned, most of the hyperparameter values that were used are described in detail. Nicholas *et al.* in [18] also evaluated the performance of an LSTM model in the flow-based data of CIDDS-001 and compared the obtained results with other traditional classifiers.

In [19], Abdulhammed *et al.* addressed the problem of disproportional class distribution. The research considered only the flows which corresponded to the target values *Normal* and *Attack* for the dataset's *Class* label and employed several class balancing techniques, such as, minority class up-sampling, majority class down-sampling, spread sub-sampling and class balancing by assigned weights inversely proportional to class frequencies. The authors, although believing that the results may not be generalized to a broader range of problems, concluded that class unbalance had a light impact on the classification process for that specific situation. Advantages of the usage of RF for anomaly detection have been appointed, such as, the ability to estimate missing data and maintaining reasonable accuracy values even when a large proportion of data is missing.

Finally, in [2], Rashid *et al.* performed a comparative analysis between the benchmark datasets NSL-KDD and CIDDS-001. A hybrid feature selection method was used in order to reduce the number of attributes of each dataset and six machine learning models were tested. For the CIDDS-001 dataset, NB and k-NN achieved the best results with an accuracy score of 99%. A detailed comparison between deep learning methods, such as, Convolutional Neural Networks (CNN), Deep Belief Networks (DNB) and Recurrent Neural Networks (RNN) have been appointed as further research. The authors also mentioned the possibility of applying feature learning on the raw data of network traffic headers in order to stimulate the maximum potential of neural networks.

Other works performed on different up-to-date and state of the art datasets were also important to this research and, therefore, should be carefully described.

In [20], Roy and Cheung, presented, within the IoT environment, a novel technique for anomaly detection using a Bi-directional LSTM network. The model was trained and tested in a smaller randomly selected sample of the UNSW-NB15 dataset and obtained substantially good results, namely an accuracy score of 95.71% and a f1-score value of 98.00%. Similarly to [17], sequence size was not mentioned.

In [12], Gwon *et al.*, used an LSTM model with a feature embedding layer on the UNSW-NB15 dataset obtaining excellent results. Instead of using traditional encoding methods for the categorical variables, the authors employed a word embedding layer that transformed those variable into numeric vectors that were able to translated their inherited semantic meaning. Network-based anomaly detection was addressed from a sequential perspective and flaws of traditional classifiers were appointed since they struggle to learn sequential patterns from data. The LSTM model was selected to overcome this problem since it can analyse entire sequences of flows and learn meaningful patterns from sequential data. Learning strategies such as many-to-one or many-to-many were carefully studied and the performance of the model was analysed for different sequence sizes. For binary classification the model achieved an accuracy score of 99.72% while for multi-class the accuracy score was 86.98%.

In [3], He *et al.*, proposed a combination between a Multimodal Deep Auto Encoder (MDAE) and a LSTM for conducting anomaly detection. This novel approach was tested on three datasets from 1999 to 2017, namely, NSL-KDD, UNSW-NB15 and CICIDS2017 achieving, for multi- class classification, accuracy scores of 80.20%, 86.20% and 98.60%, respectively.

Our work mainly differs from the others presented because, to best of our knowledge:

- No work has addressed the target variable *AttackType* of the CIDDS-001 in order to perform multi-class classification for intrusion detection;
- Only Gwon *et al.* in [12] has addressed network intrusion detection from a sequential perspective by using LSTM. However, the experimental results were achieved for the UNSWNB-15 dataset which is two years older than the CIDDS-001, contains considerable less data for training and testing and considers distinct attack types.
- Only in [17] LSTM was used for the CIDDS-001 dataset. Nevertheless, the work does not mention any study about the flow window size and its impact in the model's evaluation metrics. It is also important to highlight that in [17] only the External Server data was used and the *Class* label was selected as target value. This analysis is different than the one presented in our work.

3. Materials and Methods

This section describes the main characteristics and properties of the selected dataset as well as the nature of the machine and deep learning techniques that were applied. Each model is briefly explained and all configurations and parameters are detailed. The employed evaluation metrics are also carefully explained so that the comparison between methods can be better understood.

3.1. Dataset

The CIDDS-001 (Coburg Intrusion Detection Data Set), disclosed by Markus Ring *et al.* in [8], contains about four weeks of network traffic from two different environments, an emulated small business environment (OpenStack) and an External Server that captured real and up-to-date traffic from the internet. The OpenStack environment includes several clients and typical servers like an E-Mail server or a Web server. The dataset contains labeled flow-based data that can be used to evaluate anomaly-based network intrusion detection systems considering normal activity as well as DoS, Brute Force, Ping Scans and Port Scan attacks. The python scripts used for traffic generation can be found in a github repository [21].

The collection of data provided by the CIDDS-001 dataset is represented in an unidirectional Netflow format. Netflow is a feature of CISCO routers that allows the collection of IP network traffic as it enters or exits an interface. Table 1 provides an overview of the dataset attributes. All attributes from 1 to 12 are default Netflow features whereas those from 13 to 16 result from the labelling process.

Table 1. Specification of the CIDDS-001 dataset features. Adapted from [8].

Nº	Name	Desription
1	Src IP	Source IP Address
2	Src Port	Source Port
3	Dest IP	Destination IP Address
4	Dest Port	Destination Port
5	Proto	Transport Protocol (e.g. ICMP, TCP, or UDP)
6	Date first seen	Start time flow first seen
7	Duration	Duration of the flow
8	Bytes	Number of transmitted bytes
9	Packets	Number of transmitted packets
10	Flags	OR concatenation of all TCP Flags
11	Tos	Type of Service
12	Flows	Not specified
13	Class	Class label (Normal, Attacker, Victim, Suspicious and Unknown)
14	AttackType	Type of Attack (PortScan, DoS, Bruteforce, PingScan)
15	AttackID	Unique Attack id. Allows attacks which belong to the same class carry the same attack id
16	AttackDescription	Provides additional information about the set attack parameters (e.g. the number of attempted password guesses for SSH-Brute-Force attacks)

Regarding the *AttackType* label, CIDDS-001 is a very unbalanced and realistic dataset. The majority class corresponds to benign traffic. Each of the other classes represents one of four distinct attack types that are not equally distributed over time.

As previously stated, data was captured approximately over the course of four weeks. It started at 5:43:57 p.m., March 3, 2017 up until 11:59:30 p.m., April 18, 2017 comprising a total of nearly 33 million flows. Due to this considerable number of flows it would be necessary high hardware requirements to conduct this study. In order to solve this problem and substantially decrease the required amount of memory, processing power and time only a portion of data was used.

Since one of the objectives of this research is to better understand the effect of temporal dependencies in intelligent attack detection and classification, a random and stratified split approach could not be used because it would not preserve the flow sequence. Efforts were made to find a smaller flow interval with similar class distribution. Only the first two weeks of OpenStack environment were considered and a sample of 2535456 flows between 2:18:05 p.m., March 17, 2017 and 5:42:17 p.m., March 20, 2017 was selected. Table 2 establishes a comparison in terms of size and class proportion between the chosen sample, the first two weeks of OpenStack environment and the whole dataset.

Table 2. AttackType class distribution comparison.

Class	Dataset	OpenStack 1stHalf	Sample
Total Records	32630424(100%)	18762253(100%)	2535456(100%)
Brute Force	9888(00.03%)	4992(00.03%)	1262(00.05%)
DoS	2959027(09.07%)	2959027(15.77%)	390440(15.40%)
No Attack	29352063(89.95%)	15526226(82.75%)	2092550(82.53%)
Ping Scan	6090(00.02%)	6090(00.03%)	1068(00.04%)
Port Scan	303356(00.93%)	265918(01.41%)	50136(01.98%)

In this study, only the introduced sample was used and three labels were omitted: *Class*, *AttackID* and *AttackDescription*. Only *AttackType* is considered because this research is focused on the evaluation of different machine and deep learning approaches for attack recognition and specification.

Date first seen feature was used to index the data in order to preserve the flow sequence and *Flows* column was removed because it had the same value for each dataset entry. After these operations there were 7 categorical features and 3 numerical features remaining. The input value of the employed algorithms, is expected to be a numerical matrix so all non-numerical features, such as *Src IP* and *Dst*

IP, had to be encoded into a representative numerical form. Finally, every feature was normalized between 0 and 1 to enhance the performance of the mentioned techniques.

3.2. Models

One machine learning model, Random Forest [19], and two deep learning models, Multi-Layer Perceptron [22] and Long-Short Term Memory [12,17] were employed. These state of the art techniques for intrusion detection systems presented very promising results in previous researches for multiple state of the art datasets, namely for the CICIDS2017 [9] and the UNSW-NB15 [10]. In order to better understand the main differences of each technique a brief description of their nature is provided as well as all the considered parameters and configurations.

Random Forest Classifier [23]. A Random Forest (RF) consists in a large number of Decision Trees that operate together as an ensemble. Each Decision Tree is a decision tool that works in a tree-like model of decisions and outcomes. For a given dataset entry each tree of a Random Forest model predicts a given class and the most voted one is elected as the model's output. The underlying theory behind Random Forests is the wisdom of crowds. In order to assure good performance the algorithm must be trained with good representative data and the correlation between the predictions of each tree must be low [24]. Figure 1 represents the behaviour of the classifier.

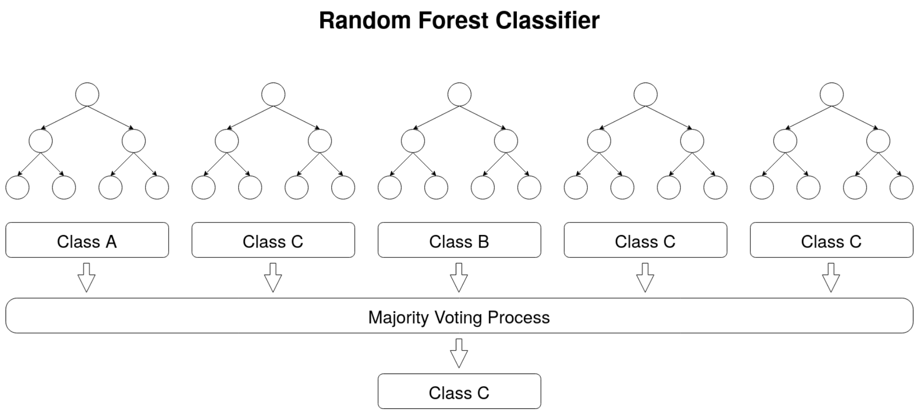


Figure 1. Representation of Random Forest prediction process.

The Random Forest Classifier has multiple parameters that can be configured. Experimental results achieved from this study can possibly be improved by mining the optimal combination of parameters. Max tree dept was set to 35 to prevent the occurrence of overfitting and to ensure generalization. Class weights were adjusted to be inversely proportional to class frequencies in the input data in order to avoid poor classification on the minority classes since the CIDDS-001 dataset is highly unbalanced [19]. Table 3 summarizes some important parameters that were applied.

Table 3. Summary of Random Forest configuration.

Parameter	Value
Nº of Estimators	100
Split Criterion	Gini Impurity
Max Dept	35
Min Samples Split	2
Min Samples Leaf	1
Max Features	$\sqrt{\text{Nº of Features}}$
Class Weights	Balanced

Multi-Layer Perceptron [25]. A Multi-Layer Perceptron (MLP) is a type of Artificial Neural Network (ANN). This computational model, strongly inspired in biological neural networks, learns through exemplification and solves complex problems without the need to be programmed with

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 December 2020 doi:10.20944/preprints202012.0315.v1

task-specific rules. A Feed Forward Network (FNN) is a type of ANN that can be represented as an acyclic graph with no feedback connections (the outputs of the model are not fed back into itself). An MLP is a FNN that comprises three or more layers, having one input layer, one or more hidden layers and an output layer in which each layer has multiple neurons that can be represented in mathematical notation. Each hidden layer h_i can be mathematically described by:

$$h_i(x) = f(w_i^T x + b_i) \tag{1}$$

where $x = x_1, x_2, ..., x_{n-1}, x_n$ is the input vector, w_i is the weights vector, b_i is the bias and f is a non-linear activation function like *sigmoid*, *hyperbolic tangent* or *softmax* (prefered for the output layer). These activation function are mathematically represented as:

$$sigmoid = \frac{1}{1 + e^{-1}} \tag{2}$$

$$tangent = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3}$$

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=i}^{n} e^{x_j}} \tag{4}$$

where x defines a given input. Figure 2 graphically represents a fully connected architecture of a MLP with one input layer, one hidden layer and one output layer.

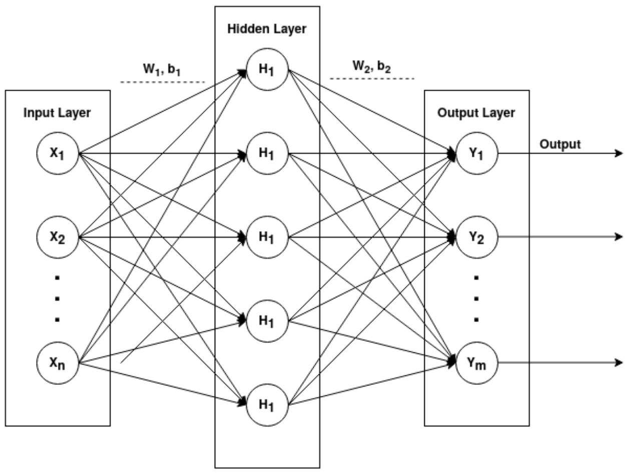


Figure 2. Architecture of a Multi-Layer Perceptron.

The employed MLP consists in a 4-layered architecture. The input layer node number is the same as the number of considered features (10 for single flow classification). The output layer node is 5, the same as the number of different categories for the *AttackType* label. Two hidden layers of 100 neurons each were considered. The Dropout of each layer was set to 0.2 in order to prevent the occurrence of overfitting. Table 4 for describes the model's architecture.

Table 4. Employed MLP architecture.

Layer	Size	Activation	Dropout
Dense	10 × Window	-	-
Dense	100	ReLU	0.2
Dense	100	ReLU	0.2
Dense	5	Softmax	-

Rectified Linear Unit (*ReLU*) was selected as the activation function of the hidden layers. This activation function has proven to be very computationally efficient and it was one of the main breakthroughs in the neural network history for reducing the vanishing and exploding gradient phenomenon [26]. Its mathematical representation is:

$$f(x) = max(0, x) \tag{5}$$

where x defines the input.

Softmax was used for the output layer since it assigns decimal probabilities to the prediction of each class in a multi-classification problem. The sum of all probabilities add up to one. Categorical Cross-Entropy was chosen as loss objective function and Adam was used as optimization function. Table 5 summarizes the main configurations of the MLP model.

Table 5. Summary of MLP configuration.

Parameter	**Value**
Epoch	50
Batch Size	1024
Optimizer	Adam
Learning Rate	0.001
Objective Loss	Categorical Cross-Entropy

Long-Short Term Memory [27]. A Long-Short Term Memory (LSTM) is a type of Recurrent Neural Network (RNN). A RNN contains feedback connections that allow information to travel in loop from layer to layer. These networks store information about past computations through an hidden state that represents the network memory. Therefore the output, o_t, for a given input, x_t, at a given timestep, t, is influenced by the inputs of its previous timesteps, $x_{t-1}, x_{t-2}, ..., x_{t-n}$, where n defines the total number of prior timesteps. This characteristic allows RNN's to be very suited for working with sequential data. Figure 3 represents the unfold form of a standard *many to many* RNN.

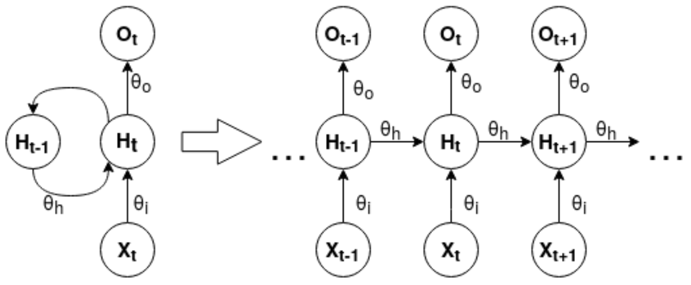


Figure 3. Standard many to many RNN in unfold form (influenced by [28] and [29]).

Unfold is a concept associated with RNN graphical representation. The network is expanded accordingly to the size of its input and output sequence. RNN can be modeled for *one to one*, *one to many*, *many to one* and *many to many* problems. The difference between each of the presented problems is the distinct cardinality between the input and output of the network. A RNN can be mathematically expressed by the following equations [28]:

$$h_t = g(h_{t-1}, x_t; \theta) \tag{6}$$

$$o_t = f(h_t; \theta) \tag{7}$$

where x_t is the input, o_t is the output and h_t is the hidden state at a given timestep t. h_{t-1} is the hidden state of previous timestep and θ comprises the weights and biases of the network.

RNN's main issue is that they have difficulties in learning long term relationships between the elements of the input sequence due to the vanishing and exploding gradient problem. LSTM cells were designed to overcome this problem through the usage of cell memory and gating units. Therefore an input x_t at a given timestep t can change or even override the cell state. This process is carefully regulated by three gating units, the *input gate*, the *forget gate* and the *output gate* [29]. Figure 4 describes the basic architecture of a LSTM cell.

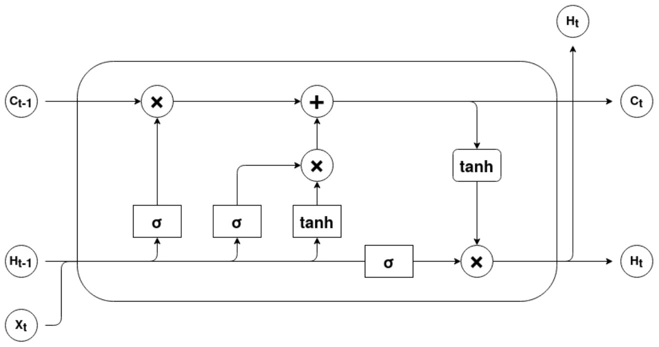


Figure 4. Basic LSTM cell architecture (adapted from [29]).

The *forget gate* is a *sigmoid* layer that outputs a result between 0 and 1 for each value of the cell state C_{t-1}. This results determines which information should be erased or kept. It can be mathematically represented as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{8}$$

The input gate determines which values should be updated and the *tanh* layer creates a vector of candidate values $\hat{C}_t$. The current cell state, C_t, results from the addition of the forget computation, obtained by the multiplication of C_{t-1} and f_t, with the scaled candidate values. This process can be represented by the following equations:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{9}$$

$$\hat{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{10}$$

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{11}$$

The *output gate* establishes the output information. The *sigmoid* layer determines what cell status information will be output. The cell state C_t is processed by *hyperbolic tangent* in order to scale its values between -1 and 1. Finally the multiplication between these outputs is the LSTM cell output value. The process can be mathematically defined as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{12}$$

$$h_t = o_t * tanh(C_t) \tag{13}$$

The employed LSTM network model has one input layer of shape (1, 10), two hidden layers with 100 nodes each and one output layer with 5 nodes, one for each class. Changes in the length of the flow window implies altering the input shape from (1, 10) to (Window Size, 10). Each hidden layer has

Dropout set to 0.2 to prevent the network to overfit the training data. Table 6 describes the employed LSTM network architecture.

Table 6. Employed LSTM network architecture.

Layer	Size	Activation	Dropout
Dense	(Window, 10)	-	-
Dense	100	Tanh	0.2
Dense	100	Tanh	0.2
Dense	5	Softmax	-

The LSTM network configurations are the same as the ones configured for the MLP. Categorical Cross-Entropy remains as loss objective function and Adam optimizer with default 0.001 learning rate was used. The number of epochs was set to 50 and batch size to 1024.

3.3. Anomaly Detection

In this research anomaly detection is addressed from two distinct viewpoints. The first viewpoint consists on finding patterns in single flow features while the second attempts to make a more informed analysis by considering an entire sequence of flows.

Regarding the single-flow viewpoint, each enumerated technique was trained with the preprocessed data of the CIDDS-001 dataset. RF and MLP expected a 2D matrix as input while LSTM, that was specifically designed for sequential data, requires a 3D input. Figure 5 exemplifies how the two dimensional CIDDS-001 preprocessed data and its corresponding 3D transformation can be visualized.

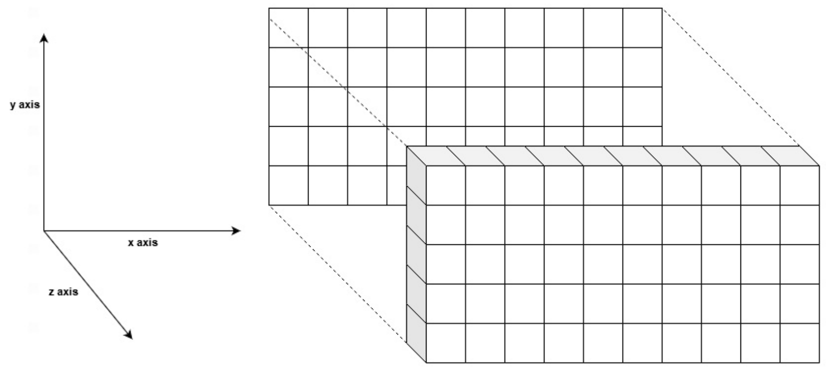


Figure 5. Single flow input visualization. The x axis stands for the number of features of each flow, the y represents the number of samples and the z, required for the LSTM input, corresponds to the number of timesteps (1 timestep for single flow).

The multi-flow approach addresses the problem from a different point of view. Patterns acquired from single-flow features can be insufficient to correctly detect and differentiate anomalies. From a temporal and sequential analysis of a given window of flows is possible to combine individual patterns with short to long term relations between flows that can potentially lead to better results. In order to conduct this analysis data had to be reorganized and prepared to reflect the intended temporal properties. The original 2D data was reshaped to a 3D format and different window sizes of flows were considered. For a selected window size s, a dataset entry e_i can be defined as $e_i = x_{t-s}, ..., x_{t-1}, x_t$ where x is a given flow at a given timestep t. Algorithm 1 describes how the introduced transformation was implemented.

Algorithm 1 Pseudo-code for the data reshape algorithm

```
1:  Input
2:      $D$    2D matrix of original data, where $D = \{x_1, ..., x_{n-1}, x_n\}$
3:      $s$    Window size
4:
5:  Output
6:      R    3D resulting matrix, where $R = \{e_1, ..., e_{n-1}, e_n\}$
7:
8:  procedure TransformToSequential($D, s$)                    ▷ Transforms D into a 3D matrix
9:      $R \leftarrow \{\}$
10:     $l \leftarrow len(D)$
11:     for $i = 0, 1, ..., l-1$ do
12:         if i >= s then                                     ▷ The first $s$ rows should be removed
13:             for $j = 0, 1, ..., s$ do
14:                 $z \leftarrow s - j$
15:                 $R_{i,j} \leftarrow D_{i-z}$    ▷ $R_{i,j} = x_j \wedge x_j \in e_i \wedge e_i \in R$    ▷ $D_{i-z} = x_{i-z} \wedge x_{i-z} \in D$
16:             end for
17:         end if
18:     end for
19:     return $R$
20: end procedure
```

If a given entry e_i contains $s+1$ flows, the first s rows of the algorithm's output matrix should not be considered because they will contain invalid data due to the insufficient number of previous timesteps. For this study the first 99 rows were removed to assure that the algorithms are trained and tested with the same data independently of the selected window size (that is never superior to that value).

As previously explained, LSTM is a model designed to deal with three-dimensional temporal data and already expects a 3D matrix as input while RF and MLP require the conventional 2D input. In order to provide such input, the sequential data was flattened into a two dimensional space. This operation implies a significant loss of information. There is no distinction between the features of independent flows and data although containing information from previous timesteps can't be interpreted as a sequence. This process is represented in Figure 6.

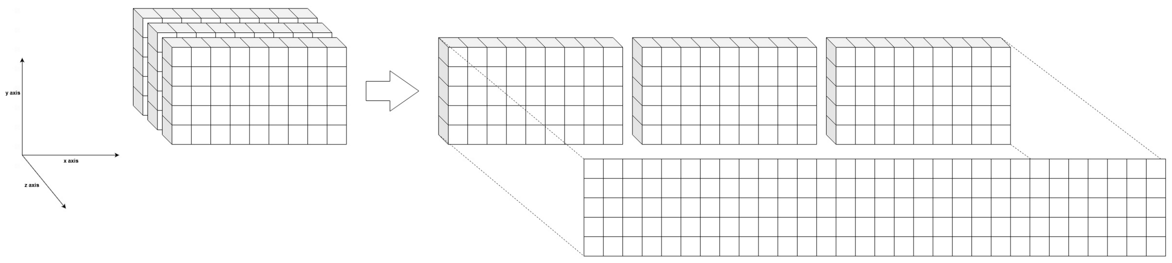


Figure 6. Multi-flow as a two-dimensional space. The x axis stands for the number of features, y represents the number of samples and z describes the number of time steps.

After applying the required transformations, 70% of the total data was selected for training while the remaining 30% were used for testing. From the training set, 10% of data was put aside for validation.

The full scope of our approach is summarized in Figure 7.

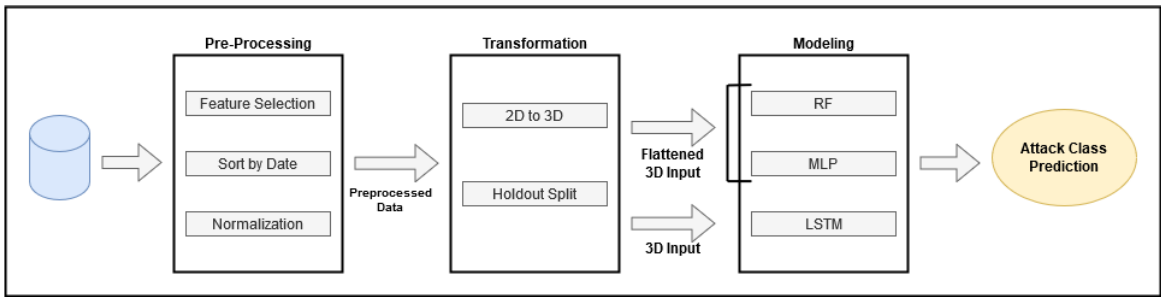


Figure 7. Anomaly Detection Approach.

3.4. Evaluation Metrics

There are several metrics, such as Accuracy, Recall, Precision and F1-Score, that can be used to evaluate the performance of a given classifier [30]. However the employment of these metrics should not be done carelessly, particularly in the context of intrusion detection systems where situations with high class imbalance are fairly common. This section will briefly describe the metrics that were used on this research, their mathematical representation [31] and how they can be interpreted in order to better understand the results obtained from the application of the previously enumerated methods.

Accuracy is one of the most common metrics used in classification problems and usually gives a reliable measure of the model's performance. It can be calculated through the following equation [32]:

$$accuracy(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} I(y_i = \hat{y}_i) \tag{14}$$

where $n_{samples}$ is the number of samples of a given dataset, $\hat{y} = \hat{y}_1, \hat{y}_2, ..., \hat{y}_{n-1}, \hat{y}_n$ is the vector of predicted values, $y = y_1, y_2, ..., y_{n-1}, y_n$ is the corresponding vector of true values and I is the indicator function, which returns 1 if y_i matches $\hat{y}_i$ and 0 otherwise.

In this particular case accuracy is not a good metric to be used due to the fact that CIDDS-001 is a highly unbalanced dataset. In the selected sample, *82.53%* of flows are benign traffic so even if the classifier fails to classify every other attack class and only predicts correctly the benign flows it would still achieve an accuracy value of *82.53%*. Accuracy is biased towards the majority class when working with unbalanced data. In order to overcome this situation other metrics were considered.

Regarding the number of true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN) reported by the confusion matrix, macro-averaged precision can be expressed as:

$$P_{macro} = \frac{1}{n_{classes}} \sum_{i=1}^{n_{classes}} \frac{TP_i}{TP_i + FP_i} \tag{15}$$

Precision measures the amount of labels a model has incorrectly predicted to be positive that were actually negative. TP_i is the number of flows correctly classified as class c_i and FP_i is the amount of flows which true value corresponds to a given class c_j where $j \in [1, n_{classes}] \wedge j \neq i$ that were incorrectly labeled as c_i.

On the other hand, recall expresses the ability of a model to find relevant instances in a dataset. Macro-averaged recall can be defined as:

$$R_{macro} = \frac{1}{n_{classes}} \sum_{i=1}^{n_{classes}} \frac{TP_i}{TP_i + FN_i} \tag{16}$$

where FN_i stands for the number of flows of a class c_i that were incorrecly labeled as a class c_j.

F1-score is a metric that considers both precision and recall through the computation of their harmonic mean. The macro-averaged f1-score is well suited for unbalanced datasets such as CIDDS-001 and it can be mathematically described as:

$$F1_{macro} = 2\frac{P_{macro} \cdot R_{macro}}{P_{macro} + R_{macro}} \tag{17}$$

False Positive Rate (FPR), or Fall-Out, expresses the probability of false alarm and can be determined by the following mathematical statement:

$$FPR_{macro} = \frac{1}{n_{classes}} \sum_{i=1}^{n_{classes}} \frac{FP_i}{FP_i + TN_i} \tag{18}$$

In this research, results are expressed through all the above measures. Understanding each evaluation metric as well as the nature of each employed model is crucial in order to make sense of the obtained results.

4. Results and Discussion

The results presented and discussed in this section derive from the implementation of the previously described models, viewpoints and metrics through the usage of the Python programming language and its appropriate libraries.

- Scikit-learn [32] was used to implement RF and to preprocess data.
- NumPy [33] and Pandas [34] were also used for data preprocessing and manipulation.
- Tensorflow [35] and Keras [36] were used to implement MLP and LSTM.
- Matplotlib [37] was used for result visualization.

The required hardware support was granted by Google Colab [38] that provided free access to virtual machines equipped with GPUs and significant amount of disk space and RAM.

4.1. Single-flow

For the single-flow viewpoint, each model was trained and tested with the the CIDDS-001 preprocessed data. Table 7 summarizes the obtained results.

Table 7. Single-Flow results for the LSTM, RF and MLP models.

		Single-flow (%)			
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**FPR**
LSTM	99.91	98.37	71.40	74.23	00.05
RF	99.90	79.43	95.68	85.04	00.02
MLP	99.92	78.68	73.75	75.79	00.06

Even though there are little differences between the accuracy results of each model, the values for the other metrics are quite different. LSTM presents the highest precision score with little recall while RF presents the highest recall value with considerable less precision. On the other hand, MLP exhibits a minor difference between its precision and recall values. This balance translates into a greater f1-score value than the one related to the LSTM model. The f1-score for the RF model is substantially greater than the ones presented by every other technique. Regarding the FPR, the RF assures lesser occurrence of false alarms than both the LSTM and MLP.

The combination between high accuracy values and differentiated f1-scores can lead to the conclusion that models performed quite differently on the minority classes. Figure 8 expresses the f1-score values of each model for every class.

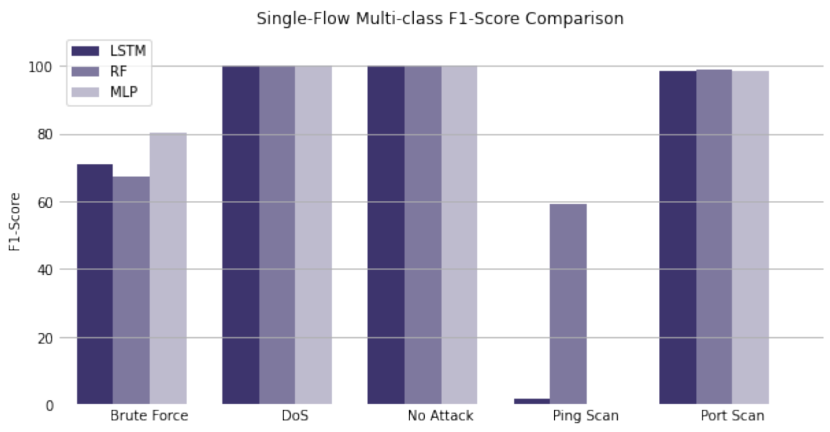


Figure 8. Single-Flow f1-score values of the LSTM, RF and MLP models for the Brute Force, DoS, No Attack, Ping Scan and Port Scan classes.

The RF performed exceedingly better for the Ping Scan class than every other model. Since Ping Scan class only represents 00.04% of the total sample data, the miss classification of this particular class has little affect on the accuracy value. For the macro-averaged f1-score this does not happen because the miss classification of any class, more or less prevalent in the sample, translates into a heavy penalization in its value.

On the general point of view, choosing the best model of this experiment is not a straightforward process. The RF assures that most of the attacks are detected even though with the occurrence of a considerable amount of false positive errors while the LSTM fails to detect a meaningful quantity of attacks but grants a lower occurrence of such errors. Regarding the balance between precision and recall, RF outperforms the remaining techniques but that is only significant when compliant with the requirements and constraints determined by the deployment situation.

4.2. Multi-flow

For the multi-flow viewpoint, the CIDDS-001 preprocessed data was transformed as multiple sequences and several windows sizes from 0 to 70 were used. Figure 9 describes the evolution of the f1-score for each model as the window size increases.

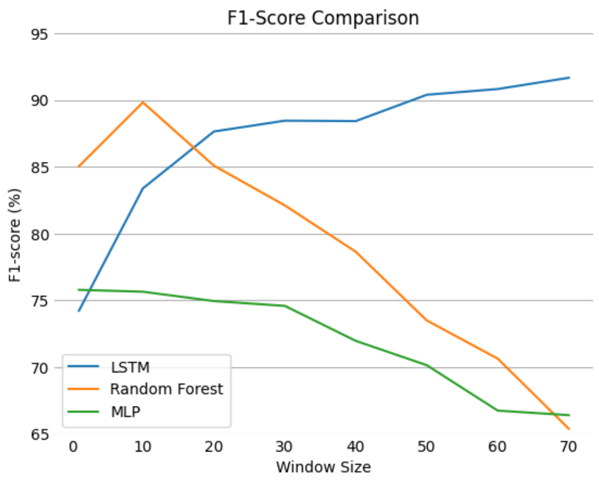


Figure 9. Evolution of the f1-score for the LSTM, RF and MLP models as the window size increase.

The LSTM's f1-score increases alongside the window size while the corresponding score for the remaining models consistently decreases. Despite this trend, the RF model for a window of 10 flows presented a f1-score of 89.82. This result is considerably close to the best recorded value, 91.66, exhibited by the LSTM for the maximum window size.

A possible explanation to the decrease of the f1-score value for the RF model after reaching such a relevant result and the consistent increase of the values presented by the LSTM can be related to the nature of these techniques. While the LSTM model has a cell state carefully regulated by gates that allows it to learn only meaningful short to long term relationships across a given sequence of flows (independently of its size), the RF model was not designed to deal with sequential data. With the increase in window size, the feature space of each entry for the RF substantially increases as well. Its inability to learn sequence based relationships from data can potentially lead to worst results with the increase of feature space. On the other hand, Random Forests generally behave quite well with noise and are not prone to suffer from overffiting. Table 8 compares the scores of the RF and LSTM models that exhibited the best f1-score results.

Table 8. Comparison between the f1-scores of LSTM-70 and RF-10.

			Multi-flow (%)		
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**FPR**
LSTM-70	99.94	94.03	89.71	91.66	00.04
RF-10	99.95	96.83	85.65	89.82	00.04

Both techniques presented significant results with LSTM exhibiting better recall score and slightly less precision. Although the fact that RF's performance becomes worst as the window size increases it is still a quite reliable model for recognising patterns in small sequences of flows. The scores for the LSTM model are more consistent and could potentially be even better for larger window sizes. The value of FPR is the same for both approaches.

Since f1-score is obtained through the harmonic mean of precision and recall, the analysis of these metrics can be important to better understand the model's behaviour. Figure 10 translates the variations in LSTM's precision and recall values.

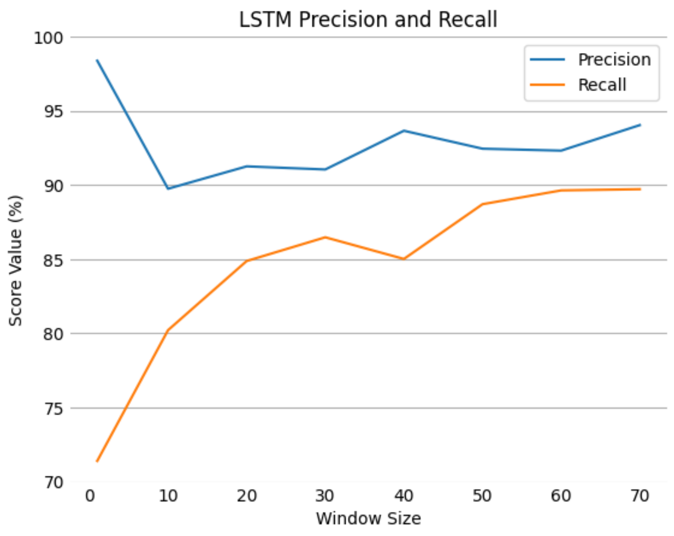


Figure 10. Precision and recall values for the LSTM model as window size increases.

For a window of 10 flows, precision dropped from 98.37 to 89.74 and recall increased from 71.40 to 80.21. This trade-off resulted in a better balanced between the values of each metric which translated into a greater f1-score value. Although casual decreases have occurred for some sequence lengths, both precision and recall got consistently better for further window sizes.

Macro-averaged f1-score can also be detailed by understanding how the individual score for each class evolves. Figure 11 expresses the model's behaviour for its minority classes.

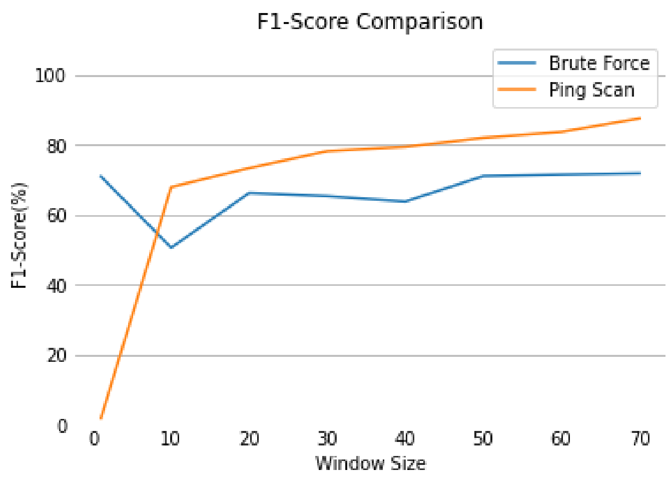
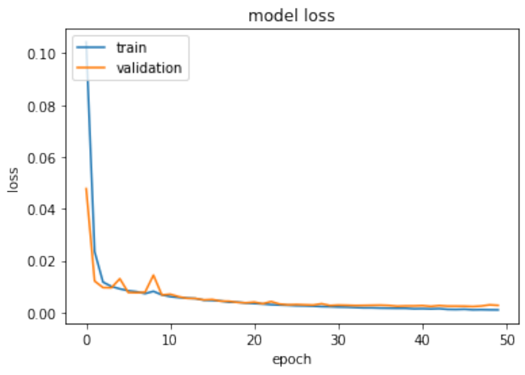


Figure 11. F1-score changes for the Brute Force and Ping Scan.

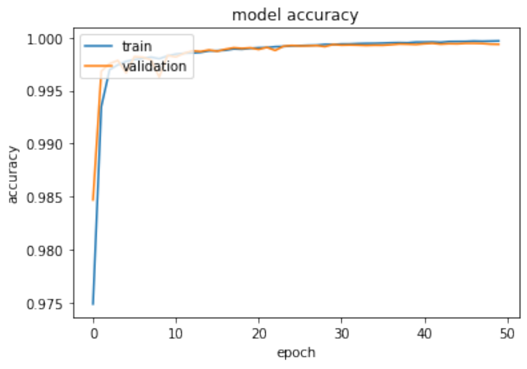
The most prevalent classes were not represented because there were no significant changes in their values. For the DoS class the score remained at 99.99, for attack absence at 99.77 and for Port Scan it increased from 98.48 to 99.12. Regarding the minority classes, despite the considerable drop of Brute Force's f1-score for the 10 flow window, it steadily increased from 70.84 to 71.76 for further sizes. On the other hand, the Ping Scan class presented the most relevant increase, from 01.86 to 87.46.

The presented results can lead to the interpretation that the LSTM is able to detect consistently more attacks with less false positive errors as the flow sequences grow larger. This general improvement does not seem to imply favoring a class performance over another as the f1-score of every class either remained the same or increased.

To ensure that the model has been trained properly, the learning curves, represented in Figure 12, were carefully studied.



(a) Optimization learning curve.



(b) Performance learning curve.

Figure 12. Training and validation learning curves.

The loss value decreases over each epoch for both training and validation sets without any evidence of the occurrence of underffiting or overffiting. The accuracy score for both sets is also quite similar and it slowly improves as loss value gets minimized.

4.3. Conclusion

Overall, it can be concluded that anomaly-based intrusion detection for the CIDDS-001 dataset can be better addressed from a multi-flow perspective. The obtained results lead to the interpretation that an analysis that considers features from individual flows as well as patterns expressed by flow sequences with specific lengths builds a considerably better classification model. The employment of the LSTM model has proven to be very promising for detecting such patterns in data when compared with other state of the art techniques and its performance as a classifier seems to get better as the flow sequence size increases.

5. Conclusions

With the recent technological advances in software, hardware and network topologies, the nature of cyber-attacks is becoming increasingly dynamic and complex. Machine learning models can be a fundamental piece to understand such complex information and to reliably detect the occurrence of a given attack that can compromise sensible information. Over the years, the lack of realistic datasets have been appointed has one of the main difficulties for conducting anomaly detection research. Recently, several datasets were introduced in order to overcome this problem and many researches have obtained quite meaningful results.

The CIDDS-001 is currently one of the most relevant datasets for testing NIDS techniques and although some works have already addressed this dataset, the *AttackType* target variable remained unexplored. This research used that variable as a label in order to teach machine learning methods, such as RF, MLP and LSTM, to correctly identify a given dataset entry as either a benign event or a DoS, Brute Force, Ping Scan or Port Scan attack. These models were implemented from two distinct perspectives and the obtained results were compared in order to determine which was best suited for the dataset nature.

For the single-flow perspective, only individual flow characteristics were addressed by the models and RF achieved the best result with a f1-score of 85.04%. On the other hand, for the multi-flow viewpoint, both individual flow features and short to long term relationships between flows of a given sequence were considered. Several sequence sizes were tested and the best f1-score value, 91.66%, was obtained by the LSTM model for a window size of 70. Although the performance of the RF considerably drops with the increase in sequence length, for a window size of 10, it achieved a f1-score of 89.82% that is relatively close to the best recorded value.

From the results it can be concluded that learning sequential relationships between flows seem to considerably improve anomaly detection. The LSTM as proven to be very reliable model for capturing these sequential patterns and its performance appears to get better for bigger flow sequences.

This research was very important to conclude that, in the scope of the SATIE project, LSTM are an interesting approach to deal with the temporal complexity of security alerts with multiple heterogeneous sources. This technique will be part of a Machine Learning module to be integrated in an investigation tool, where security operators can better grasp the context of incident occurrence.

In the future, it would be pertinent to train the LSTM for larger windows in order to better understand the model's behaviour. More refined feature engineering and data preparation methods can be performed for improving the results and other LSTM architectures with attention mechanisms can further be experimented.

Author Contributions: Conceptualization, I.P., N.O. and E.M.; methodology, I.P. and N.O.; implementation, N.O.; validation, I.P. and O.S.; investigation, E.M. and N.O.; writing, I.P. and N.O.; supervision, I.P. and O.S.; project administration, I.P.; funding acquisition, I.P. All authors have read and agreed to the published version of the manuscript.

Funding: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 832969.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Debar, H.; Dacier, M.; Wespi, A. Towards a taxonomy of intrusion-detection systems. *Computer Networks* **1999**, *31*, 805 – 822.

2. Rashid, A.; Siddique, M.J.; Ahmed, S.M. Machine and Deep Learning Based Comparative Analysis Using Hybrid Approaches for Intrusion Detection System. 2020 3rd International Conference on Advancements in Computational Sciences (ICACS), 2020, pp. 1–9.
3. He, H.; Sun, X.; He, H.; Zhao, G.; He, L.; Ren, J. A Novel Multimodal-Sequential Approach Based on Multi-View Features for Network Intrusion Detection. *IEEE Access* **2019**, *7*, 183207–183221.
4. Papamartzivanos, D.; Gómez Mármol, F.; Kambourakis, G. Introducing Deep Learning Self-Adaptive Misuse Network Intrusion Detection Systems. *IEEE Access* **2019**, *7*, 13546–13560.
5. Zhengbing, H.; Zhitang, L.; Junqi, W. A Novel Network Intrusion Detection System (NIDS) Based on Signatures Search of Data Mining. First International Workshop on Knowledge Discovery and Data Mining (WKDD 2008), 2008, pp. 10–16.
6. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access* **2019**, *7*, 41525–41550.
7. Małowidzki, M.; Berezinski, P.; Mazur, M. Network Intrusion Detection: Half a Kingdom for a Good Dataset. Proceedings of the NATO STO SAS-139 workshop, 2015.
8. Ring, M.; Wunderlich, S.; Gruedl, D.; Landes, D.; Hotho, A. Flow based benchmark data sets for intrusion detection. In *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS) , to appear*; ACPI, 2017.
9. Sharafaldin, I.; Habibi Lashkari, A.; Ghorbani, A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. ICISSP, 2018, pp. 108–116.
10. Moustafa, N.; Slay, J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). 2015 Military Communications and Information Systems Conference (MilCIS), 2015, pp. 1–6.
11. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A survey of network-based intrusion detection data sets. *Computers & Security* **2019**, *86*, 147–167.
12. Gwon, H.; Lee, C.; Keum, R.; Choi, H. Network Intrusion Detection based on LSTM and Feature Embedding, 2019.
13. SATIE - Security of Air Transport Infrastructure of Europe. http://satie-h2020.eu/, 2020. Accessed: 2020-09-02.
14. Adhi Tama, B.; Rhee, K.H. Attack Classification Analysis of IoT Network via Deep Learning Approach. *Research Briefs on Information & Communication Technology Evolution (ReBICTE)* **2017**, *3*.
15. Vilela, D.W.F.L.; Ferreira, E.W.T.; Shinoda, A.A.; de Souza Araújo, N.V.; de Oliveira, R.; Nascimento, V.E. A dataset for evaluating intrusion detection systems in IEEE 802.11 wireless networks. 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), 2014, pp. 1–5.
16. Verma, A.; Ranga, V. Statistical analysis of CIDDS-001 dataset for Network Intrusion Detection Systems using Distance-based Machine Learning. *Procedia Computer Science* **2018**, *125*, 709 – 716. The 6th International Conference on Smart Computing and Communications.
17. Althubiti, S.A.; Jones, E.M.; Roy, K. LSTM for Anomaly-Based Network Intrusion Detection. 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), 2018, pp. 1–3.
18. Nicholas, L.; Ooi, S.Y.; Pang, Y.; Hwang, S.; Tan, S.Y. Study of long short-term memory in flow-based network intrusion detection system. *Journal of Intelligent & Fuzzy Systems* **2018**, *35*, 1–11.
19. Abdulhammed, R.; Faezipour, M.; Abuzneid, A.; AbuMallouh, A. Deep and Machine Learning Approaches for Anomaly-Based Intrusion Detection of Imbalanced Network Traffic. *IEEE Sensors Letters* **2019**, *3*, 1–4.
20. Roy, B.; Cheung, H. A Deep Learning Approach for Intrusion Detection in Internet of Things using Bi-Directional Long Short-Term Memory Recurrent Neural Network. 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), 2018, pp. 1–6.
21. Ring, M.; Wunderlich, S.; Gruedl, D.; Landes, D.; Hotho, A. Generation scripts for the coburg intrusion detection data sets (cidds). https://github.com/markusring/CIDDS, 2017. Accessed: 2020-05-11.
22. Zhiqiang, L.; Mohi-Ud-Din, G.; Bing, L.; Jianchao, L.; Ye, Z.; Zhijun, L. Modeling Network Intrusion Detection System Using Feed-Forward Neural Network Using UNSW-NB15 Dataset. 2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE), 2019, pp. 299–303.
23. Breiman, L. Random forests. *Machine learning* **2001**, *45*, 5–32.
24. Yiu, T. Understanding Random Forest. https://towardsdatascience.com/understanding-random-forest-58381e0602d2, 2019. Accessed: 2020-05-15.

25. Kain, N.K. Understanding of Multilayer perceptron (MLP). https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f/, 2018. Accessed: 2020-05-26.
26. Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics; Gordon, G.; Dunson, D.; Dudík, M., Eds.; PMLR: Fort Lauderdale, FL, USA, 2011; Vol. 15, *Proceedings of Machine Learning Research*, pp. 315–323.
27. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Compututation* **1997**, *9*, 1735–1780.
28. McGonagle, J.; Williams, C.; Khim, J. Recurrent Neural Network. https://brilliant.org/wiki/recurrent-neural-network/. Accessed: 2020-05-12.
29. Olah, C. Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. Accessed: 2020-05-14.
30. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management* **2009**, *45*, 427–437.
31. Döring, M. Performance Measures for Multi-Class Problems. https://www.datascienceblog.net/post/machine-learning/performance-measures-multi-class-problems/, 2018. Accessed: 2020-05-15.
32. Cournapeau, D. Scikit-learn Documentation. https://scikit-learn.org/. Accessed: 2020-05-15.
33. Oliphant, T. NumPy Documentation. https://numpy.org/index.html, 2020. Accessed: 2020-05-19.
34. McKinney, W. Pandas Documentation. https://pandas.pydata.org/. Accessed: 2020-05-16.
35. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D.G.; Steiner, B.; Tucker, P.; Vasudevan, V.; Warden, P.; Wicke, M.; Yu, Y.; Zheng, X. TensorFlow: A System for Large-Scale Machine Learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16); USENIX Association: Savannah, GA, 2016; pp. 265–283.
36. Chollet, F. Keras Documentation. https://keras.io/. Accessed: 2020-05-20.
37. Hunter, J.D. Matplotlib Documentation. https://matplotlib.org/, 2020. Accessed: 2020-05-19.
38. Bisong, E., Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*; Apress: Berkeley, CA, 2019; pp. 59–64.