electronics

MDPI

Article

A Composite Approach of Intrusion Detection Systems: Hybrid RNN and Correlation-Based Feature Optimization

Sunil Gautam [1], Azriel Henry [2], Mohd Zuhair [1], Mamoon Rashid [3,*], Abdul Rehman Javed [4,5] and Praveen Kumar Reddy Maddikunta [6,*]

1 Department of Computer Science and Engineering, Institute of Technology, Nirma University Ahmedabad, Ahmedabad 382481, India
2 Department of Computer Science and Engineering, Institute of Advanced Research, Gandhinagar 382007, India
3 Department of Computer Engineering, Faculty of Science and Technology, Vishwakarma University, Pune 411048, India
4 Department of Cyber Security, Air University, Islamabad 44000, Pakistan
5 Department of Electrical and Computer Engineering, Lebanese American University, Byblos 1102 2801, Lebanon
6 School of Information Technology and Engineering, Vellore Institute of Technology, Vellore 632014, India
* Correspondence: mamoon.rashid@vupune.ac.in (M.R); praveenkumarreddy@vit.ac.in (P.K.R.M.)

Abstract: Detection of intrusions is a system that is competent in detecting cyber-attacks and network anomalies. A variety of strategies have been developed for IDS so far. However, there are factors that they lack in performance, creating scope for further research. The current trend shows that the Deep Learning (DL) technique has been proven better than traditional techniques for IDS. Throughout these studies, we presented a hybrid model that is a Deep Learning method called Bidirectional Recurrent Neural Network using Long Short-Term Memory and Gated Recurrent Unit. Through simulations on the public dataset CICIDS2017, we have shown the model's effectiveness. It has been noted that the suggested model successfully predicted most of the network attacks with 99.13% classification accuracy. The proposed model outperformed the Naïve Bayes classifier in terms of prediction accuracy and False Positive rate. The suggested model managed to perform well with only 58% attributes of the dataset compared to other existing classifiers. Moreover, this study also demonstrates the performance of LSTM and GRU with RNN independently.

Keywords: intrusion detection system; deep learning; Recurrent Neural Network; CICIDS-2017 dataset

check for updates

Citation: Gautam, S.; Henry, A.; Zuhair, M.; Rashid, M.; Javed, A.R.; Maddikunta, P.K.R. A Composite Approach of Intrusion Detection Systems: Hybrid RNN and Correlation-Based Feature Optimization. *Electronics* **2022**, *11*, 3529. https://doi.org/10.3390/electronics11213529

Academic Editor: Esteban Tlelo-Cuautle

Received: 17 August 2022
Accepted: 26 October 2022
Published: 29 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

An instrument called an intrusion detection system (IDS) is effective in spotting possible cyber attacks. It uses algorithms to classify and detect the attacks. There are two categories of Intrusion Detection Systems, namely Signature-based IDS (Sig-IDS) and Anomaly-based IDS (Anom-IDS). Sig-IDS detects attacks based on previously known patterns, sequences, or a set of rules defined for the attacks. Meanwhile, Anom-IDS detects anything different than normal traffic, i.e., anomalies. This is the improvement of Anom-IDS over Sig-IDS—it can detect novel attacks in the system. Moreover, in terms of data sources, Host-based IDS (HIDS) and Network-based IDS (NIDS) are two forms of IDS. HIDS can detect attacks from inside the system by inspecting the data from the operating system, firewall logs, and database logs or audits on apps. NIDS can detect external attacks before it enters the network of computers. NIDS monitors the traffic extracted from different network data sources to detect any threats to the network [1,2].

There are different types of approaches for developing Sig-IDS and Anom-IDS. As shown in Figure 1, knowledge-based or statistical-based methods and deep machine learning are the Anom-IDS-built approaches. Statistical metrics such as the median, mode, mean,

and standard deviation are used for the statistical-based approach. Models for the same are Univariate, Multivariate, and Time-series models. For a knowledge-based approach, a set of rules is created using human knowledge. Finite State Machines, Description languages, and Expert Systems are examples of Knowledge-based IDS. The machine learning approaches can be categorized into supervised, semi-supervised, and unsupervised learning. Supervised learning uses labeled input data for training, whereas unsupervised learning uses unlabeled input data for training. Semi-supervised learning uses some labeled data and a vast number of unlabeled input data for training [3–5].

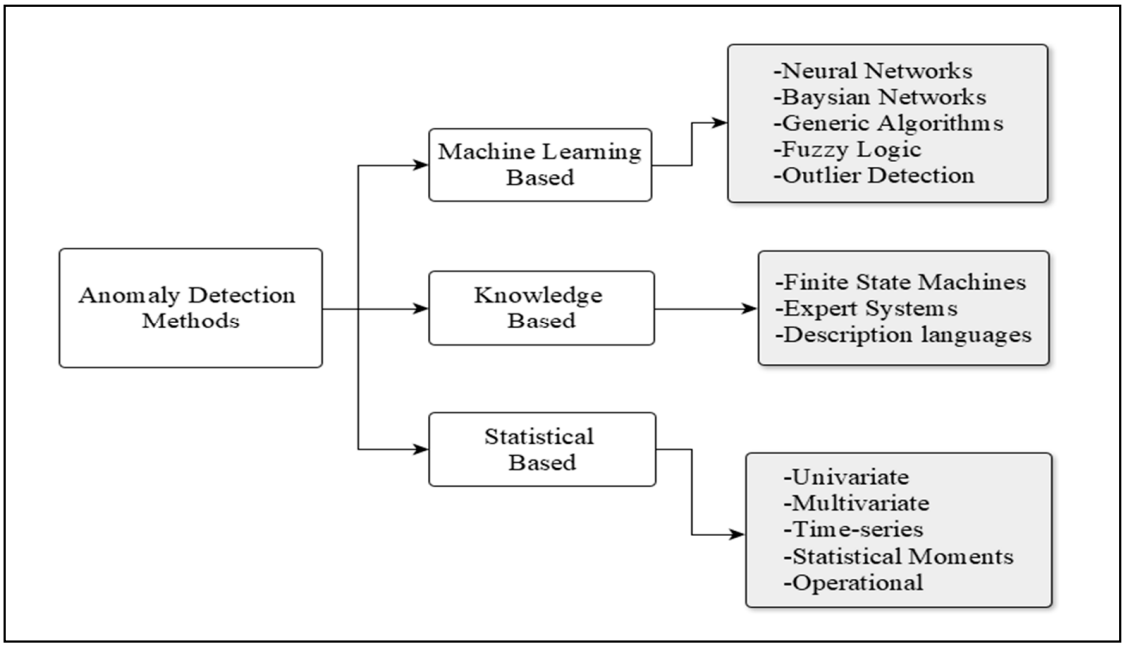


Figure 1. Classification of Anomaly-based IDS.

Deep learning provides a scope to develop flexible IDSs. Some examples of deep learning algorithms include Deep Neural Network (DNN), Deep Boltzmann Machine (DBM), Deep Belief Network (DBN), Restricted Boltzmann Machine (RBM), Recurrent Neural Network (RNN), etc., which are used to generate labels for unlabeled data [6,7].

The dataset plays a key role when using knowledge strategies. Some datasets include the deep learning and machine learning models KDD 99 dataset, ICSI Enterprise Tracing Project, Australian Centre for Cyber Security published dataset-UNSW-NB15, CAIDA dataset, DARPA Lincoln Lap Packet trace, CIDDS-001, or CICIDS 2017. CICIDS 2017 contains a broad series of common attacks as well as benign ones [8,9].

Therefore, to develop a robust IDS, we have used a deep learning algorithm-RNN. Unlike other deep learning techniques, RNN has the ability to generate cycles of edges to form self-feeding of a neuron's connections to itself over time. RNN introduces a time-step edge, which makes it different from a feed-forward Neural Network. However, it has a longer dependencies issue, which can be addressed using LSTM or GRU. The longer dependencies issue or long-term dependency arises when exponentially smaller weights are given to the long-term interactions. Hence, it becomes difficult for the model to track past information. LSTM works as a memory cell or a block to overcome the dependencies issue. GRU is the improved version of the LSTM framework with lesser gates. The RNN-LSTM architecture is widely used in many applications [10,11]. In this study, we have also tried different combinations of frameworks with the RNN to have the best possible results. Moreover, selecting appropriate features for the technique is an important part. Therefore, we have used a filter-based feature optimization technique to optimize the dataset and enhance the performance of the model.

The order of the remaining text is as follows. Section 2 examines recent research on this subject. Section 3 is the preliminary part, which discusses the components such as RNN, LSTM, GRU, and Dataset. Section 4 presents the proposed model, which is followed by Section 5, where the performance of the model is analyzed using various parameters. Section 6 concludes the work of this study.

2. Related Work

Recent time shows an increase in the development of Learning-based IDS compared to other methods. Different developments in IDS are shown in this section, along with their description and key points.

Muhuri et al. [12] proposed a model to develop IDS. Their approach is divided into two stages. First, they used a GA (Genetic Algorithm) for feature optimization, and secondly, for classification, they used a framework of deep learning called RNN. To enhance the effectiveness of RNN, they added the sequence of the LSTM unit. The NSL-KDD dataset was used to evaluate the performance of their model. Their outcomes show that the utilization of GA increases the precise categorization for binary and multiclass classification. Moreover, the accuracy of their recommended model is more advanced than the Support Vector Machine and Random Forest for multiclass classification.

Shurman et al. [13] showcased the deep learning model RNN with the Long Short-Term Memory (LSTM) framework for attack detection for Distributed Denial of Service (DDoS) and Denial of Service (DoS) attacks. They proposed two LSTM models, i.e., two LSTM layers and three LSTM layers. They claim that the three-layer LSTM model achieves higher accuracy than other models used.

Belavagi et al. [14] have shown the use of algorithms for supervised machine learning to implement an IDS; specifically, four classification algorithms, Logistic Regression, Support Vector Machine, Gaussian Naive Bayes, and Random Forest, are used. These algorithms are evaluated using parameters such as True Positive Rate (TPR), False Positive Rate (FPR), Precision, Recall, F1-Score, and accurateness. After testing the models on the NSL-KDD dataset, they found that Random Forest performs better than its counterparts.

Gaikwad et al. have suggested an approach to implement IDS using the Machine learning technique known as Bagging-Ensemble. The ensemble method combines the outputs or predictions of the estimators used. They used a primary classifier partial decision tree. Moreover, they reduced features from 41 to 15 of the NSL-KDD datasets using the Genetic Algorithm. The classification accuracy of their model was 99.7166% when evaluated using a cross-validation method. However, the classification accuracy of the test dataset was less than that of the C4.5 classifier [15].

Ahmad et al. [16] introduced the implementation of the ELM (Extreme Learning Machine) to create a system for detecting intrusions. ELM is a single or multiple feed-forward neural network with a hidden layer that learns faster and achieves higher generalization proficiency than other feed-forward neural networks. Moreover, it can perform better with complex datasets. They compared ELM with other categorization techniques, such as SVM and Random Forest. To evaluate these models, they divided the NSL-KDD dataset into the following samples: Full Samples, Half Samples, and $\frac{1}{4}$ Samples. They have reported that ELM outperforms SVM and RF in Precision, Recall, and Accuracy to complete a set of data. However, the SVM classifier performed better on half data samples and $\frac{1}{4}$ of the data samples.

Kasongo et al. [17] designed a Deep Learning (DL) technique called FFDNNs to implement the IDS. Unlike Artificial Neural Networks (ANNs), DNNs may contain hundreds of hidden layers. NSL-KDD is the dataset utilized in this study. They fused their model with Feature Extraction Unit (FEU) to reduce the input dimension. They performed classification with binary as well as multiclass datasets. Their model was in comparison to features such as SVM, Decision Tree, and KNN. They found that FFDNNs outperformed other classifiers for both binary and multiclass problems. Moreover, with full and FEU-reduced feature space, it achieved superior performance.

Yang et al. [18] mentioned a DL method based on Improved Convolution NN called (ICNN)-based wireless network Intrusion Detection Model (IBWNIDM) by implementing Wireless IDS. The ICNN consists of five convolutional layers, four completely integrated layers, two pooling layers, and one SoftMax layer. In this study, the NSL-KDD dataset was used to evaluate the performance of their model. The model training of ICNN includes forward propagation and backward propagation process. IBWNIDM was compared with other models, namely LeNet-5, DBN, and RNN. They reported that their suggested model's proposed detection rate was more than that of LeNet-5 and DBN models but lower than RNN.

Yins [19] have introduced a DL method RNN to implement an IDS. The essential difference between RNNs and other FFNNs is the ability of RNNs to memorize the previous information and use it in the current layer. The suggested design was evaluated on the NSL-KDDCUP dataset, which shows that RNN has a high rate of consistency and a relatively low false-positive rate compared to other conventional methods, including Naïve Bayesian and J48.

Kim et al. [20] presented an AI-based architecture using optimal CNN with LSTM and SFL character encoding in normalized UTF-8 on real-time data. UTF-8 allows feature extraction without including unnecessary encryption, compression, entropy calculations, etc. They measured the performance of their model using parameters such as f-score, accuracy, precision, specificity, and recall by performing investigations on two open datasets, namely CSIC-2010 HTTP and CICIDS-2017 Http datasets. They reported that AI-based IDS can distinguish between regular and abnormal HTTP traffic and could not be using signature-based IDS. Moreover, they have also claimed that the proposed model can create and enhance Snort rules for IDS that uses signatures.

Wang et al. [21] suggested a technique to interpret with optimizing decisions of models used to build an Intrusion Detection System. They have shown two categories of model interpretability, namely Global and Local interpretability. Global interpretability explains the model by its overall structure, whereas local interpretability examines the input. They used a framework called Shapley Additive Explanations (SHAP), which provides an explanation of the prediction of any instance based on the contribution of the characteristics. The NSL-KDD dataset was used to evaluate their framework. They claimed that the interpretability results of their framework depend on the properties of specific attacks. However, it is not possible to use SHAP in real time.

Hao et al. [22] showcased the use of variant Gated Recurrent Units (GRUs) using encoders for developing an Intrusion detection system. GRU is a gating mechanism used with recurrent neural networks. Two variants, namely encoded GRU and encoded binarized GRU, were used in this study. This study claims that encoded GRU gives a better representation of inputs while encoded binarized GRU, along with the better representation of inputs, also reduces the access time and memory size.

The related works show a variety of techniques to implement IDS. The majority of such proposed works use KDDCUP and the NSL-KDD datasets to assess the performance of their proposed system. However, these datasets do not show much diversity in attributes. In this paper, we have used the CICIDS-2017 dataset, which has a variety of attacks and nearly double the attributes of the KDD datasets. Moreover, this proposed work also focuses on reducing the input dimension, i.e., attributes, using a suitable feature selection method to develop an efficient technique.

3. Preliminaries

3.1. Recurrent Neural Network

Intelligence is highly dependent on past memory. Past memory can be long-term as well as short-term in certain cases. For example, generating words from sound combinations needs short-term memory, whereas predicting the next word in a sentence depends on the context used earlier, needing long-term memory. Long-term dependencies and variable correlation issues can cause the model to work inaccurately. A Recurrent Neural Network

(RNN) provides the ability to resolve these issues. RNN enables the network to add feedback on the outputs of the time step before the current time step. The basic advantage of using RNN is that it provides memory cells to function based on short-term and long-term memories [23,24].

Figure 2 displays the basic architectural style of the RNN along with the unfolding structure of recurrent layers. Inputs for the RNN refer to the raw or structured data, such as audio, video, tabular data, etc. The Recurrent layer has 'x' inputs, 'h' as a hidden layer, 'y' as an output layer, and 'w' as weights. Representing the input layer, hidden layer, and output layer at time-step 't'. Output 'y' can be calculated as follows.

$$a^t = b_1 + Wh^{(t-1)} + Ux^t \tag{1}$$

$$h^t = A\left(a^{(t)}\right)$$

$$y^t = b_2 + Vh^{(t)}$$

where 'b_1' and 'b_2' are bias vectors and 'W','U', and 'V' are the weights of layer connections, namely hidden–hidden, input–hidden, and hidden–output, respectively, and 'A' is an Activation function. Sigmoid, relu, softmax, etc., are some of the activation functions used in the recurrent layers.

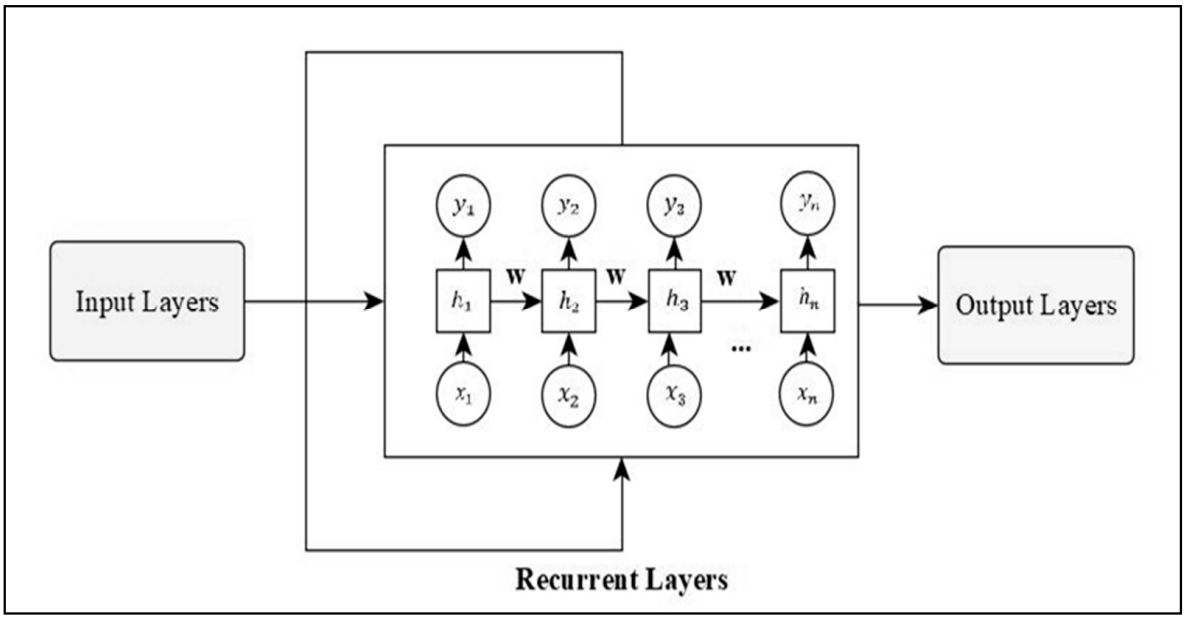


Figure 2. Architecture of RNN.

Moreover, there are different types of recurrent layers used with RNN, such as LSTM, GRU, Convolution LSTM (ConvLSTM), and Quasi-RNN. Among these recurrent layers, LSTM is the most popular and widely used in different applications. GRU is a variant of LSTM, which combines input and forget gates into a single gate. The LSTM layer is discussed in detail in the next section. ConvLSTM replaces all matrix-to-vector multiplications with 2D convolutions, which is better when the input data contains spatial relations. Quasi-RNN is designed in a way to make computation and parallelization easier in the recurrent layer. It removes the previous time-step and previous time-step states from all matrix-to-vector multiplications. In addition, Simple Recurrent Unit (SRU), LSTM with Projection Layer, and LSTM with Peepholes are some of the other variants of the recurrent layer [25,26].

3.2. Long Short-Term Memory (LSTM)

RNNs face difficulties such as exploding or vanishing gradients when learning long-term dependencies. LSTM is specially designed to address such difficulties. A well-composed basic structure of LSTM is called vanilla LSTM, i.e., a combination of a cell, an input gate, and an output gate. Later, another gate called forget gate was introduced by Gers et al., which developed the ability to clear memory blocks once their information becomes useless. The primary distinction between RNN and LSTM is the addition of the three gates in LSTM [27,28].

Figure 3 shows the standard structure and working of the LSTM block; here, the input gate governs the size of the novel information or knowledge to the cell state, the forget gate determines what information is unhelpful and needs to be deleted from the cell state, and finally, the output gate decides what info requirements to be given as output. Simple multiplication and addition in the long-term memory C of LSTM allow the information to flow smoothly and thereby effectively address the gradient dispersion problem. Sigmoid and tanh are the two mainly used activation functions in LSTM. Equations (2) and (3) elaborate on these functions as follows [29].

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{2}$$

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3}$$

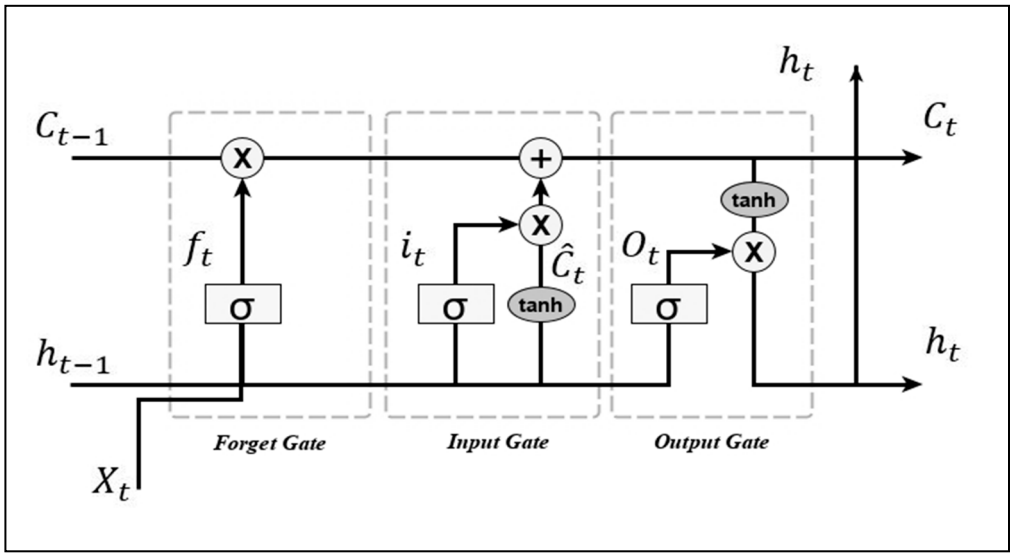


Figure 3. Architecture of LSTM.

The blocks in the LSTM structure from left to right, as depicted in Figure 3, are the input gate, forget gate, and output gate. The working of these gates is elaborated in the set of equations as follows.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{4}$$

Here, 'f_t' represents the forgetting factor, 'x_t' is the state of the input, and 'h_{t-1}' denotes the state of the short-term memory transferred from the previous LSTM unit. Whereas 'W_f' and 'b_f' are the weight and the bias of the forgetting factor. The output of this gate decides the information that has to be discarded.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{5}$$

$$\hat{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{6}$$

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{7}$$

Here, 'C_t' and '$\hat{C}_t$' are the newly generated long-term memory cell and the alternative vector required for the updating process, respectively. '*' denotes the multiplication operation. Finally, the output gate is as follows.

$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{8}$$

$$h_t = O_t * \tanh(C_t) \tag{9}$$

Here, 'O_t' is a representation of the output gate, whereas the notation 'h_t' denotes the final status of short-term memory information. The weights in Equations (4)–(6) and Equation (8) are updated using backpropagation through time (BPTT) based on the difference between actual and output values. Hence, using this highly flexible gate structure, LSTM can combat the vanishing and exploding gradient problem of RNN [30].

3.3. Gated Recurrent Unit

Another framework designed to combat the exploding or vanishing gradient problem is the Gated Recurrent Unit (GRU), i.e., the improved version of the LSTM framework discussed in the previous section. GRUs also have a gate structure (like LSTM) for controlling the flow of information. However, GRU does not have an output gate (like LSTM), which results in the full exposure of the content. GRU has no cell state and only two gates, namely, the reset gate and update gate. The update gate is the combination of the input gate and forgets gate of the LSTM framework. GRU has a simple structure and fewer parameters compared to LSTM, which increases its performance. Figure 4 shows the structure of the GRU framework.

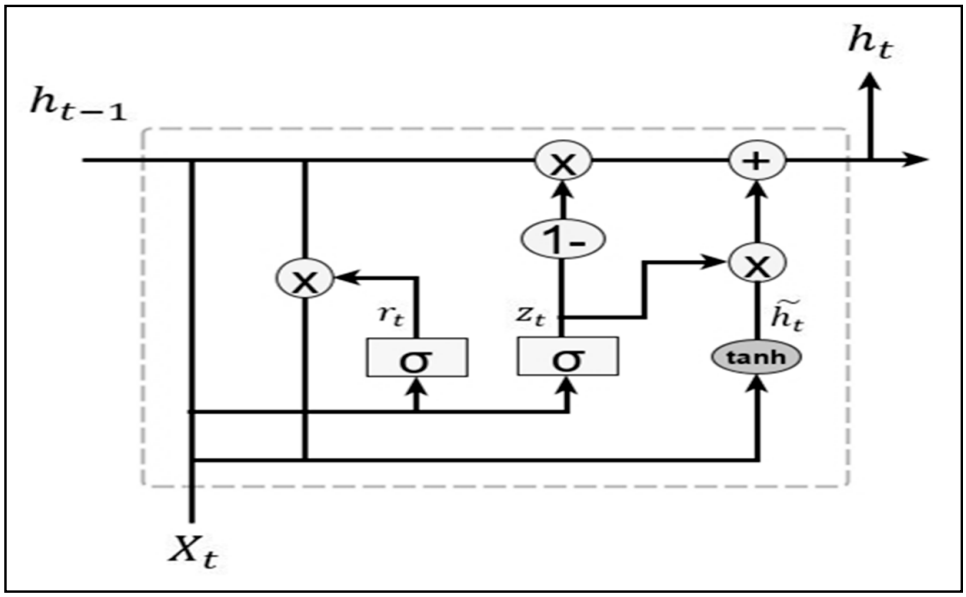


Figure 4. Architecture of GRU.

The formulation of GRU is given by the following equations.

$$1 \tag{10}$$

$$z_t = sigm(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{11}$$

$$\tilde{h_t} = tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \tag{12}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h_t}) \tag{13}$$

where 'r_t', z_t, 'x_t', and 'h_t' are the reset gate, update gate, input vector, and output vector, respectively, '$\widetilde{h_t}$', is the and candidate activation vector, while operator '$\odot$' denotes the Hadamard product. Like LSTM, 'b' denotes the biases and 'W' denotes the weight matrices. While '*sigm*' and '*tanh*' are the activation functions of sigmoid and tangent, respectively. LSTM and GRU are both capable of handling the longer dependencies. However, there are some differences in terms of performance. In this study, we have used both frameworks to analyze their performance in classifying network traffic [31,32].

3.4. Dataset

There are various datasets available for IDS, such as the DARPA dataset, KDD CUP 99, revised KDD Cup dataset (NSL-KDD), AWID dataset, Canadian Institute of Cyber security IDS dataset (CICIDS), etc. To develop a robust IDS, the dataset needs to be properly designed considering factors such as traffic diversity, volume, and variety of attacks. In this study, we used the CICIDS-2017 dataset to examine the performance of our model. The CICIDS-2017 dataset was developed by a team of researchers at the Canadian Institute of Cybersecurity. The CICIDS-2017 dataset reflects the current trends and covers a variety of features and metadata compared to some older datasets. It consists of the results from eight sessions designed to collect data using victim and attack networks. The first session result contains only normal traffic data, i.e., benign samples. The dataset includes seven families of common attacks that meet real-time traffic criteria. There are 77 features in each subset of the CICIDS-2017 dataset [33,34]. In this study, we removed the redundant data from the dataset. Table 1 provides information about the dataset.

Table 1. Data distribution in CICIDS-2017 dataset.

Sub-Dataset	Type of Traffic	Number of Instances	
		With Redundancy	Without Redundancy
Tuesday Samples	BENIGN FTP_Patator SSH_Patator	445,909	425,240
Wednesday Samples	BENIGN DoS_Goldeneye DoS_Hulk DoS_Slowhttptest DoS_slowloris Heartbleed	692,703	613,287
Thursday Morning Samples	BENIGN Web Attack_Brute Force Web Attack-Sql Injection Web Attack_XSS	170,366	164,300
Thurs. Afternoon Samples	BENIGN Infiltration	288,602	254,625
Friday Morning Samples	BENIGN Bot	191,033	184,145
Friday Afternoon Samples-DDoS	BENIGN DDoS	225,745	223,666
Friday Afternoon Samples-PortScan	BENIGN PortScan	286,467	214,114

4. Propose Model

This section manifests the key components of our proposed work, namely the Feature Selection method and the Bidirectional RNN model. Figure 5 shows the complete procedure of the work in three stages, namely data pre-processing, classification, and comparison. Data pre-processing includes the removal of redundancy and selecting the optimum feature set. The

second stage consists of the classification of the data and the calculation of various parameters. Finally, the third stage is the comparative results of the proposed model with other recent algorithms. These three stages are subsequently discussed in Sections 4.1, 4.2 and 5.2.

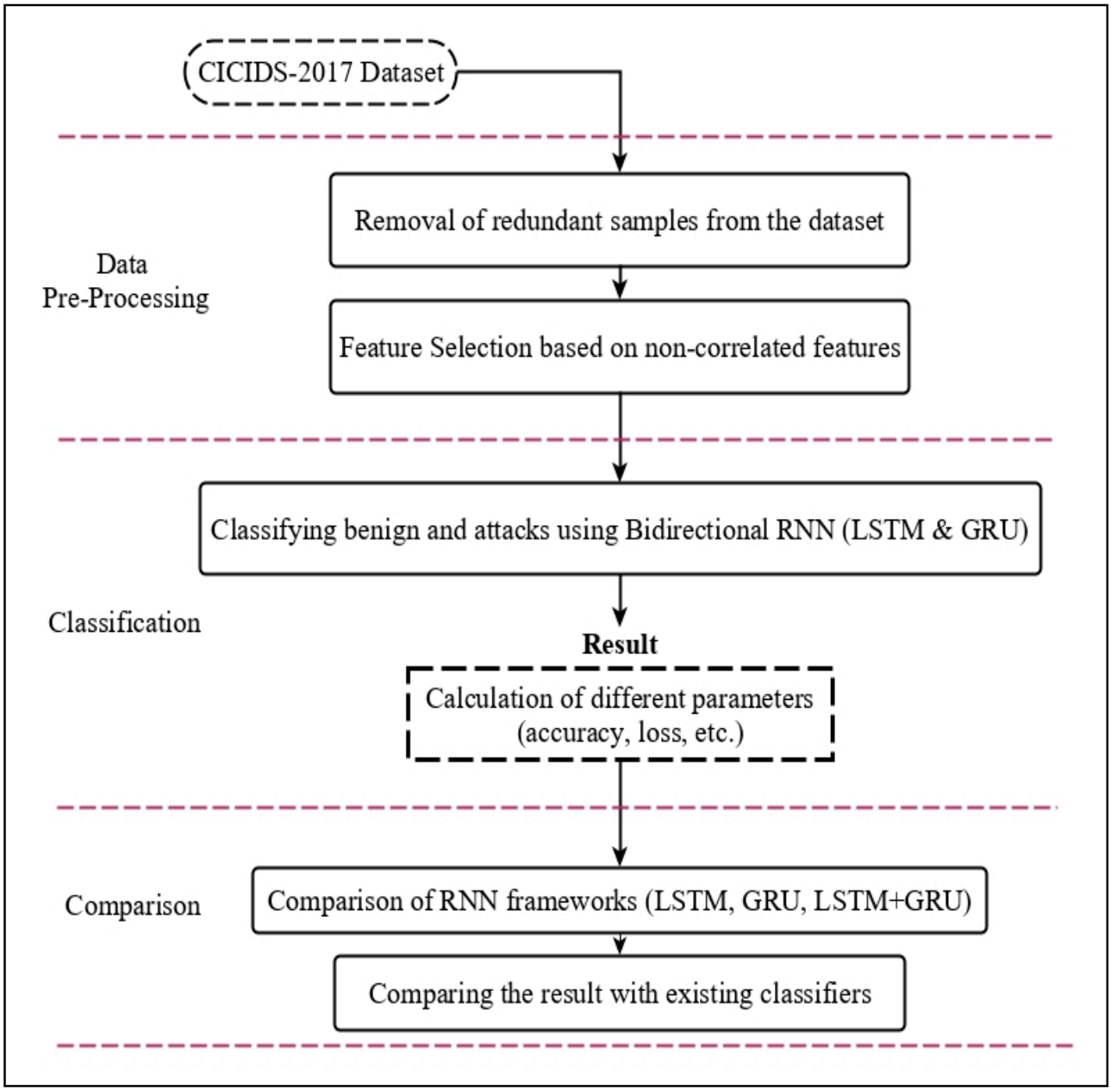


Figure 5. Proposed model layout.

4.1. Selection of Features

This research used a selection of feature methods to reduce the input dimension by selecting an optimum feature subset. The filter method, called Pearson's Correlation Coefficient, is used to find the discriminative attributes from the feature set. It measures the degree of linear correlation of the features in the dataset and delivers a correlation coefficient value that ranges from -1 to 1. When the correlation factor is 1, it denotes a strong correlation, whereas when it is equal to -1, it denotes the weakest correlation. This implies that the higher the correlation coefficient, the more strongly the features

are connected and vice versa. The equation of Pearson's Correlation Coefficient is as follows [35,36].

$$= \frac{E((X - \mu_X)(Y - \mu_Y))}{\sigma_X \sigma_Y} \tag{14}$$

$$= \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}\sqrt{B(Y^2) - E^2(Y)}} \tag{15}$$

where '$\mathrm{cov}(X, Y)$'is a covariance between 'X' and 'Y', 'σ_X' and 'σ_Y'are the standard deviation of X and Y, respectively, and '$E(X)$' is the expected value of 'E'.

Table 2 displays feature subsets of the dataset calculated using the equation of the Pearson Correlation Coefficient. Each subset consists of around 40 features selected from a total of 77 features of the original dataset listed in Table 2. Moreover, these features are selected on the reduced set of samples having no redundancy or duplicate instances. The original dataset is pre-processed, thereby reducing the complexity and making it efficient when used with the proposed model.

Table 2. Dataset features.

Feature	Feature Name	Feature	Feature Name
F_1	Destination_Port	F_40	Max_Packet_Length
F_2	Flow_Duration	F_41	Packet_Length_Mean
F_3	Total_Fwd_Packets	F_42	Packet_Length_Std
F_4	Total_Backward_Packets	F_43	Packet_Length_Variance
F_5	Total_Length_of_Fwd_Packets	F_44	FIN_Flag_Count
F_6	Total_Length_of_Bwd_Packets	F_45	SYN_Flag_Count
F_7	Fwd_Packet_Length_Max	F_46	RST_Flag_Count
F_8	Fwd_Packet_Length_Min	F_47	PSH_Flag_Count
F_9	Fwd_Packet_Length_Mean	F_48	ACK_Flag_Count
F_10	Fwd_Packet_Length_Std	F_49	URG_Flag_Count
F_11	Bwd_Packet_Length_Max	F_50	CWE_Flag_Count
F_12	Bwd_Packet_Length_Min	F_51	ECE_Flag_Count
F_13	Bwd_Packet_Length_Mean	F_52	Down/Up_Ratio
F_14	Bwd_Packet_Length_Std	F_53	Average_Packet_Size
F_15	Flow_Bytes/s	F_54	Avg_Fwd_Segment_Size
F_16	Flow_Packets/s	F_55	Avg_Bwd_Segment_Size
F_17	Flow_IAT_Mean	F_56	Fwd_Avg_Bytes/Bulk
F_18	Flow_IAT_Std	F_57	Fwd_Avg_Packets/Bulk
F_19	Flow_IAT_Max	F_58	Fwd_Avg_Bulk Rate
F_20	Flow_IAT_Min	F_59	Bwd_Avg_Bytes/Bulk
F_21	Fwd_IAT_Total	F_60	Bwd_Avg_Packets/Bulk
F_22	Fwd_IAT_Mean	F_61	Bwd_Avg_Bulk_Rate
F_23	Fwd_IAT_Std	F_62	Subflow_Fwd_Packets
F_24	Fwd_IAT_Max	F_63	Subflow_Fwd_Bytes
F_25	Fwd_IAT_Min	F_64	Subflow_Bwd_Packets
F_26	Bwd_IAT_Total	F_65	Subflow_Bwd_Bytes
F_27	Bwd _IAT_Mean	F_66	Init_Win_bytes_forward
F_28	Bwd_IAT_Std	F_67	Init_Win_bytes_backward
F_29	Bwd_IAT_Max	F_68	act_data_pkt_fwd
F_30	Bwd_IAT_Min	F_69	min_seg_size_forward
F_31	Fwd_PSH_Flags	F_70	Active_Mean
F_32	Bwd_PSH_Flags	F_71	Active_Std
F_33	Fwd_URG_Flags	F_72	Active_Max
F_34	Bwd_URG_Flags	F_73	Active_Min
F_35	Fwd_Header_Length	F_74	Idle_Mean
F_36	Bwd_Header_Length	F_75	Idle_Std
F_37	Fwd_Packets/s	F_76	Idle_Max
F_38	Bwd_Packets/s	F_77	Idle_Min
F_39	Min_Packet_Length		

4.2. Bidirectional RNN

In this study, instead of using a single directional neural network layer, we have used a bidirectional RNN approach. In bidirectional RNN, there are two layers piled high on top of one another. Table 3 displays selected features. A single layer moves in the forward direction, while the other layer moves in the backward direction. The layer from behind does not interact with the forward layer and is solely attached to the output layer. The output of both layers is then combined using operations such as concatenate, sum, multiply, etc. This bidirectional structure allows the networks to have both backward and forward information about the sequence at every time step. In our study, we have used one LSTM sequence for a forward pass and one GRU sequence for a backward pass. Initially, we tested the sequences in the same direction for both LSTM and GRU sequences. However, the performance for the bidirectional outperformed the initial tests; as a result, we have considered the bidirectional approach. The performance of the bidirectional structure is better because it passes the instructions in both directions, which allows the model to have as much information as possible. Thus, the model performs more accurately compared to the other structures. We have combined the outputs of both sequences using the concatenate operation. Figure 5 shows the structure of a bidirectional RNN [37,38].

Table 3. Selected Features.

Sub-Dataset	Features	Total
Tuesday	{'F1', 'F2', 'F3', 'F5', 'F7', 'F8', 'F9', 'F11', 'F12', 'F15', 'F16', 'F17', 'F18', 'F20', 'F23', 'F31', 'F32', 'F33', 'F34', 'F35', 'F36', 'F38', 'F39', 'F44', 'F46', 'F47', 'F48', 'F49', 'F50', 'F52', 'F56', 'F57', 'F58', 'F59', 'F60', 'F61', 'F66', 'F67', 'F68', 'F69', 'F70', 'F71', 'F75'}	43
Wednesday	{'F1', 'F2', 'F3', 'F5', 'F7', 'F8', 'F11', 'F12', 'F15', 'F16', 'F17', 'F20', 'F25', 'F26', 'F27', 'F28', 'F31', 'F32', 'F33', 'F34', 'F38', 'F39', 'F44', 'F46', 'F47', 'F48', 'F49', 'F50', 'F52', 'F56', 'F57', 'F58', 'F59', 'F60', 'F61', 'F66', 'F67', 'F69', 'F70', 'F71', 'F75'}	41
Thursday Morning	{'F1', 'F2', 'F3', 'F7', 'F8', 'F9', 'F11', 'F12', 'F15', 'F16', 'F17', 'F18', 'F20', 'F23', 'F31', 'F32', 'F33', 'F34', 'F38', 'F39', 'F44', 'F46', 'F47', 'F48', 'F49', 'F50', 'F52', 'F56', 'F57', 'F58', 'F59', 'F60', 'F61', 'F66', 'F67', 'F69', 'F70', 'F71', 'F75'}	39
Thursday Afternoon	{'F1', 'F2', 'F3', 'F5', 'F7', 'F8', 'F9', 'F11', 'F12', 'F15', 'F16', 'F17', 'F18', 'F20', 'F23', 'F31', 'F32', 'F33', 'F34', 'F38', 'F39', 'F44', 'F46', 'F47', 'F48', 'F49', 'F52', 'F56', 'F57', 'F58', 'F59', 'F60', 'F61', 'F66', 'F67', 'F68', 'F69', 'F70', 'F71', 'F75'}	40
Friday Morning	{'F1', 'F2', 'F3', 'F7', 'F8', 'F11', 'F12', 'F15', 'F16', 'F17', 'F20', 'F23', 'F31', 'F32', 'F33', 'F34', 'F38', 'F39', 'F44', 'F46', 'F47', 'F48', 'F49', 'F50', 'F52', 'F56', 'F57', 'F58', 'F59', 'F60', 'F61', 'F66', 'F67', 'F69', 'F70', 'F71', 'F75'}	37
Friday Afternoon-DDoS	{'F1', 'F2', 'F3', 'F5', 'F8', 'F11', 'F12', 'F15', 'F16', 'F17', 'F20', 'F25', 'F26', 'F27', 'F31', 'F32', 'F33', 'F34', 'F38', 'F39', 'F44', 'F46', 'F47', 'F48', 'F49', 'F50', 'F52', 'F56', 'F57', 'F58', 'F59', 'F60', 'F61', 'F66', 'F67', 'F69', 'F70', 'F71', 'F75'}	39
Friday Afternoon-PortScan	{'F1', 'F2', 'F3', 'F5', 'F7', 'F8', 'F11', 'F12', 'F15', 'F16', 'F17', 'F20', 'F23', 'F31', 'F32', 'F33', 'F34', 'F38', 'F44', 'F46', 'F47', 'F48', 'F49', 'F50', 'F52', 'F56', 'F57', 'F58', 'F59', 'F60', 'F61', 'F66', 'F67', 'F69', 'F70', 'F71', 'F75'}	37

The Algorithm 1 starts with the data pre-processing step which includes reading the data, separating the target class from the feature set, and separating the dataset into two fragments, namely testing and training samples. The second step states the removal of duplicate or unwanted samples from the data. Then, correlated features are recognized using Pearson's Correlation Coefficient and dropped from the original feature set. Finally,

benign activity and attacks are classified using the proposed model. The performance assessment is carried out using different parameters discussed in Section 5.1.

Algorithm 1 Proposed Algorithm Bidirectional RNN

Input: data instances
Output:

1. data pre-processing
2. reduction of duplicate samples from the dataset
3. computing correlation of the features set $\boldsymbol{C}$ using Equation (10)
4. creating Correlated Feature Set C_f
5. if correlation value > 0.8
6. add feature to C_f
7. else
8. increment in original feature set $\boldsymbol{C}$
9. return C_f
10. training Bidirectional RNN model with training samples
11. applying model to testing samples
12. return Confusion Matrix C_{m*m}
13. calculate Accuracy, Precision, Recall, TPR, and FPR

5. Experiments and Results

5.1. Measures of Performance

To assess the effectiveness of the recommended design, we have applied several performance metrics for each sub-dataset. The final result of the model is the confusion matrix. A confusion matrix is a useful metric to analyze how precisely the model identifies the instances of different labels. Figure 6 shows the diagram of the confusion matrix. Precision and Recall are the most common metrics used to evaluate deep learning models. Precision is the ratio of correct or incorrect learning predictions, whereas Recall is the proportion of the percentage of positive matches found to the total number of positive matches. These metrics are formulated as follows [39–41].

$$Precision = \frac{TP}{(TP + FP)} \tag{16}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{17}$$

$$TPR = \frac{TP}{(TP + FN)} \tag{18}$$

$$FPR = \frac{FP}{(FP + FN)} \tag{19}$$

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{20}$$

where FN, TN, TP, and FP are False Negative, True Negative, True Positive, and False Positive.

TPR is the percentage of accurate instance matches among actual correct instances. FPR is the proportion of incorrect matches among the non-target instances. For example, the perfect model will produce a TPR of 100% and an FPR of 0%. Equations (18) and (19) are the equations of TPR and FPR, respectively. Finally, efficiency is used to measure the effectiveness of the proposed model. Accuracy is the proportion of instances correctly classified by the model and is calculated as in Equation (15) [42–45].

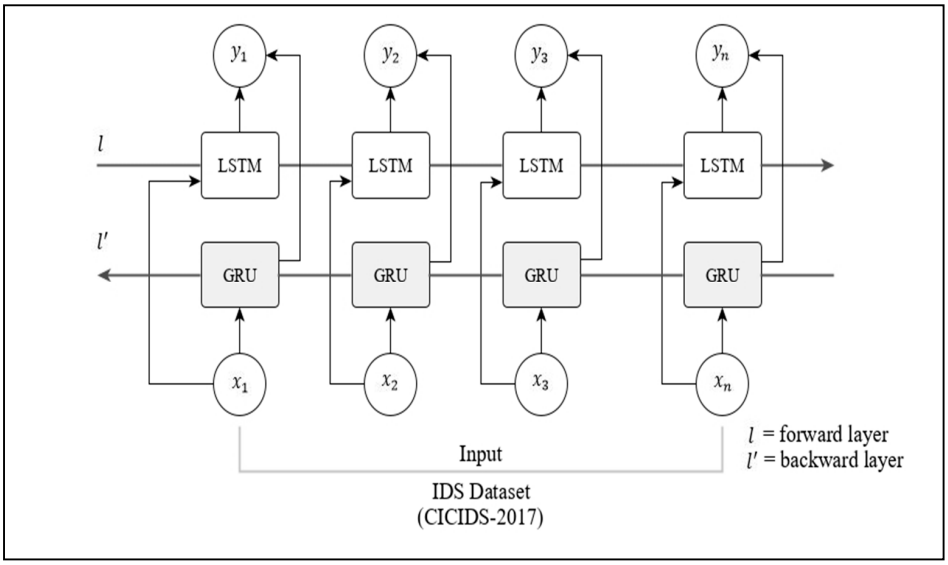


Figure 6. Bidirectional RNN architecture.

5.2. Performance Analysis

The experiments were performed with the Python language in the Google Colab Cloud Environment. The outcome of the tests is the confusion matrix of datasets. Figures 7–13 present the confusion matrix of each sub-datasets of the CICIDS 2017 dataset. Tables 4–10 shows various parameters derived from the confusion matrix, as discussed in the section. The CICIDS 2017 dataset was split into 67% for training and 33% for testing.

		Predicted Class	
		Class A	**Class B**
Actual Class	**Class A**	TP	FP
	Class B	FN	TN
Total instances = A +B		Total A = TP+FN	Total B = TN+FP

Figure 7. Confusion matrix.

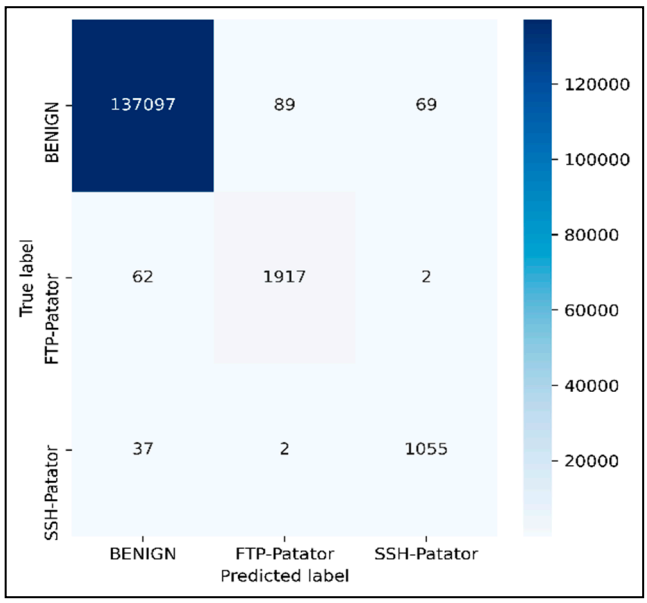


Figure 8. Tuesday confusion matrix data set.

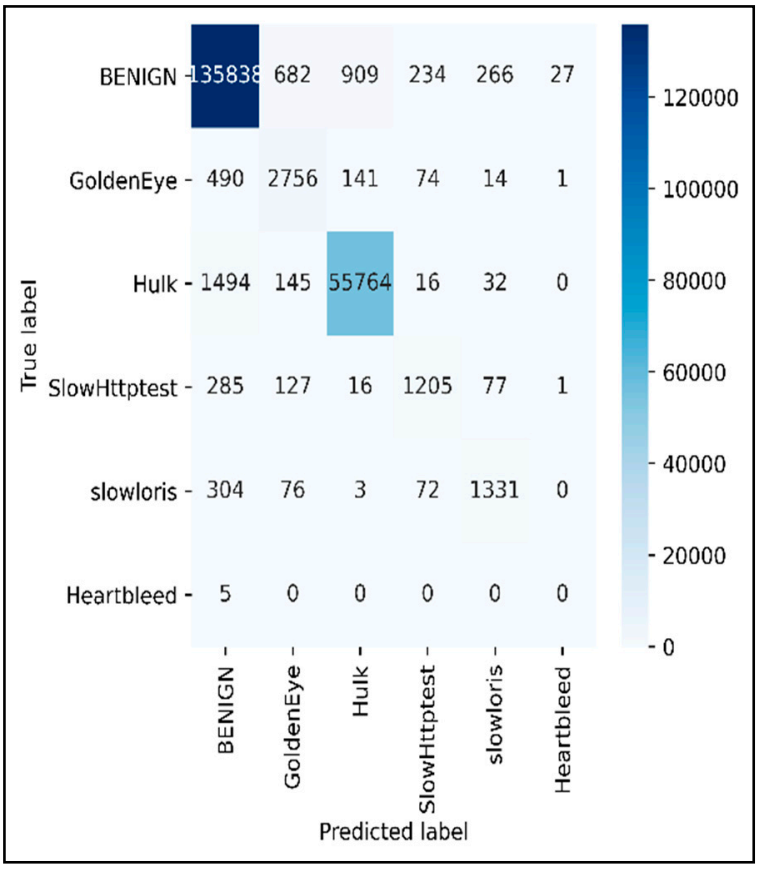


Figure 9. Wednesday confusion matrix data set.

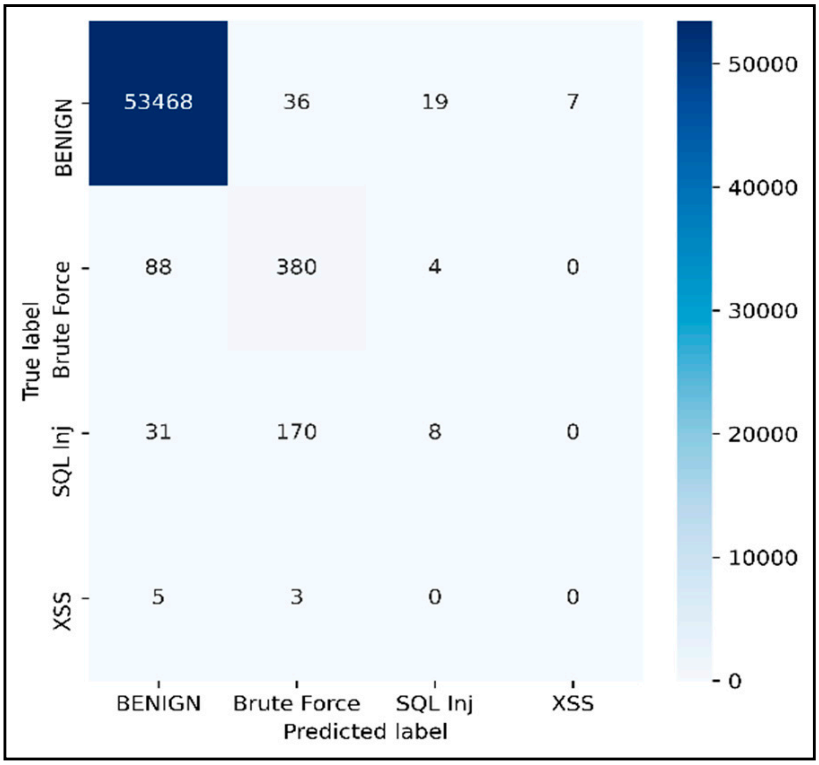


Figure 10. Thursday morning confusion matrix.

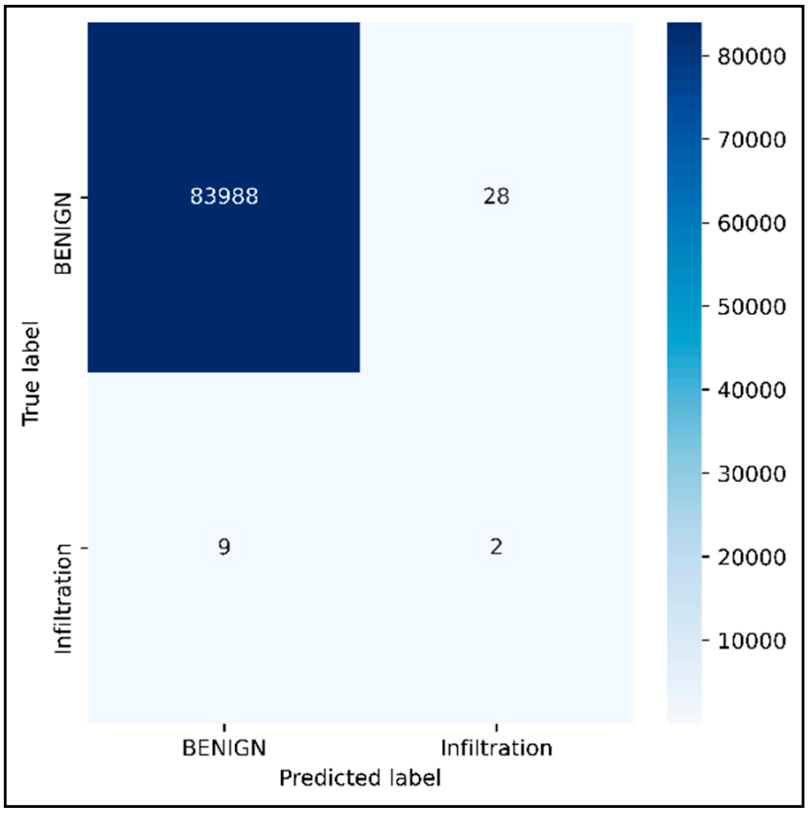


Figure 11. Thursday afternoon confusion matrix.

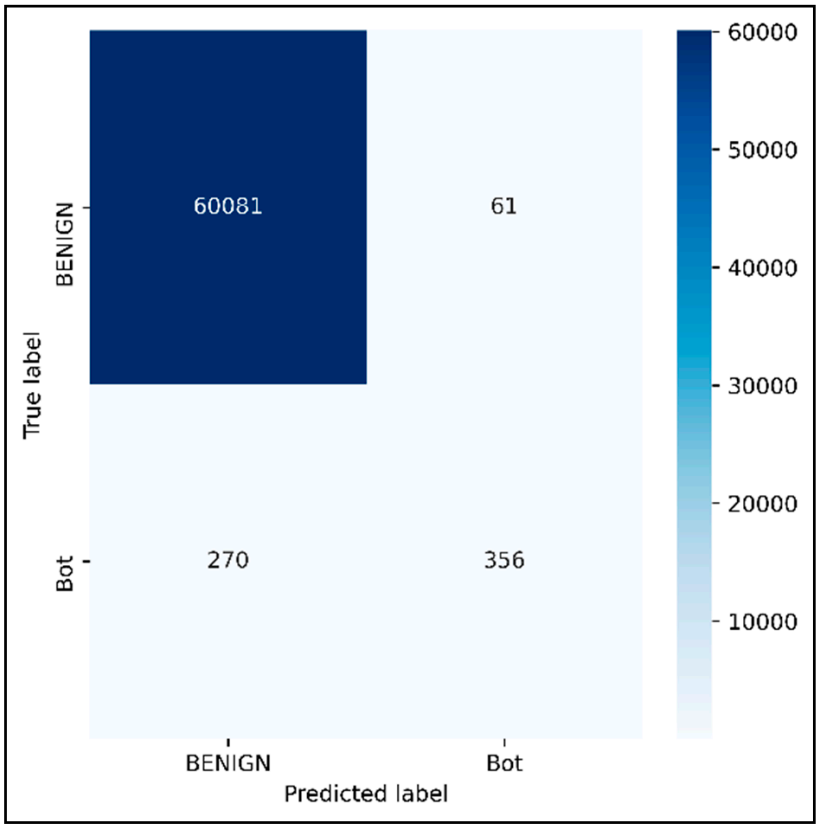


Figure 12. Friday morning confusion matrix.

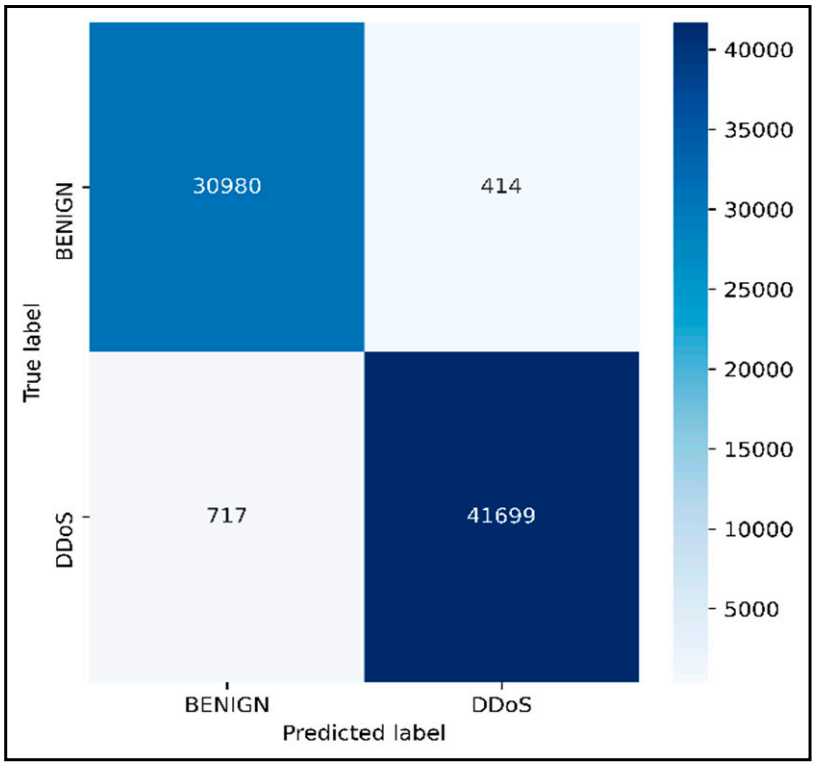


Figure 13. Friday afternoon-DDoS confusion matrix.

Table 4. Statistical report of Tuesday test dataset.

Parameters	BENIGN	FTP-Patator	SSH-Patator
TP	137,097	1917	1055
TN	2976	138,258	139,165
FP	99	91	71
FN	158	64	39
TPR	0.9988	0.9676	0.9643
FPR	0.0321	0.0006	0.0005
Precision	0.9992	0.9546	0.9369
Recall	1.00	0.97	0.96

Table 5. Statistical report of Wednesday test dataset.

Parameters	BENIGN	Goldeneye	Hulk	Slowhttptest	Slowloris	Heartbleed
TP	135,838	2756	55,764	1205	1331	0
TN	61,851	197,879	143,865	200,278	200,210	202,351
FP	2578	1030	1069	396	389	29
FN	2118	720	1687	506	455	5
TPR	0.9846	0.7928	0.9706	0.7042	0.7452	0
FPR	0.0400	0.0051	0.0073	0.0019	0.0019	0.0001
Precision	0.9813	0.7279	0.9811	0.7526	0.7738	0
Recall	0.98	0.79	0.97	0.70	0.75	0

Table 6. Statistical report of Thursday morning test dataset.

Parameters	BENIGN	Brute Force	Sql Inj	XSS
TP	53,468	380	8	0
TN	565	53,538	53,987	54,204
FP	124	209	23	7
FN	62	92	201	8
TPR	0.9988	0.8050	0.0382	0
FPR	0.179	0.0038	0.0004	0.0001
Precision	0.9976	0.6451	0.2580	0
Recall	1.00	0.81	0.04	0.00

Table 7. Statistical report of Thursday afternoon test dataset.

Parameters	BENIGN	Infiltration
TP	83,988	2
TN	2	83,988
FP	9	28
FN	28	9
TPR	0.9996	0.1818
FPR	0.8181	0.0003
Precision	0.9998	0.1818
Recall	1.00	0.18

Table 8. Statistical report of Friday morning test dataset.

Parameters	BENIGN	Bot
TP	60,081	356
TN	356	60,081
FP	270	61
FN	61	270
TPR	0.9989	0.5686
FPR	0.4313	0.0010
Precision	0.9955	0.8537
Recall	1.00	0.57

The CICIDS-2017 dataset contains 77 features, as listed in Table 2, and around two million samples. Removing correlated features and redundant samples from the dataset allowed the model to work more efficiently. The metrics of performance are calculated from the confusion matrix. The rows of the matrix represent the actual samples of classes and the column represents the predicted samples of the classes. The True Positive (TP) metric of the Tuesday dataset is the values located diagonally in the matrix, i.e., (137, 097, 1917, and 1055). A large True Positive shows that the model has a high classification rate, whereas True Negative samples are samples inaccurately classified by the model. Similarly, other metrics are calculated from the confusion matrix using their respective formulas discussed in Section 5.1.

Figure 8 and Table 4 show the statistical parameter of the Tuesday sub-dataset. It shows the prediction of FTP-Patator and SSH-Patator attacks. The findings show that the model achieved a high TP rate and precision value for each class. Figure 9 and Table 5 show

the statistical report of the Wednesday sub-dataset, which is the largest sub-dataset of the original CICIDS 2017 dataset. It also has the maximum number of attacks compared to the other sub-datasets. The attacks predicted were GoldenEye, Hulk, Slowhttptest, slowloris, and Heartbleed. The Hulk attack showed the highest True Positive rate, while the model did not correctly predict any Heartbleed attack samples. Figure 10 and Table 6 show the statistical data of the Thursday morning sub-dataset. It shows the prediction of Brute Force, SQL Injection, and XSS attacks. The Brute Force attack was more accurately predicted than SQL Injection and XSS attacks. Figures 11–14 and Tables 7–10 show the statistical parameters of Thursday and Friday sub-datasets. These reports show the prediction of attacks such as Infiltration, Bot, DDoS, and PortScan attacks. The model did well while predicting Bot, DDoS, and PortScan. However, it did not perform well while predicting the Infiltration attack.

Table 9. Statistical report of Friday afternoon-DDoS test dataset.

Parameters	BENIGN	DDoS
TP	30,980	41,699
TN	41,699	30,980
FP	717	414
FN	414	717
TPR	0.9868	0.9830
FPR	0.0169	0.0131
Precision	0.9773	0.9901
Recall	0.99	0.98

Table 10. Statistical report of Friday afternoon PortScan test dataset.

Parameters	BENIGN	PortScan
TP	40,358	30,049
TN	30,049	40,358
FP	83	168
FN	168	83
TPR	0.9958	0.9972
FPR	0.0027	0.0041
Precision	0.9979	0.9944
Recall	1.00	1.00

Table 11 shows the accuracy calculation for all the sub-datasets in the CICIDS 2017 dataset. The accuracy of each sub-dataset was calculated using multiple epoch settings, considering the training and validation data loss. The model successfully achieved more than 97% accuracy for all sub-datasets. The accuracy of the Thursday afternoon sub-dataset was the highest among all other subsets. Furthermore, we have contrasted the suggested model with different machine learning classifiers, such as Random Forest, Naïve Bayes, and K-Nearest Neighbor (KNN); the value of k is 9 in Table 12. The proposed Bidirectional RNN model scored a 99.13% classification accuracy with nearly half (i.e., ≈58%) of the total features of other classifiers. The False Positive rate (FPR) of the proposed technique was 0.074 which is less than the Naïve Bayes and KNN classifiers.

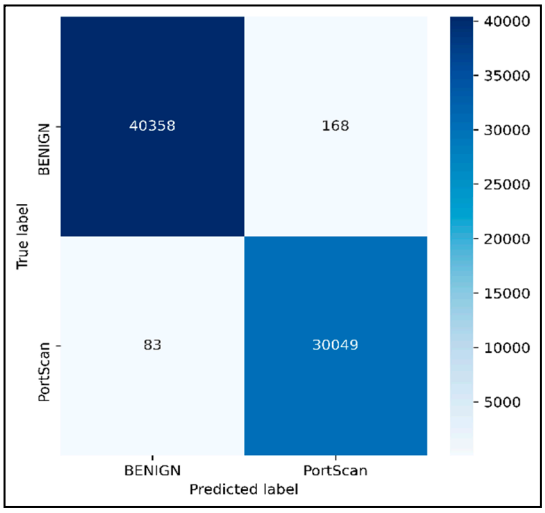


Figure 14. Friday afternoon PortScan confusion matrix.

Table 11. Accuracy.

Sub-Dataset	Accuracy
Tuesday	0.9981
Wednesday	0.9728
Thursday Morning	0.9933
Thursday Afternoon	0.9995
Friday Morning	0.9945
Friday Afternoon-DDoS	0.9846
Friday Afternoon-PortScan	0.9964

Table 12. Performance of models.

Model	No. of Features	Accuracy (%)	FPR	TPR	Precision	Recall
Random Forest	77	99.90	0.017	0.8951	0.9138	0.8957
Naïve Bayes	77	64.49	0.157	0.7291	0.5171	NA
K-Nearest Neighbor	77	99.67	0.076	0.8277	0.8247	0.8276
Bidirectional RNN (LSTM + GRU)	<44	99.13	0.074	0.7467	0.7566	0.7471
Bidirectional RNN (LSTM)	<44	98.85	0.078	0.7388	0.7285	0.7385
Bidirectional RNN (GRU)	<44	99.00	0.075	0.7706	0.7480	0.7704

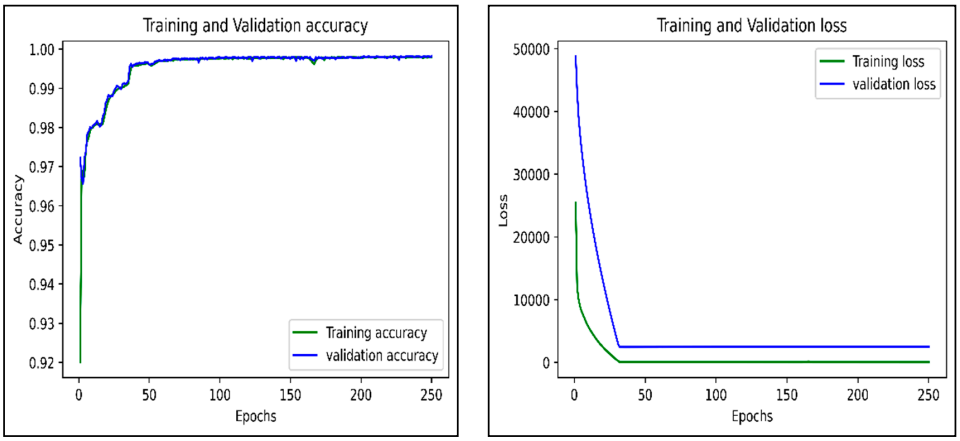
In addition to the proposed model having both LSTM and GRU frameworks, we also tested them separately. The GRU model marginally outperformed compared to the LSTM model in terms of accuracy. However, out of seven sub-datasets, the LSTM model performed better than the GRU and the proposed model, while GRU performed well only in one sub-dataset. Table 13 shows the comparison of traditional techniques with the proposed model. Various traditional techniques, such as Decision Tree, Random Forest, K-means, etc., were compared with our proposed model. The ensemble classifier method [46] tested these techniques using 10 and 13 features from the CICIDS2017 dataset. Their technique achieved 98.4% accuracy with 0.15 of FPR, when using 10 selected features. However, with 13 features, their model scored an accuracy of 97.3% with 0.12 FPR. KODE [47] tested the technique using 11, 8, and 13 features. They were able to achieve accuracy of 96.4%, 98.3%, and 98% with 11, 8, and 13 features, respectively. Their FPR scores and precision score were 1.15, 0.14, 1.12 and 0.99, 0.992, 0.99, respectively. The proposed model outperformed these

traditional techniques in detecting attacks accurately. Moreover, the FPR was also higher in the traditional methods compared to the proposed model. Figures 15–20 show the accuracy and loss curves of training and validation data for all sub-datasets. The x-axis represents the number of epochs executed and the y-axis represents accuracy and loss.

Table 13. Performance comparison (traditional vs. proposed model).

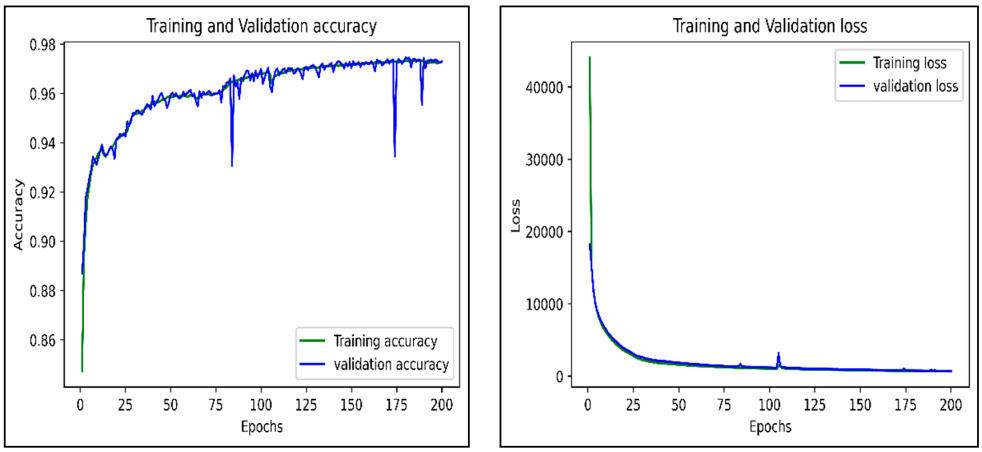
Model	No. of Features	Accuracy (%)	FPR	TPR	Precision	Recall
Traditional ML techniques						
Ensemble classifier [46] (Random Forest, Decision Tree, ForestPA)	10	98.4	0.15	NA	NA	NA
	13	97.3	0.12	NA	NA	NA
KODE [47] K-means, SVM, DBSCAN, Expectation-Maximization	11	96.4	1.15	NA	0.99	NA
	8	98.3	0.14	NA	0.992	NA
	13	98	1.12	NA	0.992	NA
Proposed technique						
Bidirectional RNN (LSTM + GRU)	<44	99.13	0.074	0.7467	0.7566	0.7471
Bidirectional RNN (LSTM)	<44	98.85	0.078	0.7388	0.7285	0.7385
Bidirectional RNN (GRU)	<44	99.00	0.075	0.7706	0.7480	0.7704

(NA—Not Avaialable).



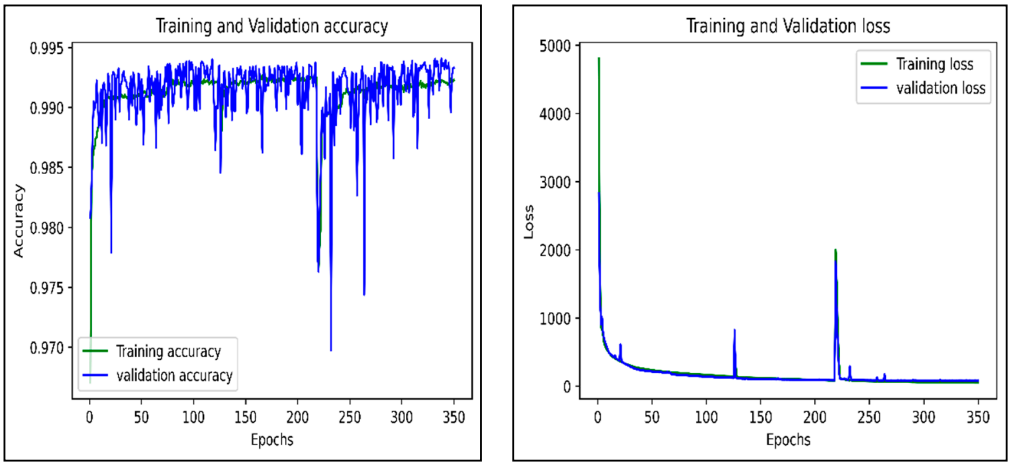
(**A**) (**B**)

Figure 15. (**A**,**B**): Accuracy and Loss curves for Tuesday sub-dataset.



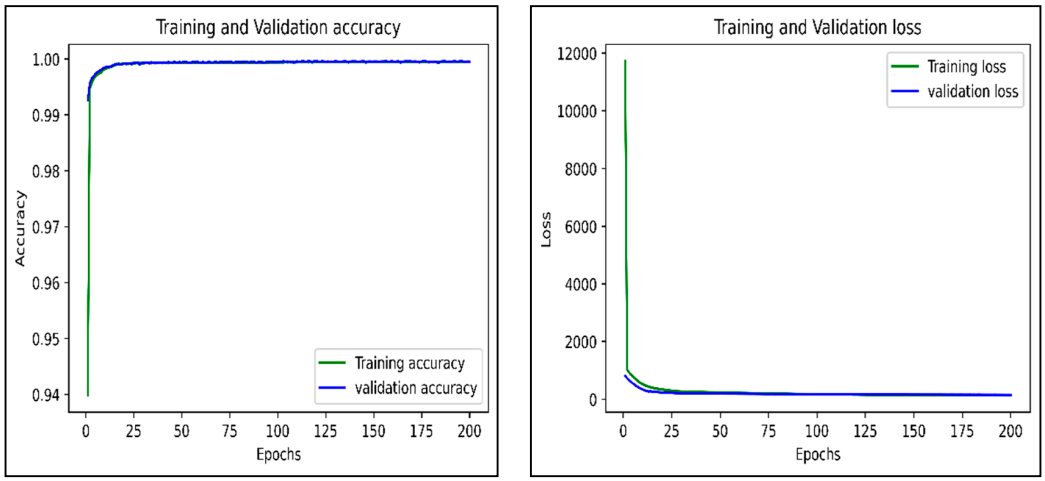
(**A**) (**B**)

Figure 16. (**A**,**B**): Accuracy and Loss curves for Wednesday sub-dataset.



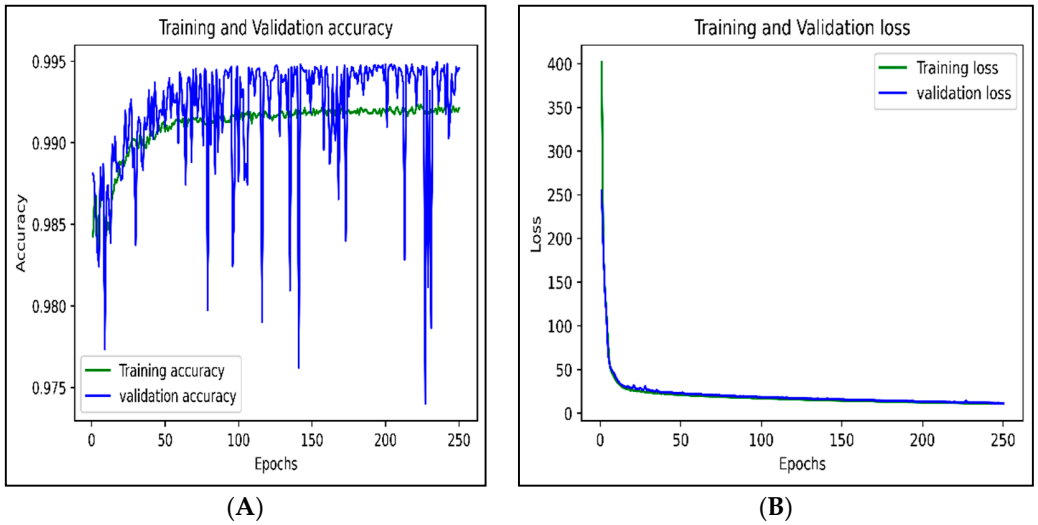
(**A**) (**B**)

Figure 17. (**A**,**B**): Accuracy and Loss curves for Thursday morning sub-dataset.



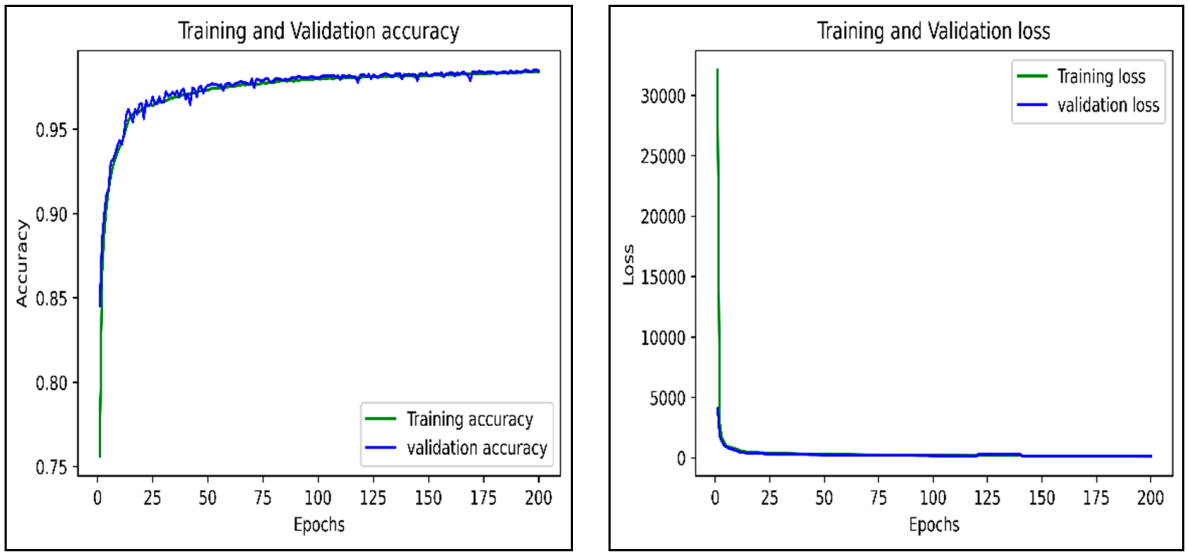
(**A**) (**B**)

Figure 18. (**A**,**B**): Accuracy and Loss curves for Thursday afternoon sub-dataset.



(**A**) (**B**)

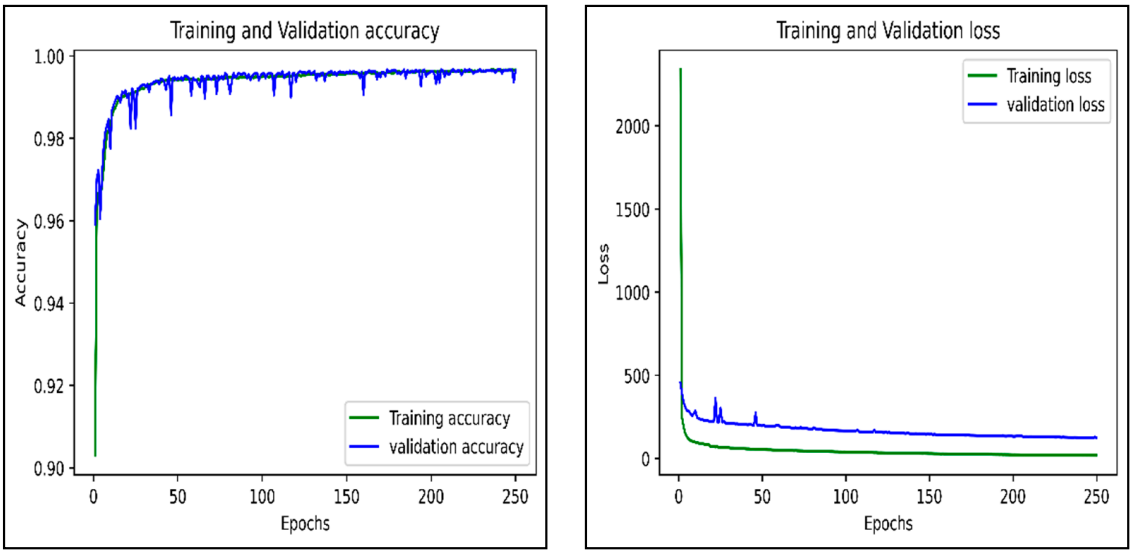
Figure 19. (**A**,**B**): Accuracy and Loss curves for Friday morning sub-dataset.



(**A**) (**B**)

Figure 20. (**A**,**B**): Accuracy and Loss curves for Friday afternoon-DDoS sub-dataset.

In this study, as shown in Figures 15–21, we have used a different number of epochs to gain maximum accuracy and minimum loss for each sub-dataset. The loss represents the incorrect prediction of the labels by the model. We applied the Sparse Categorical Cross-Entropy method to calculate the loss in our proposed model. We observed that the loss curve, as well as the accuracy curve, started flatten after the 100th epoch for most of the sub-datasets. After the 100th epoch, there was very little decrement and increment in loss and accuracy values, respectively.



(**A**) (**B**)

Figure 21. (**A**,**B**): Accuracy and Loss curves for Friday afternoon PortScan sub-dataset.

6. Conclusions

This study's proposal was a model based on deep learning to develop an Intrusion Detection System. We used RNN with LSTM and GRU frameworks, which resolves the longer dependency issue of RNN. We used one LSTM sequence and one GRU sequence with directions opposite to each other instead of using a single directional layer structure. The outputs of these two sequence layers are then combined using concatenate merge mode.

We tested different combinations of LSTM and GRU. In addition, the different directions of LSTM and GRU sequences were also tested. However, the present strategy resulted in better performance. Different optimizers were explored to select the best optimizer for the model. Adaptive Gradient (AdaGrad) proved to be suitable for our proposed model. The suggested model was verified via the CICIDS-2017 dataset.

Evaluation of the proposed model was conducted using performance metrics derived from the confusion matrix. It was observed that the suggested model functioned admirably in predicting the attacks. The proposed model scored a classification accuracy of 99.13% with a very low False Positive rate. The suggested design was compared with other classifiers such as Random Forest, Naïve Bayes, k-Nearest Neighbor (KNN), and Ensemble method. The precision of the suggested model was less than Random Forest, KNN, and the Ensemble method but higher than the Naïve Bayes classifier. However, our model used nearly half the number (i.e., less than 44 features for each sub-datasets) of features when compared to all other classifiers. Moreover, we also tested LSTM and GRU frameworks separately with RNN, which resulted in GRU performing better than LSTM in terms of accuracy.

Although the proposed approach has shown encouraging performance, it can be improved by further optimizing the classifier. The proposed model cannot effectively classify a few attacks. Therefore, in future studies on one side, we aim for the optimum training data for all the attacks and an optimum classifier. Moreover, a strategy to implement the proposed approach in a real-time environment.

Author Contributions: Conceptualization, S.G.; methodology, S.G. and A.H.; validation, M.Z. and M.R.; formal analysis, A.R.J.; writing—original draft preparation, S.G.; writing—review and editing, A.R.J. and P.K.R.M.; supervision, A.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Data in this research paper will be shared upon request to the corresponding author.

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

References

1. Prasad, M.; Tripathi, S.; Dahal, K. An efficient feature selection-based Bayesian and Rough set approach for intrusion detection. *Appl. Soft Comput. J.* **2020**, *87*, 105980. [CrossRef]
2. Dutt, I.; Borah, S.; Maitra, I.K. Immune System Based Intrusion Detection System (IS-IDS): A Proposed Model. *IEEE Access* **2020**, *8*, 34929–34941. [CrossRef]
3. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
4. Sultana, N.; Chilamkurti, N.; Peng, W.; Alhadad, R. Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-Peer Netw. Appl.* **2018**, *12*, 493–501. [CrossRef]
5. Jyothsna, V.; Prasad, V.V.R.; Prasad, K.M. A Review of Anomaly based Intrusion Detection Systems. *Int. J. Comput. Appl.* **2011**, *28*, 26–35. [CrossRef]
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. [CrossRef]
7. Kwon, D.; Kim, H.; Kim, J.; Suh, S.C.; Kim, I.; Kim, K.J. A survey of deep learning-based network anomaly detection. *Clust. Comput.* **2017**, *22*, 949–961. [CrossRef]
8. Fernandez, G.C.; Xu, S. A Case Study on using Deep Learning for Network Intrusion Detection. In Proceedings of the MILCOM 2019–2019 IEEE Military Communications Conference (MILCOM), Norfolk, VA, USA, 12–14 November 2019.
9. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A survey of network-based intrusion detection data sets. *Comput. Secur.* **2019**, *86*, 147–167. [CrossRef]
10. Guo, L.; Li, N.; Jia, F.; Lei, Y.; Lin, J. A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing* **2017**, *240*, 98–109. [CrossRef]

11. Kong, W.; Dong, Z.Y.; Jia, Y.; Hill, D.J.; Xu, Y.; Zhang, Y. Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. *IEEE Trans. Smart Grid* **2019**, *10*, 841–851. [CrossRef]
12. Devarakonda, N.; Pamidi, S.; Kumari, V.V.; Govardhan, A. Intrusion Detection System using Bayesian Network and Hidden Markov Model. *Procedia Technol.* **2012**, *4*, 506–514. [CrossRef]
13. Sajjad, S.M.; Bouk, S.H.; Yousaf, M. Neighbor Node Trust based Intrusion Detection System for WSN. *Procedia Comput. Sci.* **2015**, *63*, 183–188. [CrossRef]
14. Belavagi, M.C.; Muniyal, B. Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection. *Procedia Comput. Sci.* **2016**, *89*, 117–123. [CrossRef]
15. Gaikwad, D.; Thool, R.C. Intrusion Detection System Using Bagging with Partial Decision TreeBase Classifier. *Procedia Comput. Sci.* **2015**, *49*, 92–98. [CrossRef]
16. Ahmad, I.; Basheri, M.; Iqbal, M.J.; Rahim, A. Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection. *IEEE Access* **2018**, *6*, 33789–33795. [CrossRef]
17. Kasongo, S.M.; Sun, Y. A Deep Learning Method With Filter Based Feature Engineering for Wireless Intrusion Detection System. *IEEE Access* **2019**, *7*, 38597–38607. [CrossRef]
18. Yang, H.; Wang, F. Wireless Network Intrusion Detection Based on Improved Convolutional Neural Network. *IEEE Access* **2019**, *7*, 64366–64374. [CrossRef]
19. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
20. Kim, A.; Park, M.; Lee, D.H. AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection. *IEEE Access* **2020**, *8*, 70245–70261. [CrossRef]
21. Wang, M.; Zheng, K.; Yang, Y.; Wang, X. An Explainable Machine Learning Framework for Intrusion Detection Systems. *IEEE Access* **2020**, *8*, 73127–73141. [CrossRef]
22. Hao, Y.; Sheng, Y.; Wang, J. Variant Gated Recurrent Units With Encoders to Preprocess Packets for Payload-Aware Intrusion Detection. *IEEE Access* **2019**, *7*, 49985–49998. [CrossRef]
23. Rezk, N.M.; Purnaprajna, M.; Nordstrom, T.; Ul-Abdin, Z. Recurrent Neural Networks: An Embedded Computing Perspective. *IEEE Access* **2020**, *8*, 57967–57996. [CrossRef]
24. Wei, X.; Liu, Y.; Gao, S.; Wang, X.; Yue, H. An RNN-Based Delay-Guaranteed Monitoring Framework in Underwater Wireless Sensor Networks. *IEEE Access* **2019**, *7*, 25959–25971. [CrossRef]
25. Feng, W.; Guan, N.; Li, Y.; Zhang, X.; Luo, Z. Audio visual speech recognition with multimodal recurrent neural networks. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017.
26. Yuan, J.; Wang, H.; Lin, C.; Liu, D.; Yu, D. A Novel GRU-RNN Network Model for Dynamic Path Planning of Mobile Robot. *IEEE Access* **2019**, *7*, 15140–15151. [CrossRef]
27. Gers, F. Learning to forget: Continual prediction with LSTM. In Proceedings of the 9th International Conference on Artificial Neural Networks: ICANN '99, Edinburgh, UK, 7–10 September 1999.
28. Houdt, G.V.; Mosquera, C.; Nápoles, G. A review on the long short-term memory model. *Artif. Intell. Rev.* **2020**, *53*, 5929–5955. [CrossRef]
29. Xiao, H.; Sotelo, M.A.; Ma, Y.; Cao, B.; Zhou, Y.; Xu, Y.; Wang, R.; Li, Z. An Improved LSTM Model for Behavior Recognition of Intelligent Vehicles. *IEEE Access* **2020**, *8*, 101514–101527. [CrossRef]
30. Yan, K.; Li, W.; Ji, Z.; Qi, M.; Du, Y. A Hybrid LSTM Neural Network for Energy Consumption Forecasting of Individual Households. *IEEE Access* **2019**, *7*, 157633–157642. [CrossRef]
31. Shewalkar, A.; Nyavanandi, D.; Ludwig, S.A. Performance Evaluation of Deep Neural Networks Applied to Speech Recognition: RNN, LSTM and GRU. *J. Artif. Intell. Soft Comput. Res.* **2019**, *9*, 235–245. [CrossRef]
32. Xu, C.; Shen, J.; Du, X.; Zhang, F. An Intrusion Detection System Using a Deep Neural Network With Gated Recurrent Units. *IEEE Access* **2018**, *6*, 48697–48707. [CrossRef]
33. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy, Madeira, Portugal, 22–24 January 2018.
34. Stiawan, D.; Idris, M.Y.; Bamhdi, A.M.; Budiarto, R. CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection. *IEEE Access* **2020**, *8*, 132911–132921.
35. Zhu, H.; You, X.; Liu, S. Multiple Ant Colony Optimization Based on Pearson Correlation Coefficient. *IEEE Access* **2019**, *7*, 61628–61638. [CrossRef]
36. Feng, W.; Zhu, Q.; Zhuang, J.; Yu, S. An expert recommendation algorithm based on Pearson correlation coefficient and FP-growth. *Clust. Comput.* **2018**, *22*, 7401–7412. [CrossRef]
37. Ullah, A.; Ahmad, J.; Muhammad, K.; Sajjad, M.; Baik, S.W. Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features. *IEEE Access* **2018**, *6*, 1155–1166. [CrossRef]
38. Mulder, W.D.; Bethard, S.; Moens, M.-F. A survey on the application of recurrent neural networks to statistical language modeling. *Comput. Speech Lang.* **2015**, *30*, 61–98. [CrossRef]
39. Zhang, Q.; Kong, Q.; Zhang, C.; You, S.; Wei, H.; Sun, R.; Li, L. A new road extraction method using Sentinel-1 SAR images based on the deep fully convolutional neural network. *Eur. J. Remote Sens.* **2019**, *52*, 572–582. [CrossRef]

40. Acheson, E.; Volpi, M.; Purves, R.S. Machine learning for cross-gazetteer matching of natural features. *Int. J. Geogr. Inf. Sci.* **2019**, *34*, 708–734. [CrossRef]
41. Sheba, K.; Raj, S.G. An approach for automatic lesion detection in mammograms. *Cogent Eng.* **2018**, *5*, 1444320. [CrossRef]
42. Wahlberg, F.; Dahllöf, M.; Mårtensson, L.; Brun, A. Spotting Words in Medieval Manuscripts. *Stud. Neophilol.* **2014**, *86* (Suppl. S1), 171–186. [CrossRef]
43. Syed, N.F.; Baig, Z.; Ibrahim, A.; Valli, C. Denial of service attack detection through machine learning for the IoT. *J. Inf. Telecommun.* **2020**, *4*, 482–503. [CrossRef]
44. Bhattacharya, S.; Maddikunta, P.K.R.; Kaluri, R.; Singh, S.; Gadekallu, T.R.; Alazab, M.; Tariq, U. A novel PCA-firefly based XGBoost classification model for intrusion detection in networks using GPU. *Electronics* **2020**, *9*, 219. [CrossRef]
45. Swarna Priya, R.M.; Maddikunta, P.K.R.; Parimala, M.; Koppu, S.; Gadekallu, T.R.; Chowdhary, C.L.; Alazab, M. An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture. *Comput. Commun.* **2020**, *160*, 139–149. [CrossRef]
46. Zhou, Y.; Cheng, G.; Jiang, S.; Dai, M. Building an efficient intrusion detection system based on feature selection and ensemble classifier. *Comput. Netw.* **2020**, *174*, 107247. [CrossRef]
47. Jaw, E.; Wang, X. Feature Selection and Ensemble-Based Intrusion Detection System: An Efficient and Comprehensive Approach. *Symmetry* **2021**, *13*, 1764. [CrossRef]