

Article

Composition of Hybrid Deep Learning Model and Feature Optimization for Intrusion Detection System

Azriel Henry ¹, Sunil Gautam ², Samrat Khanna ¹, Khaled Rabie ^{3,4,*} , Thokozani Shongwe ⁴, Pronaya Bhattacharya ^{5,*} , Bisham Sharma ⁶  and Subrata Chowdhury ⁷

- ¹ Department of Computer Sciences and Engineering, Institute of Advanced Research, Gandhinagar 382426, Gujarat, India
² Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad 382481, Gujarat, India
³ Department of Engineering, Manchester Metropolitan University, Manchester M1 5GD, UK
⁴ Department of Electrical and Electronic Engineering Technology, University of Johannesburg, Auckland Park, P.O. Box 524, Johannesburg 2006, South Africa
⁵ Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University, Kolkata, 700135, West Bengal, India
⁶ Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura 140401, Punjab, India
⁷ Department of Masters of Computer Application, Sri Venkateswara College of Engineering and Technology (A), Chittoor 517127, Andhra Pradesh, India
* Correspondence: k.rabie@mmu.ac.uk (K.R.); pbhattacharya@kol.amity.edu (P.B.)

Abstract: Recently, with the massive growth of IoT devices, the attack surfaces have also intensified. Thus, cybersecurity has become a critical component to protect organizational boundaries. In networks, Intrusion Detection Systems (IDSs) are employed to raise critical flags during network management. One aspect is malicious traffic identification, where zero-day attack detection is a critical problem of study. Current approaches are aligned towards deep learning (DL) methods for IDSs, but the success of the DL mechanism depends on the feature learning process, which is an open challenge. Thus, in this paper, the authors propose a technique which combines both CNN, and GRU, where different CNN–GRU combination sequences are presented to optimize the network parameters. In the simulation, the authors used the CICIDS-2017 benchmark dataset and used metrics such as precision, recall, False Positive Rate (FPR), True Positive Rate (TRP), and other aligned metrics. The results suggest a significant improvement, where many network attacks are detected with an accuracy of 98.73%, and an FPR rate of 0.075. We also performed a comparative analysis with other existing techniques, and the obtained results indicate the efficacy of the proposed IDS scheme in real cybersecurity setups.

Keywords: convolution neural network; IDS; deep learning; anomaly detection



Citation: Henry, A.; Gautam, S.; Khanna, S.; Rabie, K.; Shongwe, T.; Bhattacharya, P.; Sharma, B.; Chowdhury, S. Composition of Hybrid Deep Learning Model and Feature Optimization for Intrusion Detection System. *Sensors* **2023**, *23*, 890. <https://doi.org/10.3390/s23020890>

Academic Editors: Ali Ismail Awad, Muhammad Imran and Mohammed M. Alani

Received: 8 December 2022

Revised: 3 January 2023

Accepted: 5 January 2023

Published: 12 January 2023



Copyright: © 2023 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

Information technology is growing very rapidly throughout the world. It has made information exchange very smooth and easy. However, these advancements have created many challenges for the communication system. Such a system or network faces intrusions in the form of different attacks. An IDS is a tool that can classify or detect potential cyber attacks in the host/network by applying detection algorithms. There are two classes of IDS, Signature and Anomaly based IDSs (SIDS and AIDS). In SIDS, attacks are detected considering the pre-defined pattern/signature of the attacks. In the AIDS network, traffic patterns are monitored and compared with normal or regular patterns in the network to detect any intrusion. Any change in the network is declared as an intrusion in the network by the Anomaly based IDS. It has an advantage over SIDS as it can execute the detection of new attacks in the network. Whereas SIDS can only detect attacks that match the previously

stored signatures. Moreover, for data sources, IDS can be divided into Network- and Host-based IDSs. By looking at the information from the OS, server and firewall, and database logs, or application system audit, HIDS can identify threats from within the system. NIDS can identify external attacks before they reach the computer network. To identify potential network dangers, NIDS tracks and examines network traffic that is gathered from various network data sources [1,2].

There are several approaches to developing SIDS and AIDS. AIDS can nullify the limitations of SIDS and researchers have shown more interest in it. First, AIDS can be created using schemes such as Machine-learning/deep-learning (ML/DL), knowledge-based or statistical-based methods [3]. The statistical method detects intrusion using a statistical system of measurements such as the mean, standard deviation, mode, and median. Univariate, Multivariate, and Time-series models are the models used to implement Statistical IDS. Second, models are developed using knowledge-based techniques based on a set of rules made using human knowledge. Methods to develop knowledge-based IDS involve description languages, expert systems, and finite-state systems. Thirdly, machine learning is an extensively used technique to develop an Anomaly based IDS. The ML approach comprises two classes namely, unsupervised, and supervised learning. In addition to these categories, there is hybrid learning called semi-supervised learning. In contrast to unsupervised learning, which needs unlabelled instances for training, supervised learning needs labelled instances. The semi-supervised technique utilizes fewer categorized instances and more uncategorized input instances for training [4,5].

Machine learning methods to detect cyber intrusions are being widely used because of their automatic and timely manner of action. However, cyber intrusions are continuously changing which creates the requirement for more scalable detection systems. The opportunity to create such scalable and adaptable detection systems is provided by DL techniques. DL can be used with the supervised as well as unsupervised method. Unsupervised algorithms are used to produce labels for unlabelled instances [6,7]. When using techniques such as machine learning/deep learning (ML/DL), the dataset is crucial. Various IDS datasets are available for developing an IDS such as KDD 99 dataset, Centre for Applied-Internet Data-Analysis (CAIDA) dataset, DARPA Lincoln Lap Packet trace, UNSW-NB15, Coburg University published dataset-CIDDS-001, or New Brunswick University published dataset-CICIDS 2017. It includes a wide variety of typical attacks in addition to benign ones. Attacks such as DoS/DDoS, Heartbleed, botnet, SSH bruteforce, infiltration, and web attacks are included in CICIDS 2017 dataset [8,9].

1.1. Novelty of Paper

We used a deep learning algorithm called CNN with GRU framework to develop an IDS. CNN is widely used as a deep learning technique in various fields. However, these methods have a problem with long-term dependencies. The existing deep learning solutions deal with this problem of long-term dependencies. When long-term interactions are provided exponentially decreasing weights, the long-term dependency problem occurs. Consequently, it becomes challenging for the model to track previous data. GRU can fix this issue and enhance the deep learning model. Hence to effectively detect the attacks we used the GRU framework with RNN.

1.2. Contribution of Paper

To effectively classify the attacks, we designed an IDS using a deep learning technique. We used CNN to detect the attacks in IDS. To overcome the limitations of the existing DL technique as discussed in the previous section, the technique is modified using the GRU framework. We tried different combinations of CNN layers and GRU sequences. However, we achieved better performance with three CNN layers and two GRU layers. The detailed model strategy is shown in the latter section. In addition, we optimized the CICIDS 2017 dataset for better performance. The optimization includes the removal of redundant features and the selection of distinguished features.

1.3. Organization of Paper

The remaining text is presented in the below order: Section 2 manifests the most recent research on this subject. The proposed method is showcased in Section 3. The main contribution of the research is also covered in this section and it discusses the proposed technique in detail. Section 4 analyses the model's performance using a variety of parameters, i.e., findings and discussion. The work of this study comes to a close in Section 5.

2. Related Work

The scope to enhance the IDS is prevailing despite the numerous varieties of approaches to deal with the same. Recent years have seen an increase in the development of IDSs based on machine learning relative to previous methodologies. Nevertheless, there are various methods to develop an IDS which performs differently in various scenarios.

Roberto et al. [10] illustrated a model to detect network intrusions using four ML models namely Multinomial LR, SVC-L and SVC-RBF, and RF. They evaluated their models using the UGR'16 dataset. They considered four attacks namely e Botnet, DoS, Scan, and Spam. They assert that the scientific community will benefit from their findings by developing better NIDS solutions. However, this research reflects on only four attacks.

Amar et al. [11] demonstrated ML techniques such as KNN, Naïve Bayes, Logistic regression, and SVM. They used the NSL-KDD to check the effectiveness of their models. They claimed that for binary and multi-class classification, KNN is superior to NB, and RF SVM with a precision factor of 93.28% to above 99.4% and an accuracy of 96.69%.

Iram et al. [12] demonstrated a detailed study using machine learning algorithms namely SVM, KNN, LR, NB, MLP, RF, ETC, and DT. They randomly selected the features from the NSL-KDD to reduce the dimension of the dataset. They claimed that RF, extra-tree, and DT classifiers managed to achieve an accuracy score of above 99%. However, they did not focus on the sight of executing optimization methods. Abdulsalam et al. [13] showcased the use of DT, RF, and XGBoost to develop an IDS for software-defined networks (SDN). NSL-KDD was utilized to test the effectiveness of these models. XGBoost classifier outperformed other classifiers when evaluated using various metrics such as F1 score, precision, recall, etc. However, they stated that deep learning algorithms, such as Auto-Encoder, GANs, and RNNs, such as GRU and LSTM can be used to conveniently detect the changes in the network. Raisa et al. [14] demonstrated the analysis of the learning algorithms namely DT, GBT, AdaBoost, MLP, LSTM, and GRU for detecting the intrusions. They used UNSW-NB 15 and Network TON datasets to test these models. To optimize the dataset, they used an embedded technique called the GIWRF model. The decision tree outperformed other classifiers in this study. However, this study lacks the multiclass classification.

Achmad et al. [15] demonstrated a hybrid strategy that incorporates the feature optimization method, which stands for supervised method, and the data reduction method, which stands for unsupervised method. Attribute importance DT-based technique with recursive feature removal is used to pick pertinent and important attributes, and the LOF method is used to identify anomalous or outlier data. To check the effectiveness of their model they used NSL-KDD and UNSW-NB15. They claimed that their model increases accuracy of the system when compared with other existing models. They noted that their model needs improvement in terms of sensitivity, specificity, and FAR.

Gustavo et al. [16] introduced the AB-TRAP framework to facilitate the full deployment of the solution, which allows the use of new network traffic and takes operational factors into account. Their methodology includes developing attack and legitimate datasets, training machine learning models, putting the solution into practice on a target system, and assessing the effectiveness of the security module. They demonstrated the performance of the ML models such as KNN, RF, XGB, NB, DT, MLP, SVM, and LR. They claimed that the decision tree (DT) provides the best result compared with other models. However, their model needs to be tested on the standard datasets containing a variety of attacks.

Maonan et al. [17] proposed a method to enhance interpretation of IDSs, this approach employs Shapley Additive Explanations (SHAPs) and integrates local and global explanations.

The local explanations provide the justifications for the decisions the model makes in response to a particular input. The NSL-KDD dataset is utilized to test the framework's viability. They used KNN, RF, SVM-RBF, one-vs.-all, and multiclass models as learning models. The experimental outcomes demonstrate that the interpretation outcomes produced by their framework are consistent with the traits of the attacks, and the outcomes are quite comprehensible.

Samson et al. [18] demonstrated the use of CNN to distinguish the attacks in IDS. To evaluate the model, they used CICIDS 2017 dataset. They managed to reach an accuracy rate of 94.96%. Moreover, they claimed that their model can detect new DoS instances other than the instances in the training phase. Priyanka et al. [19] showcased different IDS techniques based on CICIDS 2017 dataset. They divided the process into two categories, namely two class and multi-class for RF, NB, and Convolution Neural Network (CNN) techniques. They noticed that Nave Bayes had poor accuracy for two classes and had a lower multi-class performance. Moreover, RF has high accuracy compared with the CNN model. However, they used a partial dataset for evaluation. Sun et al. [20] proposed a method called CNN-LSTM. They modified the CNN model using LSTM for improving the detection rate. The model's effectiveness was assessed using CICIDS 2017 dataset. Their result shows an accuracy of 98.7%. However, for a few attacks such as Heartbleed and SSH-Patator attacks the model showed a low detection accuracy rate.

Mario et al. [21] showcased the experiment-based comparison of neural-based techniques. The key focus of their work is on ANN. They used CICIDS2017/2018 and KDD99 datasets to evaluate their model. They claimed that the ANN-based techniques perform outstandingly in almost all cases, but as a result of the backpropagation technique they have the disadvantage of being slow. However, they lack the feature optimization step which can reduce time complexity of the classifier.

Shi et al. [22] discussed the use of the Semi-Supervised Deep Reinforcement method. The SSDDQN uses an auto encoder to recreate the attributes first, and a deep NN as a second step. NSL-KDD and AWID datasets were used in their experiment for testing and training. They claimed that for abnormal traffic their model achieved good results. However, the SSDDQN model's optimization impact is constrained, and it has essentially no detection capacity for smallest number of U2R anomalous attack traffic.

Charlotte et al. [23] demonstrated the comparison of random forest with two deep learning algorithms: RNN and CNN. They carried out the experiment using Sentinel-2 time series. Analysing the algorithms, they claimed that CNN obtains the highest accuracy. Moreover, they also claimed that RNNs have more time complexity and less accuracy.

Joohwa et al. [24] presented an approach for deep learning classification using features that were extracted, not as a classification approach, but as a pre-processing technique for feature extraction. An unsupervised deep learning autoencoder model that is typically employed is classified by the Random Forest (RF) classification method, and features are extracted from that model using a deep sparse autoencoder. They used CICIDS 2017 dataset to perform the experiments. They claimed that the proposed approach was compared with existing methods for feature extraction and it was superior. However, the performance of the approach was somewhat poor for the rare class that existed in the network.

Mohammadnoor et al. [25] showcased a multi-stage optimised ML-based NIDS framework. They examined how oversampling approaches affect the size of the training instances for models and establishes smallest training sample size. They compared gain and correlation-based techniques for feature selection. They used two datasets to evaluate their model, namely CICIDS 2017 and UNSW-NB 2015. They claimed that their model achieves the accuracy rate of more than 90% while using only up to 50% of features.

The related works demonstrate a range of Intrusion Detection System implementation strategies. Most of these suggested works test their models using the datasets KDDCUP and NSL-KDD. These datasets do not, however, exhibit a lot of attribute variability. We selected the CICIDS-2017 dataset for this analysis since it has more features and dangers than the KDD dataset. To produce an effective strategy, this proposed effort concentrates

on reducing the input size, or features, by using a legitimate feature optimization technique. Table 1 shows the summary of the related works discussed above.

Table 1. Summary of the related work.

| Reference | Existing Technique | Summary |
|--------------------------------|--|--|
| Roberto et al., (2020) [10] | NIDS -ML Approaches | Classifier: Logistic regression (LR), SVC-L, SVC-RBF, RF Dataset: UGR'16 Metrics: Precision, Recall, F1, AUC Programming: Python, scikit-optimize |
| Amar et al., (2020) [11] | Hybrid IDS using ML | Classifier: KNN, NB, LR, SVM Dataset: NSL-KDD Metrics: Accuracy, DR, FAR, Precision, Recall, F1 |
| Iram et al., (2020) [12] | An ML Approach for IDS on NSL-KDD Dataset | Classifier: SVM, KNN, LR, NB, MLP, RF, ETC, DT Dataset: NSL-KDD Metrics: Precision, Recall, Accuracy, F1 |
| Abdulsalam et al., (2021) [13] | NIDS Based on Machine Learning for SDNs | Classifier: DT, RF and XGBoost Dataset: NSL-KDD Metrics: Accuracy, DR, ROC, F-score, Precision, Recall |
| Raisa et al., (2022) [14] | IDS using GIWRF feature selection technique | Classifier: DT, GBT, MLP, AdaBoost, LSTM, GRU Dataset: UNSW-NB 15, Network ON_IoT Metrics: Precision, Recall, F1, FPR Programming: Python (Jupyter Notebook) |
| Achmad et al., (2021) [15] | Hybrid ML method for increasing the performance of network IDS | Classifier: Decision Tree Dataset: NSL-KDD, UNSW-NB15 Metrics: Accuracy, sensitivity, specificity, false alarm rate Programming: Python (Jupyter Notebook) |
| Gustavo et al., (2021) [16] | An End-to-End Framework for ML-Based NIDS | Classifier: KNN, RF, XGB, NB, DT, MLP, SVM, and LR Dataset: proposed dataset, MAWILab dataset Metrics: F1, FPR, precision, recall, storage, time, CPU and RAM usage Programming: Python (scikit-learn) |
| Maonan et al., (2020) [17] | An explainable ML Framework for IDS | Classifier: KNN, RF, SVM-RBF, one-vs.-all, and multiclass classifier Dataset: NSL-KDD Metrics: Accuracy, F1, FPR, precision, recall, connections between certain characteristics and attack types Programming: Python (Pytorch) |
| Samson et al., (2021) [18] | Detection of Known, Innovative Cyberattacks Using CNN | Classifier: Convolution Neural Network CNN Dataset: CICIDS 2017 Metrics: TNR, DR, accuracy, FPR |
| Priyanka et al., (2020) [19] | Performance Assessment of IDS-CICIDS-2017 Dataset | Classifier: RF, NB, CNN Dataset: CICIDS2017 (partial) Metrics: Precision, Recall, F1, accuracy Programming: Python (Jupyter Notebook) |
| Sun et al., (2020) [20] | CNN-LSTM hybrid network | Classifier: CNN-LSTM Dataset: CICIDS 2017 Metrics: Accuracy, TPR, FPR, F1 Programming: Python |
| Mario et al., (2020) [21] | Neural-based approaches for Network Intrusion Management | Classifier: ANN Dataset: CICIDS2017/2018 and KDD99 Metrics: Accuracy, f-measure, precision, recall, time complexity Programming: Python |

Table 1. Cont.

| Reference | Existing Technique | Summary |
|----------------------------------|---|--|
| Shi et al., (2021) [22] | Semi-Supervised Deep Reinforcement Learning | Classifier: SSDDQN Dataset: NSL-KDD and AWID Metrics: Accuracy, precision, recall, DR, FPR, efficiency Programming: Python |
| Charlotte et al., (2019) [23] | DL for the Classification of Sentinel-2 | Classifier: RF, RNN, CNN Dataset: Sentinel-2 images Metrics: Accuracy, runtime Programming: Python |
| JooHwa et al., (2020) [24] | NIDS using Deep Sparse Autoencoder | Classifier: Single RF, DSAE-RF Dataset: CICIDS 2017 Metrics: Accuracy, precision, F1 Programming: Python |
| Mohammadnoor et al., (2020) [25] | Multi-Stage Optimized ML for NIDS | Classifier: KNN, RF Dataset: CICIDS 2017 and the UNSW-NB 2015 Metrics: Accuracy, precision, recall, FAR Programming: Python |

3. Proposed IDS Model

To propose an IDS model, we used a deep learning algorithm called CNN with GRU. CNN is widely used as a deep learning technique in various fields. However, such techniques have long-term dependencies issues which can be addressed using GRU as discussed in the Section 1. Hence to effectively classify the attacks we used the GRU framework with RNN. Moreover, to reduce the data dimensions we pre-processed the dataset using two techniques discussed in Section 3.2.

3.1. Workflow

This segment manifests the flow of our proposed work. Proposed work involves two sections, namely feature optimization technique and the CNN–GRU model. Data pre-processing and classification are the two processes that form the entire workflow in Figure 1. The elimination of repetition and choosing the ideal feature set are part of the first stage. The second stage includes categorising the data and computing several parameters. The assessment of the technique’s performance in comparison with other current algorithms is shown in the subsequent part.

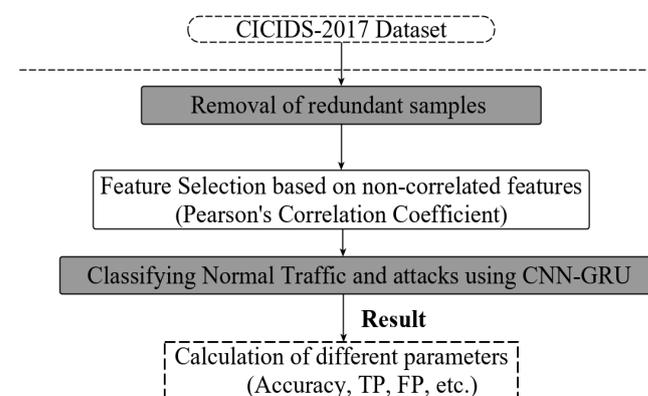


Figure 1. Proposed model layout.

3.2. Data Pre-Processing

CICIDS 2017 dataset was used to test the effectiveness of our technique. Based on different protocols, the abstract behaviour of the 25 users was analysed. They captured the

data for five consecutive days. This dataset includes various categories of attacks such as brute force, DoS/DDoS, web attacks, infiltration, botnet, port scan, etc. Table 2 displays the assault distribution in the dataset. Hence, this dataset is highly recommended to be used to test the model.

Table 2. Dataset attacks.

| Sub-Dataset | Attacks |
|-----------------------------------|---|
| Tuesday Samples | benign ftpPatatorAttack sshPatatorAttack |
| Wed. Samples | benign goldeneyeAttack hulkAttack slowhttpstestAttack slowlorisAttack heartbleedAttack |
| Thur. Morning Samples | benign bruteForceAttack SqlInjectionAttack XSSAttack |
| Thurs. Afternoon Samples | benign infiltrationAttack |
| Fri. Morning Samples | benign botAttack |
| Fri. Afternoon Samples-DDoS | benign ddosAttack |
| Friday Afternoon Samples-PortScan | benign portscanAttack |

We removed the redundant instances from dataset to decrease the dimensions of dataset for classification. The dataset contained 2,300,825 instances before the removal of the redundant samples. The samples were reduced for each sub-dataset. The pattern of the same is shown in Table 3.

Table 3. Distribution of data in the dataset.

| Sub-Dataset | Number of Instances | |
|-------------------------------|---------------------|--------------------|
| | With Redundancy | Without Redundancy |
| Tuesday_BruteForce | 445,909 | 425,240 |
| Wednesday_DoS/DDoS | 692,703 | 613,287 |
| Thurs_Morning_WebAttacks | 170,366 | 164,300 |
| Thurs. Afternoon_Infiltration | 288,602 | 254,625 |
| Fri. Morning_Bot | 191,033 | 184,145 |
| Fri. Afternoon-DDoS | 225,745 | 223,666 |
| Fri. Afternoon-PortScan | 286,467 | 214,114 |

Moreover, by choosing the best feature subset, we employed a feature optimization strategy to lower the input dimension. The discriminative features are found in the feature set using a filter method called Pearson's Correlation Coefficient. It determines how comparable the dataset's features or qualities are and provides a correlation coefficient value in the $[-1, 1]$ range. It indicates a fully positive correlation when it equals 1, and a fully negative correlation when it equals -1 . This suggests that a substantial connection

exists between features and a high value for the coefficient, and vice versa. The Pearson Correlation Coefficient equation is as below [26,27].

$$\rho_{X,Y} = \frac{(X,Y)}{\sigma_X\sigma_Y} \quad (1)$$

Which can be derived as

$$\begin{aligned} &= \frac{E((X - \mu_X)(Y - \mu_Y))}{\sigma_X\sigma_Y} \\ &= \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}\sqrt{E(Y^2) - E^2(Y)}} \end{aligned}$$

where ' (X,Y) ' is a covariance measure for ' X ' and ' Y ', ' σ_X ' and ' σ_Y ' are the standard deviations for ' X ' and ' Y ' respectively, ' $E(X)$ ' is the expected value of ' E '.

The dataset's feature subsets calculated using the Pearson Correlation Coefficient equation are displayed in Table 4. Each subset consists of about 40 features out of the original dataset's total of 77 features. To choose these attributes, a smaller set of samples without duplicate or redundant instances is also used. Pre-processing the original dataset makes it simpler and more effective when used with the suggested model.

Table 4. Selected attributes.

| Sub-Dataset | Total Features | Selected Features |
|--------------|----------------|-------------------|
| BruteForce | 77 | 43 |
| DoS/DDoS | 77 | 41 |
| WebAttacks | 77 | 39 |
| Infiltration | 77 | 40 |
| Bot | 77 | 37 |
| DDoS | 77 | 39 |
| PortScan | 77 | 37 |

3.3. Gated Recurrent Unit (GRU)

GRU is a new structure made to address the vanishing/exploding gradient problem. The upgraded LSTM framework is called GRU. For regulating the information flow, GRUs also has a gate structure similar to that of an LSTM. However, unlike LSTM, GRU lacks an output gate, allowing the content to be fully exposed. The reset and update gates are the only two gates in the GRU. The input and forget gates of the LSTM framework are combined in the second gate. Compared with LSTM, GRUs have a simpler structure and fewer parameters, which improves performance. The GRU framework has the following structure in Figure 2.

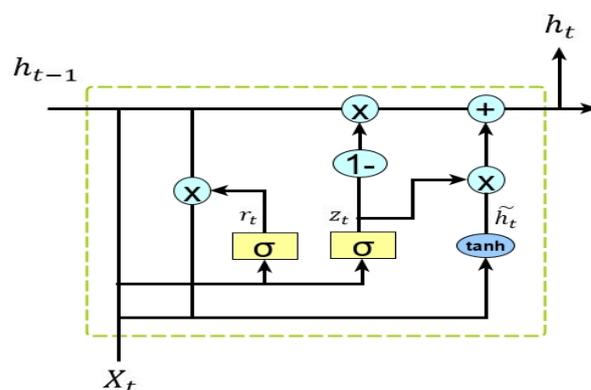


Figure 2. Architecture of GRU.

The equations below provide the GRU formulation.

$$r_t = \text{sigm}(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \quad (2)$$

$$z_t = \text{sigm}(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \quad (3)$$

$$\tilde{h}_t = \text{tanh}(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \quad (4)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \quad (5)$$

where ' x_t ', ' h_t ', ' r_t ', and ' z_t ' are the input and output vectors, reset and update gates respectively. Similar to LSTM, ' b ' stands for biases and ' W ' for weight, while sigmoid and tangent have the functions ' sigm ' and ' tanh ', respectively, for activation. Both LSTM and GRU can manage the longer dependencies. However, in terms of performance, there are some variations. In this study, we employed both frameworks to evaluate how well they classified network traffic [28,29].

3.4. Convolution Neural Network (CNN)

The basic architecture of CNN includes three main components, namely, convolutional, pooling, and output layer. The pooling layer is elective. The classic CNN structure having three convolution layers is widely used in image classification. It contains one input layer, several hidden layers (hidden layers include convolutional, pooling, and normalization) and a layer entirely connected to the last layer called the output layer. The neurons in one layer communicate with those in layers next to it. To reduce the proportions of the input, pooling and sub-sampling processes are executed. CNN classifier receives the input images as a group of small sub-sections which are called receptive fields. The response to the following layer is calculated with the help of mathematical convolution operations of the first or the input layer [30]. We modified the CNN architecture using the framework called GRU which is discussed in the previous section. Table 5 shows the complete architecture and strategy used in the proposed approach. The basic structure of CNN is shown in Figure 3.

Table 5. Model strategy.

| Parameter | 1 | 2 | 3 | 4 | 5 | 6 |
|-----------------|------|------|------|------|------|----------------|
| Type | C | C | C | G | G | H |
| No. of neurons | 32 | 32 | 32 | 64 | 64 | No. of classes |
| Filter Size | (1) | (1) | (1) | (1) | (1) | (1) |
| Activation Fun. | ReLU | ReLU | ReLU | ReLU | ReLU | S_{\max} |

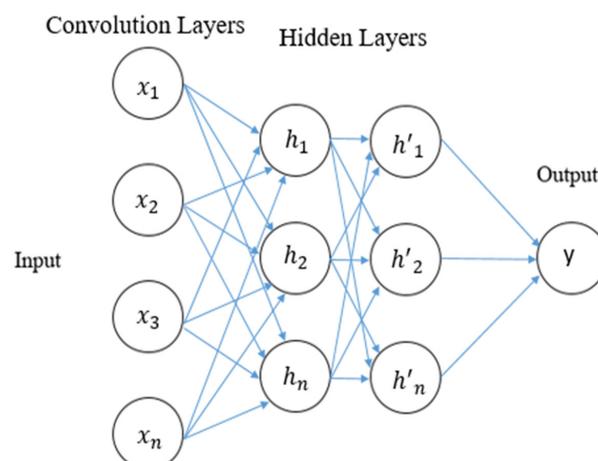


Figure 3. The architecture of CNN.

As shown in Table 5, the CNN–GRU model involves three convolution layers (C) followed by two GRU layers (G) and one hidden layer (H). We used 32 neurons for the C layers and 64 neurons for the G layers. The neurons for the last layer (i.e., the output layer) are equivalent to the labels in the dataset. G and C layers use the ReLU function for the activation. The rectified linear activation function, or ReLU, output zero if the input is negative and the input directly if it is positive. The soft-max (S_{max}) activation function, which provides the output in terms of prediction probabilities, is used in the final layer. With the use of Softmax, a vector of numbers can be converted into a vector of probabilities, where each value's probability is inversely proportional to its relative scale. The detailed steps are shown in the Algorithm 1.

Algorithm 1. Proposed Model

Feature Optimization and CNN–GRU

Input:

Data instances

Output:

Confusion Matrix

(Accuracy, precision, recall, FPR, TPR)

Dataset Optimization

Remove the redundant instances

Feature Selection

Using Pearson's Correlation equation, compute the correlation of the attribute set Set C_f .

if corr_value > 0.8

 add attribute to C_f

else

 increment in an attribute set C

return C_f

Classification

Create training and testing sets from the dataset.

Training set: 67%

Testing set: 33%

add model

 three Convolution layers (activation = 'relu')

 two GRU layers (activation = 'relu')

model compilation

 loss function: 'categorical_crossentropy'

 optimizer='adagrad'

 training CNN–GRU technique with training instances

 employing techniques to test instances

return Confusion Matrix $C_{m \times m}$

4. Findings and Discussion

This section includes the assessment of the proposed technique along with a discussion of the parameters used to test its performance. In the dataset, the confusion matrix is computed for each sub-dataset. The observations of the same are discussed in the later section called performance analysis.

4.1. Evaluation Metrics

For each sub-dataset, we considered a number of performance metrics to test the technique. The confusion matrix is the method's final output. A confusion matrix is an effective tool for analysing how exactly the model locates occurrences of various labels. A confusion matrix is sometimes referred to as a model's performance summary. The confusion matrix diagram is presented in Figure 4. Precision and recall are the most often used metrics for gauging the effectiveness of DL models. Precision is the ratio of the technique's correct or wrong estimates, whereas recall refers to the percentage of the total

number of true matches and the total number of positive matches [31]. These metrics are showcased in the form of equations as follows [32,33].

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \quad (6)$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \quad (7)$$

$$\text{TPR} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \quad (8)$$

$$\text{FPR} = \frac{\text{FP}}{(\text{FP} + \text{FN})} \quad (9)$$

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \quad (10)$$

| | | 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 4. Confusion matrix.

True Positive, False Positive, False Negative, and True Negative, respectively, are represented by the abbreviations TP, FP, FN, and TN.

Moreover, True Positive Rate (TPR) and False Positive Rate (FPR) are also used to analyse the model performance. The TPR measures how many actual correct instances there are in the correct matches. The FPR counts the percentage of mismatched events among non-target occurrences. The ideal model, for instance, generates a TPR of 100% and an FPR of 0%. The TPR and FPR formulae are provided in Equations (8) and (9) correspondingly. Effectiveness of the method is then evaluated using accuracy. It is determined by applying Equation (10) to calculate the fraction of cases that the model properly classified [34,35].

4.2. Performance Analysis

Python programming was used to conduct the experiments on the Google Colab Cloud Environment. The dataset's confusion matrix serves as the test's output. Figures 5–11 display the confusion matrix for each subset of the entire dataset. Table 6 through 12 display several parameters that were produced from the confusion matrix that was explained in Section 4.1. The whole dataset is distributed into 67% of the training data and 33% of the testing data.

The dataset contains over two million samples and 77 attributes. The model performed better when redundant samples and correlated features were removed from the dataset. The confusion matrix is used to determine the testing measures. The rows correspond to the true samples of the labels, while the columns of the matrix show the expected samples of the classes. Values are found diagonally in the matrix, i.e., the True Positive (TP) metric, of the brute force dataset (137,150, 1947, and 968). As more samples are examined, the model's high classification rate becomes clearer. True negative samples, however, are samples that are incorrectly classified. Similar to this, other metrics are computed using the corresponding formulas described in Section 4.1 from the confusion matrix.

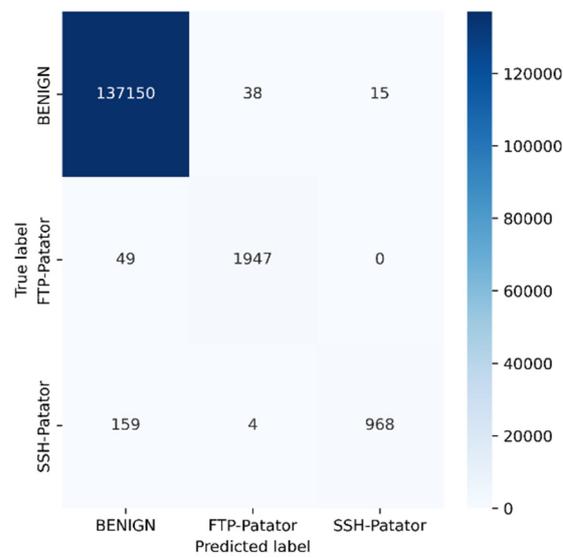


Figure 5. Brute force.

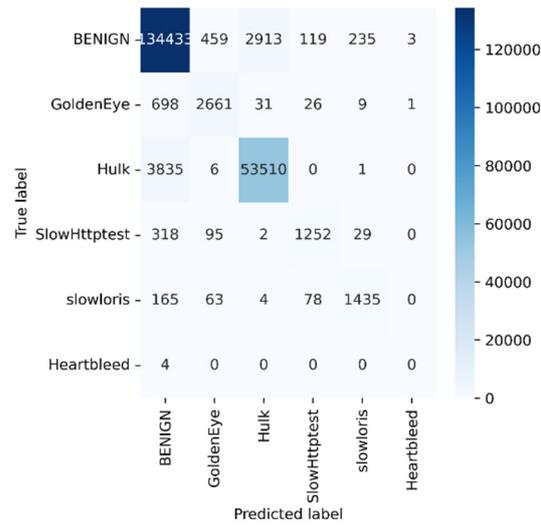


Figure 6. DoS/DDoS.

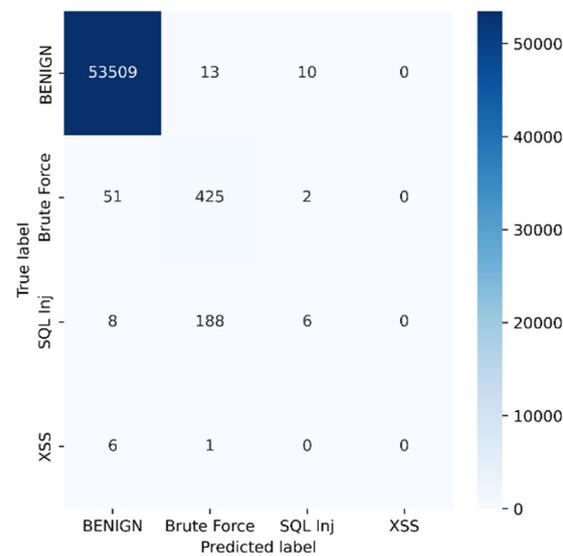


Figure 7. Web attacks.

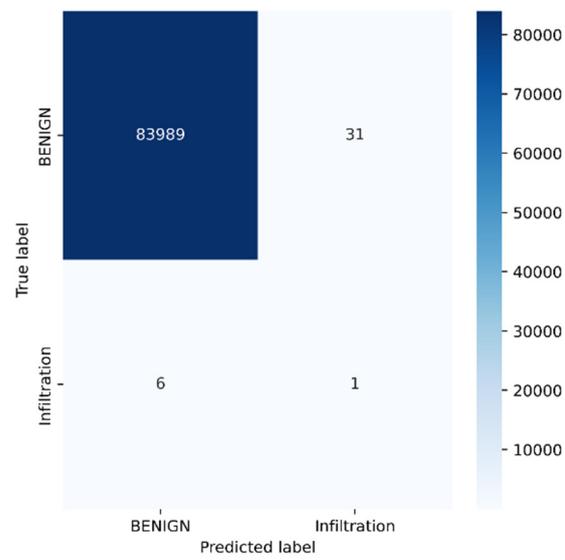


Figure 8. Infiltration.

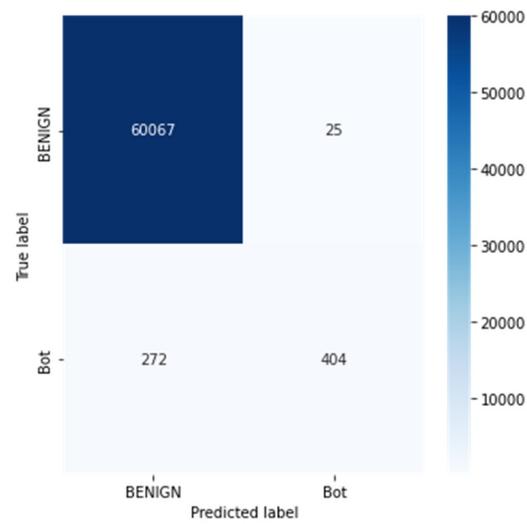


Figure 9. Bot.

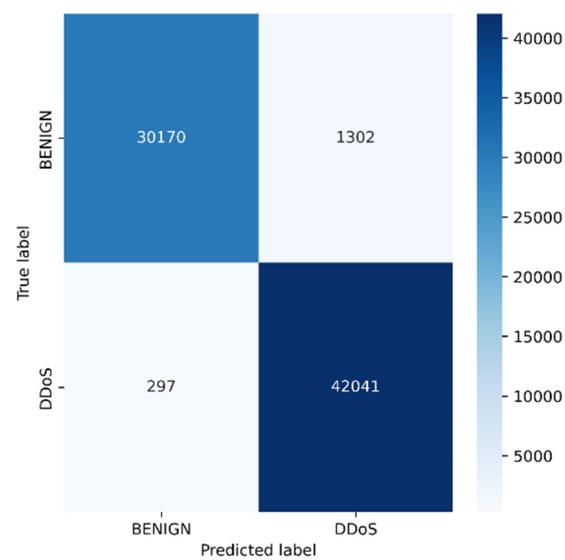


Figure 10. DDoS.

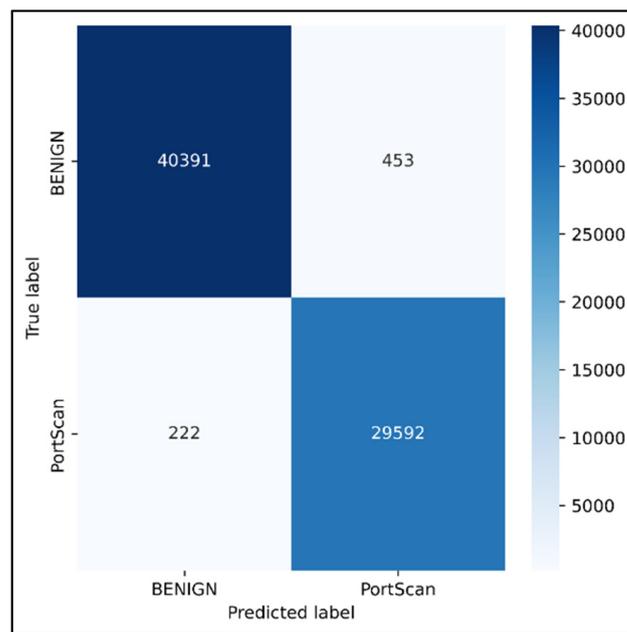


Figure 11. PortScan.

Table 6. Statistical analysis of brute force.

| Parameters | BENIGN | FTPPatator | SSHPatator |
|------------|---------|------------|------------|
| Precision | 0.9984 | 0.9788 | 0.9847 |
| Recall | 1.00 | 0.98 | 0.86 |
| TN | 2919 | 138,292 | 139,184 |
| FP | 208 | 42 | 15 |
| TP | 137,150 | 1947 | 968 |
| FN | 53 | 49 | 163 |
| FPR | 0.0665 | 0.0003 | 0.0001 |
| TPR | 0.9996 | 0.9754 | 0.8558 |

Tuesday sub-dataset's statistical parameter is displayed in Figure 5 and Table 6. It demonstrates the predicted attack types, FTP-Patator and SSH-Patator. For each class, it was noted that our technique acquired a high True Positive Rate and precision value. Figure 6 and Table 7 display the analysis of DoS/DDoS attack, which is the biggest sub-dataset among other sub-datasets. In comparison with other sub-datasets, it also experiences the most attacks. Slowhttptest, GoldenEye, slowloris, Heartbleed and Hulk are the predicted attacks. The attack Hulk is seen to have the highest True Positive Rate, but the model fails to classify any Heartbleed attack samples properly. The statistical information from the Thursday morning sub-dataset is displayed in Figure 7 and Table 8. It displays prediction of XSS, SQL Injection, and brute force attacks. Compared with SQL injection and XSS assaults, the brute force attack was predicted more precisely. Statistical properties of the Thursday and Friday sub-datasets are displayed in Figures 8–11 and Tables 9–12. These reports demonstrate the prediction of attacks including infiltration, bot, DDoS, and port scans. Our approach appears to perform well when predicting Bot, DDoS, and PortScan. It struggles while trying to foresee the Infiltration, though.

Table 7. Statistical analysis of DoS/DDoS.

| Parameters | BENIGN | Goldeneye | Hulk | Slowhttptest | Slowloris | Heartbleed |
|------------|---------|-----------|---------|--------------|-----------|------------|
| Precision | 0.9640 | 0.8102 | 0.9477 | 0.8488 | 0.8396 | 0 |
| Recall | 0.97 | 0.78 | 0.93 | 0.74 | 0.82 | 0 |
| TN | 59,203 | 198,336 | 142,083 | 200,466 | 200,366 | 202,377 |
| FP | 5020 | 623 | 2950 | 223 | 274 | 4 |
| TP | 134,433 | 2661 | 53,510 | 1252 | 1435 | 0 |
| FN | 3729 | 765 | 3842 | 444 | 310 | 4 |
| FPR | 0.078 | 0.003 | 0.020 | 0.001 | 0.001 | 0 |
| TPR | 0.9730 | 0.7767 | 0.9330 | 0.7382 | 0.8223 | 0 |

Table 8. Statistical analysis of Web attacks.

| Parameters | BENIGN | Brute Force | Sql Inj | XSS |
|------------|--------|-------------|---------|--------|
| Precision | 0.9987 | 0.6778 | 0.3333 | 0 |
| Recall | 1.00 | 0.89 | 0.03 | 0.00 |
| TN | 622 | 53,539 | 54,005 | 54,212 |
| FP | 65 | 202 | 12 | 0 |
| TP | 53,509 | 425 | 6 | 0 |
| FN | 23 | 53 | 196 | 7 |
| FPR | 0.0946 | 0.0037 | 0.0002 | 0 |
| TPR | 0.9995 | 0.8891 | 0.0297 | 0 |

Table 9. Statistical analysis of infiltration.

| Parameters | BENIGN | Infiltration |
|------------|--------|--------------|
| Precision | 0.9999 | 0.0312 |
| Recall | 1.00 | 0.14 |
| TN | 1 | 83,989 |
| FP | 6 | 31 |
| TP | 83,989 | 1 |
| FN | 31 | 6 |
| FPR | 0.8571 | 0.0003 |
| TPR | 0.9996 | 0.1428 |

Table 10. Statistical analysis of Bot.

| Parameters | BENIGN | Bot |
|------------|--------|--------|
| Precision | 0.9954 | 0.9417 |
| Recall | 1.00 | 0.60 |
| TN | 404 | 60,067 |
| FP | 272 | 25 |
| TP | 60,067 | 404 |
| FN | 25 | 272 |
| FPR | 0.4023 | 0.0004 |
| TPR | 0.9995 | 0.5976 |

Table 11. Statistical analysis of DDoS.

| Parameters | BENIGN | DDoS |
|------------|--------|--------|
| Precision | 0.9902 | 0.9699 |
| Recall | 0.96 | 0.99 |
| TN | 42,041 | 30,170 |
| FP | 297 | 1302 |
| TP | 30,170 | 42,041 |
| FN | 1302 | 297 |
| FPR | 0.0070 | 0.0413 |
| TPR | 0.9586 | 0.9929 |

Table 12. Statistical analysis of PortScan.

| Parameters | BENIGN | PortScan |
|------------|--------|----------|
| Precision | 0.9945 | 0.9849 |
| Recall | 0.99 | 0.99 |
| TN | 29,592 | 40,391 |
| FP | 222 | 453 |
| TP | 40,391 | 29,592 |
| FN | 453 | 222 |
| FPR | 0.0074 | 0.0110 |
| TPR | 0.9889 | 0.9925 |

The calculation of accuracies for each sub-dataset in the whole dataset is shown in Table 13. Multiple epoch settings were used to calculate each sub-accuracy dataset while taking the loss of training/validation data into account. For all subsets, it was noted that the model effectively attained greater than 95% accuracy. The Tuesday sub-dataset has the highest accuracy compared with the other subsets. Moreover, we also compared the proposed model with other Convolution Neural Network models. It was reported that the suggested model CNN-GRU achieves a classification accuracy score of 98.73% while using just about half (i.e., 58%) of the total features compared with other models. The proposed model can achieve this accuracy for the full CICIDS 2017 dataset. It also detects attacks such as SQL injection and SSH Patator which other models fail to detect. However, it fails to detect Heartbleed and XSS attacks.

Table 13. Accuracy of sub-datasets.

| Sub-Dataset | Accuracy |
|--------------|----------|
| BruteForce | 99.81% |
| DoS/DDoS | 95.50% |
| WebAttacks | 99.48% |
| Infiltration | 99.95% |
| Bot | 99.51% |
| DDoS | 97.83% |
| PortScan | 99.04% |

Table 14 shows the assessment of the proposed technique with other existing CNN models. The accuracy graph is also shown in Figure 12. The CNN-GRU model manages to score a better accuracy rate than models developed by Samson H. et al., 2020 [15] and Sun P. et al., 2020 [17]. The other two CNN models (Priyanka V. et al., 2021 [20] and Maseer Z. et al., 2021 [33]) have slightly higher accuracy scores than our proposed technique. However, these two models were evaluated on the partial dataset. Moreover, we analysed the classification time for each sub-datasets against their sizes and accuracies respectively. We evaluated the techniques using different epochs ranging from 20 to 100 for different sub-datasets. As shown in Figure 13, the classification time for the model is high when the size of the sub-dataset is more. The DoS/DDoS sub-dataset has the highest size among all the sub-datasets. Hence the time taken by the model to classify DoS/DDoS attacks is more compared with other sub-datasets. However, this trend is not the same for all the sub-datasets. Even though their sizes are smaller than those of other sub-datasets, the DDoS and WebAttacks sub-dataset classification times are longer. Figure 14 shows the analysis of the classification time with respect to the accuracy score. The accuracy of the DoS/DDoS sub-dataset is less compared with other sub-datasets. However, due to its size the classification time for the same is higher.

Table 14. Performance of models.

| Model | No. of Attributes | Accuracy |
|------------------|-------------------|----------|
| Proposed CNN-GRU | Less than 44 | 98.73% |
| CNN [18] | 77 | 94.96% |
| CNN [19] | 77 | 99.7% |
| CNN-LSTM [20] | 77 | 98.67% |
| CNN [36] | 77 | 99.47% |

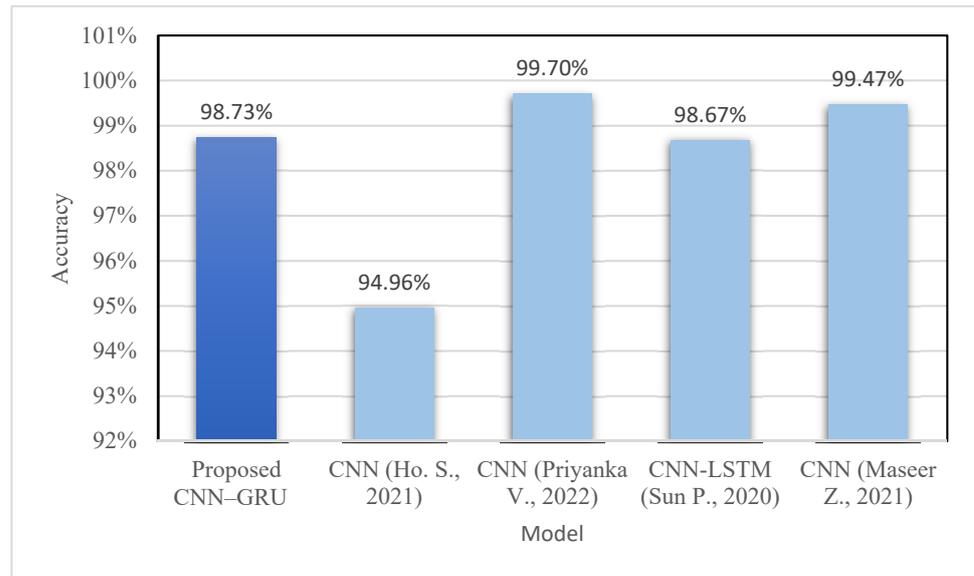


Figure 12. Accuracy graph [18–20,36].

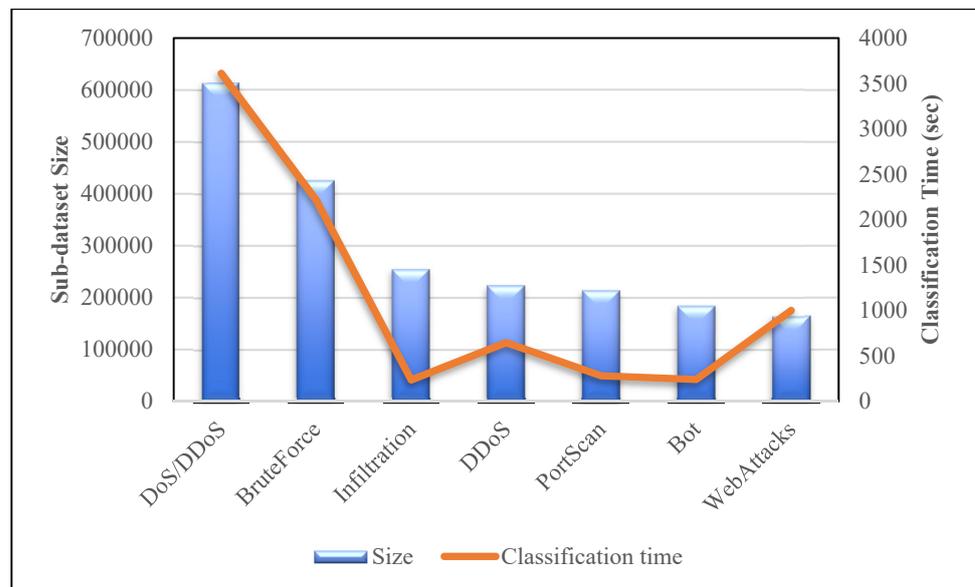


Figure 13. Dataset size vs. classification time.

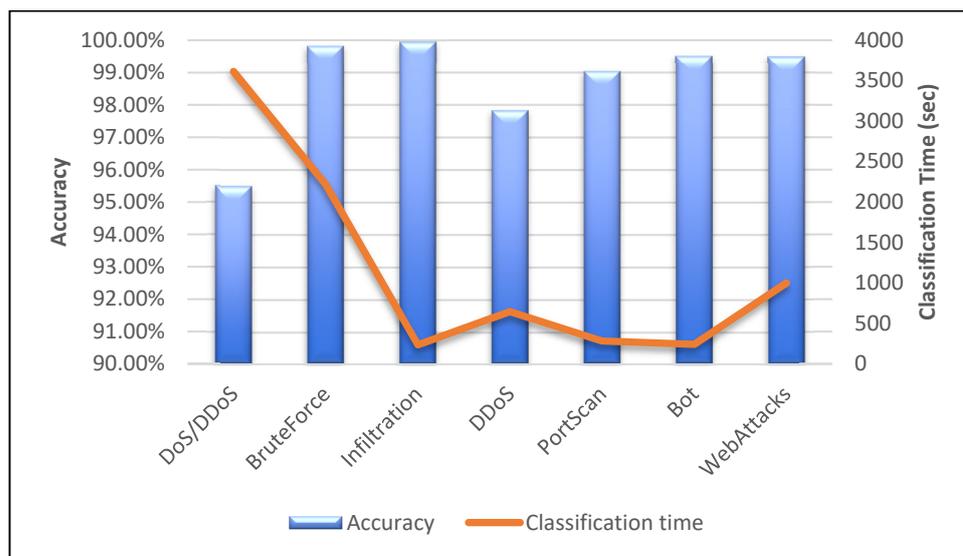


Figure 14. Accuracy vs. classification time.

To achieve optimum accuracy and minimal loss for each sub-dataset in this study, we used a varied number of epochs. The distribution of epochs for the sub-datasets is shown in Table 15. The loss is a result of the model’s inaccurate label predictions. We calculated the loss in our proposed technique using the Sparse Categorical Cross-Entropy technique. The epoch vs accuracy and loss graphs are shown in Figures 15–21.

Table 15. Epochs.

| Sub-Dataset | Epochs |
|--------------|--------|
| BruteForce | 100 |
| DoS/DDoS | 100 |
| WebAttacks | 100 |
| Infiltration | 15 |
| Bot | 20 |
| DDoS | 50 |
| PortScan | 20 |

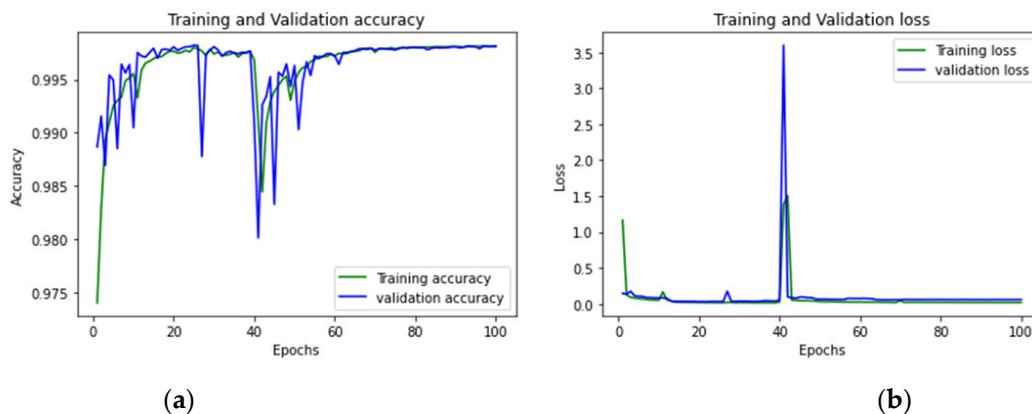


Figure 15. (a,b): Tuesday: Accuracy curve and Loss curve.

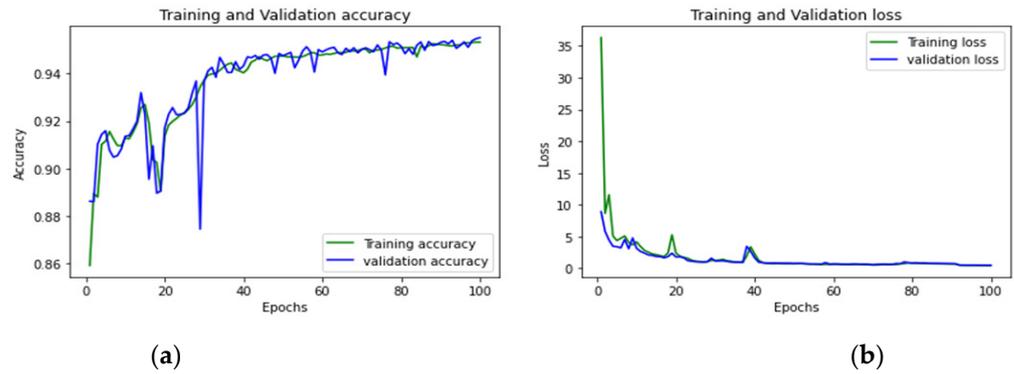


Figure 16. (a,b): Wednesday: Accuracy curve and Loss curve.

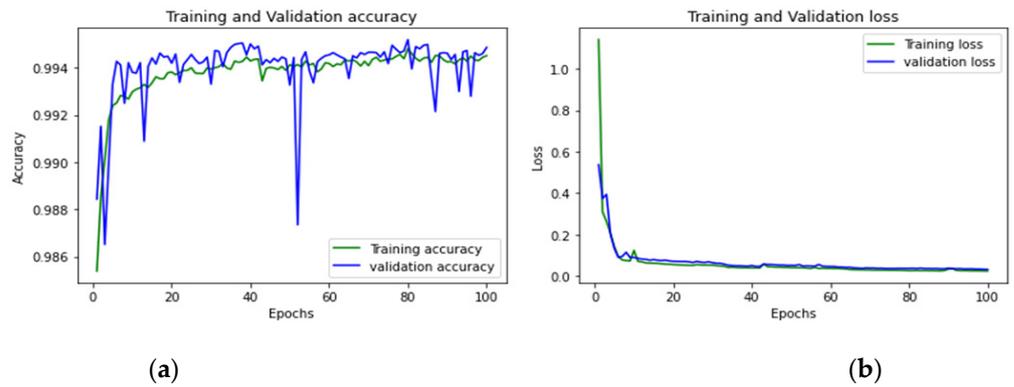


Figure 17. (a,b): Thursday morning: Accuracy curve and Loss curve.

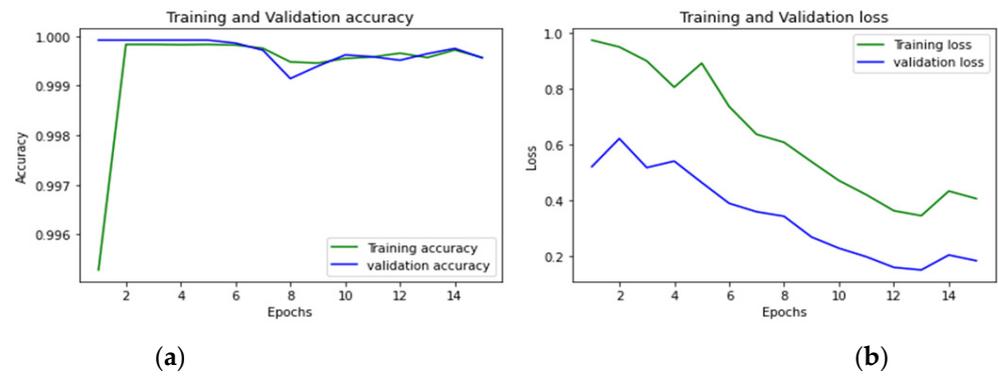


Figure 18. (a,b): Thursday afternoon: Accuracy curve and Loss curve.

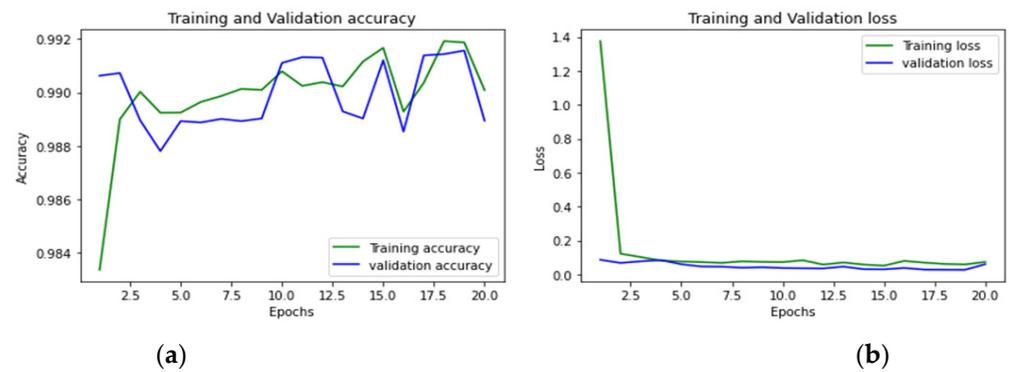


Figure 19. (a,b): Friday morning: Accuracy curve and Loss curve.

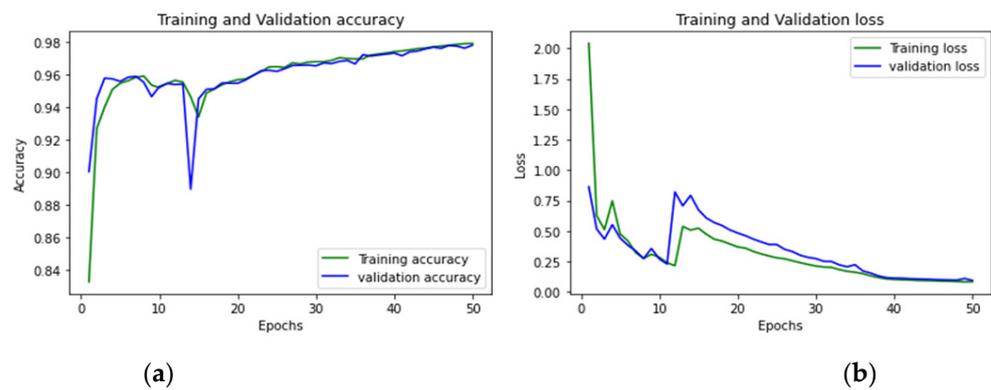


Figure 20. (a,b): Friday afternoon-DDoS: Accuracy curve and Loss curve.

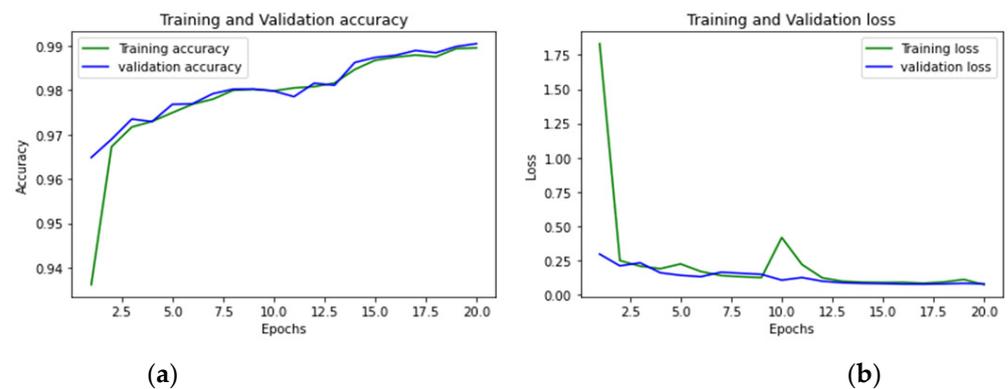


Figure 21. (a,b): Friday afternoon PortScan: Accuracy curve and Loss curve.

5. Conclusions

In this research, we designed a DL technique to develop an IDS. We used CNN with the GRU framework. We used three CNN layers and two GRU sequences. Both sequences use an activation function called ReLU to define the output of the provided inputs. Following that, the concatenate merge mode is used to combine the outputs of both sequences. The dataset considered in this work offers a diversity of attacks and a high number of instances, which was used to verify the suggested model. The dataset contains network data recorded from eight different sessions. In this study, we considered seven sessions and overlooked one session which contains just normal traffic data. We optimized the original dataset by selecting features based on Pearson's Correlation coefficient. We also removed redundant or duplicate instances from the dataset. The evaluation of the model was performed using measurement parameters of the confusion matrix. The suggested model achieved a low FP rate and a classification accuracy rate of 98.73%. The technique used nearly half of the total attributes (less than 58% of the total for each sub-dataset) compared with all other classifiers.

6. Future Scope

The proposed method performed well, but by optimizing the technique even further, performance can be improved. A few attacks were unsuccessfully categorised using the proposed approach. One of the reasons is the imbalanced data of attacks. Therefore, for all the attacks in the dataset, we strive for highly optimal training data in future study. Most classical physical-layer security techniques require CSI, but it is hard to obtain due to time-varying wireless declining [37,38]. As one of the most serious cyber attacks, APT caused global concern. APT is a persistent, multi-stage attack that aims to compromise a system and gain information from it, causing damage and financial loss [39]. The Digital Twin, biometric system, and CPS usher in a new era for commerce, particularly in the health sector, by tracking individuals' health data in order to provide needful, fast, and

efficient services to users [40,41]. IDSs are configured so that they learn from historical network traffic data and detect both normal and abnormal event connections from the monitored system. However, due to the massive amount of historical data, this system may suffer from issues such as accuracy, false alarms, and execution time [42].

Author Contributions: Conceptualization: A.H., S.G., S.K. and P.B.; writing—original draft preparation: A.H., S.G. and S.K.; methodology: S.G., S.K., K.R. and T.S.; writing—review and editing: K.R., T.S., B.S. and S.C.; Software: A.H., S.G. and S.K.; Visualization: K.R., T.S., P.B., B.S. and S.C.; Investigation: T.S., P.B., B.S. and S.C. 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: No data are associated with this research work.

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-to-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. Magán-Carrión, R.; Urda, D.; Díaz-Cano, I.; Dorronsoro, B. Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches. *Appl. Sci.* **2020**, *10*, 1775. [[CrossRef](#)]
11. Meryem, A.; Ouahidi, B.E.L. Hybrid intrusion detection system using machine learning. *Netw. Secur.* **2020**, *2020*, 8–19. [[CrossRef](#)]
12. Abrar, I.; Ayub, Z.; Masoodi, F.; Bamhdi, A.M. A machine learning approach for intrusion detection system on NSL-KDD dataset. In Proceedings of the 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 10–12 September 2020.
13. Alzahrani, A.O.; Alenazi, M.J. Designing a network intrusion detection system based on machine learning for software defined networks. *Future Internet* **2021**, *13*, 111. [[CrossRef](#)]
14. Disha, R.A.; Waheed, S. Performance analysis of machine learning models for intrusion detection system using Gini impurity-based weighted random forest (GIWRF) feature selection technique. *Cybersecurity* **2022**, *5*, 1. [[CrossRef](#)]
15. Megantara, A.A.; Ahmad, T. A hybrid machine learning method for increasing the performance of Network Intrusion Detection Systems. *J. Big Data* **2021**, *8*, 142. [[CrossRef](#)]
16. De Carvalho Bertoli, G.; Pereira Junior, L.A.; Saotome, O.; Dos Santos, A.L.; Verri, F.A.; Marcondes, C.A.; Barbieri, S.; Rodrigues, M.S.; Parente De Oliveira, J.M. An end-to-end framework for machine learning-based network Intrusion Detection System. *IEEE Access* **2021**, *9*, 106790–106805. [[CrossRef](#)]
17. Wang, M.; Zheng, K.; Yang, Y.; Wang, X. An explainable machine learning framework for Intrusion Detection Systems. *IEEE Access* **2020**, *8*, 73127–73141. [[CrossRef](#)]
18. Ho, S.; Jufout SAl Dajani, K.; Mozumdar, M. A Novel Intrusion Detection Model for Detecting Known and Innovative Cyberattacks Using Convolutional Neural Network. *IEEE Open J Comput Soc.* **2021**, *2*, 14–25. [[CrossRef](#)]

19. Priyanka, V.; Gireesh Kumar, T. Performance Assessment of IDS Based on CICIDS-2017 Dataset. In *Information and Communication Technology for Competitive Strategies (ICTCS 2020)*; Lecture Notes in Networks and Systems; Joshi, A., Mahmud, M., Ragel, R.G., Thakur, N.V., Eds.; Springer: Singapore, 2022; Volume 191.
20. Sun, P.; Liu, P.; Li, Q.; Liu, C.; Lu, X.; Hao, R.; Chen, J. DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system. *Secur. Commun Netw.* **2020**, *2020*, 8890306. [[CrossRef](#)]
21. Mauro, M.D.; Galatro, G.; Liotta, A. Experimental Review of Neural-based approaches for network intrusion management. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 2480–2495. [[CrossRef](#)]
22. Dong, S.; Xia, Y.; Peng, T. Network abnormal traffic detection model based on semi-supervised Deep Reinforcement Learning. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 4197–4212. [[CrossRef](#)]
23. Pelletier, C.; Webb, G.I.; Petitjean, F. Deep learning for the classification of sentinel-2 Image time series. In Proceedings of the IGARSS 2019—2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 28 July–2 August 2019.
24. Lee, J.; Pak, J.G.; Lee, M. Network intrusion detection system using feature extraction based on deep sparse autoencoder. In Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 21–23 October 2020.
25. Injadat, M.N.; Moubayed, A.; Nassif, A.B.; Shami, A. Multi-stage optimized machine learning framework for network intrusion detection. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 1803–1816. [[CrossRef](#)]
26. Zhu, H.; You, X.; Liu, S. Multiple Ant Colony Optimization Based on Pearson Correlation Coefficient. *IEEE Access* **2019**, *7*, 61628–61638. [[CrossRef](#)]
27. 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](#)]
28. 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](#)]
29. 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](#)]
30. Handwritten, I.; Recognition, D. Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN). *Sensors* **2020**, *20*, 3344.
31. 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](#)]
32. 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](#)]
33. Sheba, K.; Raj, S.G. An approach for automatic lesion detection in mammograms. *Cogent Eng.* **2018**, *5*, 1444320. [[CrossRef](#)]
34. Wahlberg, F.; Dahllöf, M.; Mårtensson, L.; Brun, A. Spotting Words in Medieval Manuscripts. *Stud. Neophilol.* **2014**, *86* (Suppl. S1), 171–186. [[CrossRef](#)]
35. 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](#)]
36. Maseer, Z.K.; Yusof, R.; Bahaman, N.; Mostafa, S.A.; Foozy, C.F.M. Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset. *IEEE Access* **2021**, *9*, 22351–22370. [[CrossRef](#)]
37. Deng, D.; Li, X.; Zhao, M.; Rabie, K.M.; Kharel, R. Deep Learning-Based Secure MIMO Communications with Imperfect CSI for Heterogeneous Networks. *Sensors* **2020**, *20*, 1730. [[CrossRef](#)] [[PubMed](#)]
38. Gupta, K.; Gupta, D.; Kukreja, V.; Kaushik, V. Fog Computing and Its Security Challenges. In *Machine Learning for Edge Computing*; CRC Press: Boca Raton, FL, USA, 2022; pp. 1–24.
39. Ghafir, I.; Hammoudeh, M.; Prenosil, V.; Han, L.; Hegarty, R.; Rabie, K.; Aparicio-Navarro, F.J. Detection of advanced persistent threat using machine-learning correlation analysis. *Future Gener. Comput. Syst.* **2018**, *89*, 349–359. [[CrossRef](#)]
40. Garg, H.; Sharma, B.; Shekhar, S.; Agarwal, R. Spoofing detection system for e-health digital twin using EfficientNet Convolution Neural Network. *Multimed. Tools Appl.* **2022**, *81*, 26873–26888. [[CrossRef](#)]
41. Datta, P.; Bhardwaj, S.; Panda, S.N.; Tanwar, S.; Badotra, S. Survey of security and privacy issues on biometric system. In *Handbook of Computer Networks and Cyber Security*; Springer: Cham, Switzerland, 2020; pp. 763–776.
42. Garg, S.; Singh, R.; Obaidat, M.S.; Bhalla, V.K.; Sharma, B. Statistical vertical reduction-based data abridging technique for big network traffic dataset. *Int. J. Commun. Syst.* **2020**, *33*, e4249. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.