IJEMD-CSAI, 4 (1) (2025) https://doi.org/10.54938/ijemdcsai.2025.04.1.400

International Journal of Emerging Multidisciplinaries:
Computer Science and Artificial Intelligence

Research Paper
Journal Homepage: www.ojs.ijemd.com
ISSN (print): 2791-0164 ISSN (online): 2957-5036

Anomaly Detection Via Network Intrusion Using A Hybrid CNN And LSTM

Jesse Mazadu Ismaila [1]* , Victoria Zevini Sabo [1]

1. Federal University Wukari, Nigeria.

Abstract

The challenge posed by the measurement of quality of service (QOS) in network environments has become increasingly critical, especially in light of the constantly evolving land scape of cybersecurity threats. While various research endeavors have utilized metrices such as the false positive rate to evaluate the effectiveness of intrusion Detection systems (IDSs) in safeguarding network integrity the strategies employed by malicious actors often prioritize disrupting network and consuming resources. This problem has stimulated to undergo intensive research to develop a hybrid convolutional neural Network and Long Short-Term Memory algorithm for network anomaly intrusion detection. To evaluate the performance of the model, the precision, recall, and f1-score evaluation metrics was applied. The process of hybridization involves stacking the output of the CNN layers with the LSTM layer. The experiment revealed that network anomaly detection classifier shows a significant performance for data split of 70:30 training to testing. LSTM-CNN model was demonstrated with exceptional performance, achieving an accuracy of 99.99% after only 2 epochs. This result indicates the robustness and efficiency of the proposed hybrid mode in detecting network intrusion.

Keyword: Anomaly detection, Signature detection, Heuristic, False negative, Intrusion Detection System.

INTRODUCTION

Background of the Study

The field of Information Technology (IT) has witnessed significant progress in recent times, leading to the widespread adoption of cloud computing. This has resulted in a substantial increase in the amount of data being generated and transmitted across various internet-connected devices, growing at an exponential

Email Addresses: jesse@fuwukari.edu (Jesse) , victoriasabo@fuwukari.edu.ng (Victoria).

rate [1]. Nevertheless, as computer networks continue to gain prominence in modern society, there is a concomitant escalation in the risks and vulnerabilities connected with them. This is primarily due to the expanding pool of potential infiltration threats and the increasing level of sophistication exhibited by these threats [2]. The current state of network-dependent communication renders it vulnerable to both external and internal threats. The task of monitoring incoming traffic has significant challenges due to the substantial volume of traffic and the prevalence of attacks, resulting in increased expenditures of time and financial resources in computational processes [3]. These threats are quite challenging and hence require the implementation of intrusion detection systems to identify and mitigate various possible dangers.

Intrusion detection systems (IDSs) is the software that analyses real-time network traffic and reports any odd behaviour going over the network [3]. Intrusion detection systems (IDSs) are sophisticated technologies used on the Internet to recognize the unusual behaviour of attackers based on their destructive activities. Furthermore, an IDS is commonly used for the classification of network traffic to find anomalies inside the network. In the realm of intrusion detection systems (IDSs), there exists a classification framework consisting of four distinct categories. These categories include IDSs that are signature-based, anomaly-based, specification-based, and hybrid approach-based [4]. In Intrusion Detection Systems (IDS) that rely on signatures, the detection of each assault is contingent upon the description of the attack pattern. Within this framework, a trigger message is generated when the attack aligns with the pattern that has been provided. By employing this particular Intrusion Detection System (IDS), the detection of well-documented attack methodologies can be significantly enhanced. The subsequent classification pertains to intrusion detection systems that operate based on anomalies. In this framework, information regarding the typical functioning of the devices is acquired and parameters are established. When the device's behaviour deviates from expected norms, the Intrusion Detection Systems (IDSs) classify it as suspicious behaviour.

The anomaly IDS involve the use of location and temporal constraints to identify malicious devices [5]. In the realm of anomaly IDS attacks such as DoS attacks, Sybil attacks, selective forwarding attacks, worm whole attacks, black hole attacks, sinkhole attacks, jamming attacks, and false data injection attacks as some of the major forms of anomaly attacks [4]. It is essential to note that IDSs must be developed not only to detect the known types of malicious attacks but also to detect the new types of attacks carried out on the data communicated through the Internet using different approaches. However, the deep learning technique is adaptive, enabling models to identify known and unknown forms of attacks. This has prompted this study to harness the viabilities of the deep learning techniques and hence hybridise the LSTM and CNN network. Here, the hybrid model would harness the convolutional capabilities of CNNs to extract relevant spatial features from the raw network traffic data and these extracted features could then be seamlessly input into the LSTM layer, which excels at capturing temporal dependencies in sequential data. Hence, to evaluate the viabilities of the hybrid model on anomaly intrusion detection, the NSL- KDD (National Science Foundation - Knowledge Discovery Dataset) was utilized.

The challenge posed by the measurement of Quality of Service (QoS) in network environments has become increasingly critical, especially in light of the constantly evolving landscape of cybersecurity threats. While various research endeavours have utilized metrics such as the false positive rate to evaluate the effectiveness of Intrusion Detection Systems (IDSs) in safeguarding network integrity, the strategies employed by malicious actors often prioritize disrupting networks and consuming resources [4]. This approach aims to impede legitimate users' access to essential network resources. An additional challenge pertains to the adaptability of conventional machine learning models in the face of these dynamic threats and evolving network conditions. These models may struggle to adequately capture the intricacies of rapidly changing attack patterns and novel attack vectors. Consequently, there arises a necessity to explore innovative approaches that can accommodate the evolving nature of network attacks and adapt to emerging threats in real-time. One potential avenue for addressing these challenges as proposed by this study involves the utilization of a hybrid model that seamlessly integrates the capabilities of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. By combining the strengths of both CNN and LSTM architectures, it becomes possible to enhance the processing and analysis of network data. This, in turn, enables a more comprehensive evaluation of network performance and security, as well as the accurate measurement of QoS. The aim of the study is to develop a hybrid Convolutional Neural Network and Long-Short Term Memory algorithm for network intrusion detection. The specific objectives are: perform data preprocessing using the robust scaler, apply and hybridize the convolutional neural network and Long Short-Term Memory for network anomaly detection and evaluate the performance of the algorithm.

LITERATURE REVIEW

Theoretical Review

Intrusion detection systems (IDS) refer to security mechanisms designed to identify and respond to unauthorized access attempts or malicious activities within a computer network or system. An attack can be characterized as an action that contravenes a network security policy. Intrusion is defined as any endeavour to establish or acquire unauthorized access to the information of an individual or organization. An Intrusion Detection System (IDS) refers to a combination of software and hardware components that are employed to identify and detect instances of unauthorized access or malicious activities targeting computer systems or telecommunications networks. To mitigate the risk of unauthorized access to sensitive information, it is necessary to implement certain security measures. The requirement for a proficient system that can accurately categorize communications as either malicious or benign arises [6]. Intrusion Detection Systems possess the capability to perform this task with a high degree of accuracy and effectiveness. In accordance with performance, intrusion detection systems (IDS) can be classified into two distinct categories: Signature-Based IDS (SIDS) and Anomaly-Based IDS (AIDS) [7]. The Intrusion Detection System (IDS) that relies on Signature-based detection is a form of intrusion detection mechanism that identifies attacks by comparing patterns with previously documented attack patterns. Matching approaches are employed in the context of Signature-Based IDS (SIDS) to ascertain a prior occurrence of incursion. As mentioned earlier, the detection of a signature that matches one of the pre-existing signatures results in the activation of an alert signal. One limitation of the signature-based technique is its inability to identify novel assaults, necessitating regular updates.

AIDS research encompasses two distinct phases, namely the training phase and the testing phase. During the training phase of the experiment, a model is constructed to represent the typical behavior of traffic. Subsequently, in the testing phase, a novel dataset is employed to assess the validity of the model. AIDS can be classified based on the instructional approach employed, such as statistical, knowledge-based, and machine-learning methods [8]. One significant advantage of AIDS is its ability to detect zero-day attacks without relying on the identification of anomalous user behaviours. The system triggers a warning signal for actions that are in a state of disorder [9]. Anomaly-Based IDS (AIDS) encompasses a range of potential consequences. These systems have the capability to identify covert cyber-security vulnerabilities. Hence, given the limitations of SIDS in effectively identifying novel assaults, the present study employs an AIDS methodology.

Related Work

As showed interestingly by [9], a Stacked Auto Encoder (SAE) as feature extraction for the most common KDD99 dataset. He found that even though the KDD99 dataset provides 41 distinct features, it's not necessarily that those 41 features are important to classify benign and anomaly packets. His motivation is to find the most important features that affect the classification results significantly. He assumed that upon knowing the important features, one can focus on these features only, also called feature selection. Besides that, one can also use different portions/weights in different features to get better detection results. There are several advantages to his approach. First, it reduces the researcher's workload to identify the relationship in the dataset. Second, dimensionality reduction is accomplished since features are mapped into new space. In addition, the redundant data can be filtered out.

[10] proposed a Network Intrusion Detection System that utilized a self-taught learning approach. The researchers employed a sparse autoencoder and soft-max regression in order to assess their model. The proposed model was trained using the NSL-KDD dataset, yielding an accuracy of approximately 79.10% for 5-class classification. This performance is comparable to that of other state-of-the-art models. In addition, the author explicitly stated that both the 23-class and 2-class categorization methods had a favorable performance metric. [11] on intrusion detection for wireless sensor networks, put forward a comprehensive approach for developing an anomaly-based Intrusion Detection System (IDS) called MIDS. This system utilizes the Binary Logistic Regression (BLR) statistical technique to categorize the local sensor activity as either benign or malicious. The suggested system was assessed by the authors through the examination of routing layer attacks. The findings indicate that the malicious Intrusion Detection System (mIDS) successfully identified malevolent actions with an accuracy ranging from 88% to 100%.

[12] sought to enhance the efficiency of a Network Intrusion Detection System (NIDS) by optimizing the performance of a Recurrent Neural Network (RNN) based approach. Additionally, the researchers conducted model training on the NSL-KDD dataset, evaluating both binary and multi-class classification tasks. The performance of the Recurrent Neural Network (RNN) in the context of the Intrusion Detection System (IDS) exhibited superiority in both classification tasks when compared to conventional techniques. The authors assert that the utilization of Recurrent Neural Networks (RNN) in Intrusion Detection Systems (IDS) exhibits a robust modelling capability, in contrast to the recommended Software-Defined

Networking (SDN) environment. Therefore, the model developed by the author was only trained using the six fundamental characteristics extracted from the NSL-KDD dataset, employing various learning rates. As a result, the model attains a peak accuracy of 75.75% [13].

A novel semi-supervised learning method for enhancing the intrusion detection system through the utilization of a Fuzziness based methodology by [14] employed a supervised learning approach to enhance the performance of the IDSs by optimizing the utilization of unlabeled samples. The authors employed a single hidden layer feed-forward neural network (SLFN) to train and generate a fuzzy membership vector. Subsequently, the authors utilized the fuzzy quantity to categorize unlabeled samples into low, mid, and high fuzziness categories. The classifier underwent retraining by individually adding each category into the original training set. The NSL-KDD dataset was utilized by the author for the purpose of their experiment. The findings obtained from the author's methodology for intrusion detection on the NSL-KDD dataset indicate that unlabeled samples from both low and high fuzziness groups significantly enhance the performance of the classifier in comparison to established classifiers such as naive Bayes, support vector machine, random forests, and others.

[15] put out a novel approach known as the stacked Non-Symmetric Deep Autoencoder (NDAE) to enhance the performance of Network Intrusion Detection Systems. The model has been trained on both the KDD Cup 99 and NSLKDD benchmark datasets, and its performance is being compared with a model based on Deep Belief Network (DBN). The authors have noticed, both experimentally and analytically, that the NDAE based strategy demonstrates an improvement in accuracy of up to 5% when compared to the Deep Belief Network (DBN) based approach. Additionally, the NDAE technique also shows a significant reduction in training time, namely by 98.8%.

In a study, detection technique for network attacks called the Epigenetic Algorithm-Based Detection Technique was used. This technique utilizes an Epigenetic-based algorithm (EGA), which is an enhanced version of the Genetic Algorithm (GA) that offers improved performance with reduced computational complexity. The primary objective of EGA is to quickly converge towards an optimal solution by employing genetic operators, specifically mutation and crossover. The authors applied EGA to classify attacks using a database of network traffic. Through simulation, the results demonstrate that the proposed technique outperforms the GA classifier in terms of detection rate, achieving up to 98%, and exhibits faster processing time compared to GA and other algorithms evaluated by the authors [16].

In the study on intrusion Detection Systems presented a comprehensive analysis of the relevance of the features in the KDD99 and UNSW-NB15 datasets was done. The authors employed: a rough-set theory (RST), a back-propagation neural network (BPNN), and a discrete variant of the cuttlefish algorithm (D-CFA). First, the dependency ratio between the features and the classes was calculated, using the RST. Second, each feature in the datasets became an input for the BPNN, to measure their ability for a classification task concerning each class. Third, a feature-selection process was carried out over multiple runs, to indicate the frequency of the selection of each feature. Their experimental result indicated that some features in the KDD99 dataset could be used to achieve a classification accuracy above 84%. Moreover, a few features in both datasets were found to give a high contribution to increasing the classification’s performance. These features were present in a combination of features that resulted in high accuracy; the features were also frequently selected during the feature selection process by the study [16].

Developed by [17], a system for the detection and prevention of cyber-attacks using Artificial Intelligence Methods depicts a multiple, large and hybrid DNN torrent structure called Scale-Hybrid-IDS-AlertNet. Which can be used to effectively monitor, detect, and review the impact of network traffic and host-level events to warn directly or indirectly about cyber-attacks. Besides this, their implementations are highly adaptable and flexible, with commensurate efficiency and accuracy when it comes to the detection and prevention of cyberattacks as claimed by the authors. [18] introduced the gradient hybrid leader optimization (GHLBO) method as a means of detecting DDoS attacks. The present optimized approach is tasked with training a deep stacking autoencoder (DSA) to effectively identify and detect attacks. In this study, the fusion of features is performed using a deep maxout network (DMN) incorporating an overlap coefficient. Additionally, data augmentation is conducted through the oversampling technique. Moreover, the generation of GHLBO is achieved through the integration of the gradient descent and hybrid leader-based optimization (HLBO) algorithm. The evaluation of the author's work was conducted using many performance measures, including the true positive rate (TPR), the true negative rate (TNR), and testing accuracy. The corresponding values obtained were 0.909, 0.909, and 0.917, respectively.

RESEARCH METHODOLOGY

Introduction

This section presents a comprehensive explanation of the research methodology utilized. Furthermore, it thoroughly discusses all the experimental methods employed to construct the proposed network intrusion models. It also offers an understanding of the suggested dataset, the technique used for data preprocessing and feature selection, a description of the proposed machine learning algorithm, and, most importantly, the performance evaluation metrics employed to validate the performance of each model.

Methodology Overview

When engaging in research, it is crucial to adapt and create the technique following engineering practice. This is necessary to ensure the efficient development or evaluation of the project's feasibility. A methodology refers to a systematic strategy or technique employed to analyse and present the desired outcome. The building of a machine learning model for the classification of network intrusion abnormalities is challenging and requires an effective methodology. Hence, to construct the envisioned machine learning model, this research has implemented a methodological approach consisting of three distinct phases. The initial stage involves accessing the dataset through the use of Panda's library, which offers many approaches for reading datasets in different file formats. In this study, the dataset format chosen is comma-separated values (CSV). The second phase of the procedure involves data preparation, which aims to eliminate irrelevant sounds from the dataset. In the data preprocessing stage, the robust scaler and label encoders will be utilised to standardize and encode the dataset's labels and features. This will be done before the selection of pertinent features through the examination of the correlation matrix table. In the final phase, the filtered signals are inputted into the designated deep learning models, specifically the convolutional neural network and the long short-term memory. Before this, the dataset is divided into a 70:30 ratio for training and testing purposes. The training data is utilized to train the two deep network models, while the test data is employed to assess the accuracy of the model. The third part of the study involves the integration of a performance evaluation scheme to assess the efficacy of various performance models and facilitate comparative analysis. The sequential methodology for executing the suggested network intrusion model is depicted in Figure 3.1.

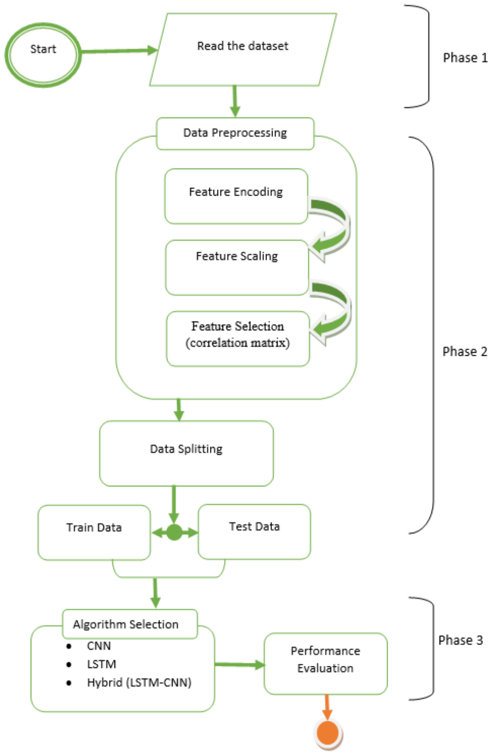
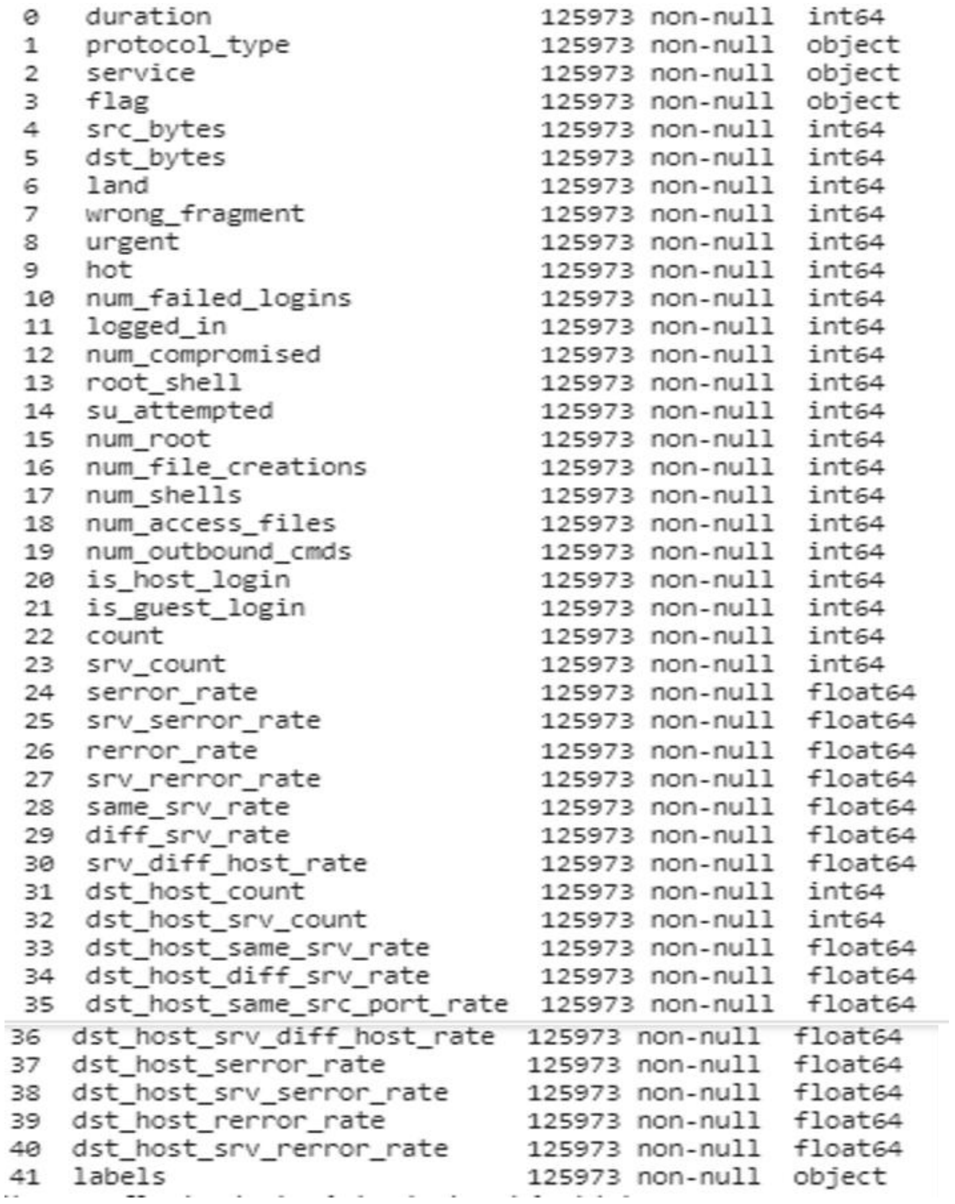


Figure 3.1: Framework for the Research Methodology

Algorithm 3.1 Working process of the proposed methodology
Step 1: Start. **Step 2:** Intrusion detection data is an input. **Step 3:** Apply preprocessing in the IDS dataset, and one hot-encoding and normalization operation are performed**.** **Step 4:** Symbolic features in the dataset are converted into numerical features using the one-hot approach. **Step 5:** Normalization processing removes the measurement unit's influence from the model training and increases the reliance on the training outcome. **Step 6:** CNN and LSTM are employed to classify the various attack categories in NIDS. **Step 7:** Performance Evaluation **Step 8:** End.

Dataset Descriptions

The NSL-KDD dataset was employed to assess the efficacy of intrusion detection and learning systems. Derived from the KDD Cup 99 dataset, the NSL-KDD dataset has been widely adopted by researchers, for IIoT intrusion detection purposes. It comprises 125,973 records for training and 22,544 for testing, encompassing 41 features. The NSL-KDD data was obtained from GitHub via the link https://www.kaggle.com/datasets/hassan06/nslkdd. The dataset includes a target feature labeled as "labels," where 0 denotes non-threat (normal) and 1 indicates a threat (attack). Figure 3.2 illustrates the attributes in the dataset along with their respective data types.



```
0   duration                     125973 non-null  int64
1   protocol_type                125973 non-null  object
2   service                      125973 non-null  object
3   flag                         125973 non-null  object
4   src_bytes                    125973 non-null  int64
5   dst_bytes                    125973 non-null  int64
6   land                         125973 non-null  int64
7   wrong_fragment               125973 non-null  int64
8   urgent                       125973 non-null  int64
9   hot                          125973 non-null  int64
10  num_failed_logins            125973 non-null  int64
11  logged_in                    125973 non-null  int64
12  num_compromised              125973 non-null  int64
13  root_shell                   125973 non-null  int64
14  su_attempted                 125973 non-null  int64
15  num_root                     125973 non-null  int64
16  num_file_creations           125973 non-null  int64
17  num_shells                   125973 non-null  int64
18  num_access_files             125973 non-null  int64
19  num_outbound_cmds            125973 non-null  int64
20  is_host_login                125973 non-null  int64
21  is_guest_login               125973 non-null  int64
22  count                        125973 non-null  int64
23  srv_count                    125973 non-null  int64
24  serror_rate                  125973 non-null  float64
25  srv_serror_rate              125973 non-null  float64
26  rerror_rate                  125973 non-null  float64
27  srv_rerror_rate              125973 non-null  float64
28  same_srv_rate                125973 non-null  float64
29  diff_srv_rate                125973 non-null  float64
30  srv_diff_host_rate           125973 non-null  float64
31  dst_host_count               125973 non-null  int64
32  dst_host_srv_count           125973 non-null  int64
33  dst_host_same_srv_rate       125973 non-null  float64
34  dst_host_diff_srv_rate       125973 non-null  float64
35  dst_host_same_src_port_rate  125973 non-null  float64
36  dst_host_srv_diff_host_rate  125973 non-null  float64
37  dst_host_serror_rate         125973 non-null  float64
38  dst_host_srv_serror_rate     125973 non-null  float64
39  dst_host_rerror_rate         125973 non-null  float64
40  dst_host_srv_rerror_rate     125973 non-null  float64
41  labels                       125973 non-null  object
```

Figure 3.2: Data attribute and data type of NSL-KDD dataset

Experimental Tools

For effective implementation of the proposed network intrusion detection model via the convolutional neural network and the long short-term memory algorithms, this study proposes the utilization of the following programming language and its applicable libraries:

- **i.** Visual studio code is the development environment for implementing the program codes.
- **ii.** A Python software development kit for the integration and development of the model's source code.
- **iii.** TensorFlow API, NumPy, Scikit-Learn, Matplotlib, and Pandas as the dependencies support.

3.5 Dataset Preprocessing

The procedure of data preparation is a crucial element in the execution of any machine learning approach. The main aim of data preparation in this study is to improve the quality and suitability of the data for the specific data mining task. This study introduces a proposed approach for data preparation, encompassing a sequence of phases designed to efficiently handle the data. The steps involved in this process encompass data cleansing (removal of missing cells), data standardization/normalization, feature selection, and data splitting.

3.5.1 ***Normalization:*** The features in the NSL-KDD dataset consist of either continuous or discrete values. The values had disparate ranges, rendering them non-comparable. The normalization process involved removing the mean from each feature and dividing it by its standard deviation for both the training and test datasets. The mean and standard deviation of each feature in the training dataset were used to normalize the test features. The Min-Max normalization method, which involves a linear transformation, is employed to rescale data within the range of (0,1). The subsequent procedure is employed to normalize the dataset values:

$$X_n = \frac{X - X_{min}}{X_{man} - X_{min}} \dots\dots\dots\dots\dots\dots 3.1$$

Feature Selection

Dimensionality reduction is a crucial component of machine learning that facilitates the elimination of duplicate features. Given the complexity of the dataset, it is imperative to leverage the effectiveness of a feature selection technique. Therefore, the study uses the correlation matrix as a method for selecting features. The use of a correlation matrix allows for the identification of strongly correlated features, hence aiding in the detection of redundant or overlapping information. The coefficient of the correlation matrix quantifies the strength of the association between different features and suggests which features should be eliminated.

Classification Model

From the dataset adapted, it was identified that the problem is a classification problem, where the attacks are classified into several labels. Essential classification involves procedures where the models try to predict the correct label of a given input data considered as network attacks. Hence, for the identification and classification of network intrusion, the study proposed the application of two machine learning models including the CNN and LSTM, while further hybridizing the aforementioned models.

Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) operates based on principles derived from the visual cortex of animals, which allows it to effectively capture sequential relationships in data. In the context of intrusion detection, a one-dimensional CNN (1D-CNN) is employed to analyze textual information for identifying patterns associated with network intrusions. The architecture of a 1D-CNN typically includes key components such as a convolutional layer, a pooling layer, and a fully-connected layer. The proposed 1D-CNN model follows this architectural paradigm.

The first step involves applying a 1D convolutional layer to the input data, where a filter is slid over the scaled feature representation of the text. This convolutional layer serves as a fundamental building block of the CNN and segments the input into partially overlapping regions. Through convolution operations, these regions are analyzed, mimicking the response of cortical neurons to visual stimuli, but in this case, applied to textual data.

Following the convolutional layer, the activation function Rectified Linear Unit (ReLU), denoted as $(x) = \max(0, x)$, is applied to introduce non-linearity into the model. This non-linear activation function helps the CNN capture complex patterns and relationships within the data, making it effective for tasks like intrusion detection where identifying subtle and intricate patterns in network traffic is crucial for accurate detection.

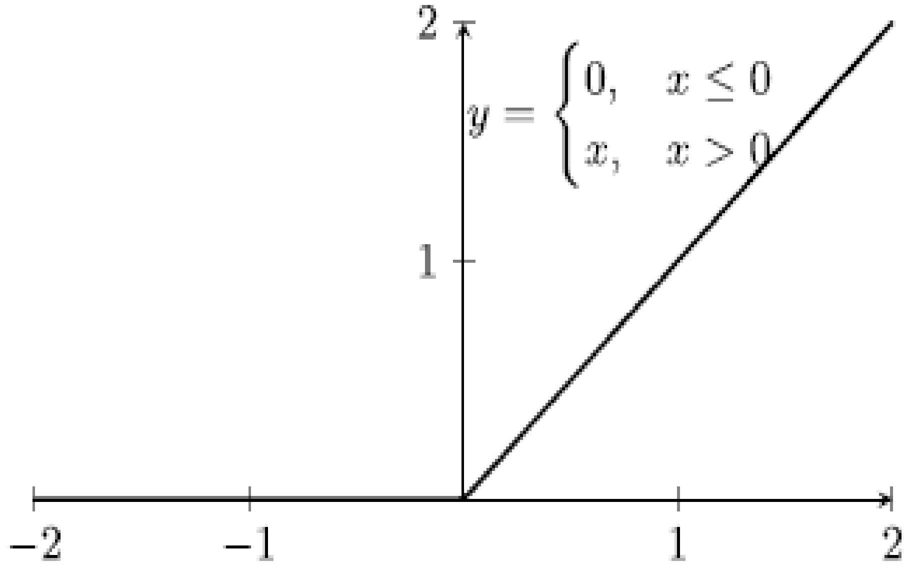


Figure 3.3: A visualization of the ReLU function $(x) = \max(0, x)$

For CNN, the study proposed the use of Adam as the optimizer and sparse categorical entropy as the loss function. Below is an algorithm for the adapted Convolutional Neural Network.

Algorithm 3.2: convolution neural network pseudocode

```
Input: $x = [x_1, x_2, \dots x_n]$
Function CNN_MODEL(x)
weights ← DEFINE WEIGHT
biases ← DEFINE BIASES
FOR x to n (no of layers) DO
    x ← RESHAPE (x, shape ←(x-dim, x-length, ...)
    conv2 ← RELU_ACTIVATION_FUNC (conv3D (x, weight (0) + biases (0))
    conv2 ← DROPOUT (0.5)
    conv2 ← MAX_POOL3D (conv2)
END FOR
fully-connect ← RESHAPE (conv2 INVERSE (weights))
fully-connect ← RELU_ACTIVATION_FUNC (fully-connect + weight + biases)
fully-connect ← DROPOUT (0.5)
output ← fully-connect * weights + biases
RETURN output
END FUNCTION
```

LSTM Neural Network

The Long Short-Term Memory (LSTM) cell, a breakthrough in recurrent neural networks (RNNs), addresses the vanishing gradient problem that affects traditional RNNs [19]. This issue hinders conventional RNNs from effectively capturing long-term dependencies in data, as they prioritize current information over past events. LSTM, in contrast, offers precise control over information flow within its neurons by employing a gating mechanism. This mechanism regulates the addition and removal of information from a continuously evolving cell state, enabling LSTM to model both short-term and long-term dependencies effectively. In classification tasks, the LSTM neural network is a popular choice. It incorporates an LSTM layer and a mean-pooling layer with fully connected input layers. This architecture is well-suited for processing groups of data and explicitly designed to learn long-term dependencies. LSTM's gates are specifically designed to store data for extended periods. These gates utilize input data, output from the previous time step, bias terms, and time intervals to calculate activation values. The sigmoid function then scales these values between 0 and 1, allowing the LSTM to manage and store information effectively for classification tasks. The stepwise procedures for the adapted Long Short-Term Memory algorithm are shown in Algorithm 3.3.

Algorithms 3.3: long short-term memory pseudocode

```
Input: Dataset
Begin
If (Dataset is valid)
        Extract Features
Else
        Collect Valid Dataset
DfTrain = Processing and MT Model Training
If (MT Model Train)
        Not Train
Else
        Perform Preprocessing & MT
DfLabel = check DfTrain Dataset length
DfTarget = Apply Label Dataset (0 Spam, 1 Not Spam)
MissingValue MV = Target Dataset check

If (TD is valid) #TD = Target Data
        Text processing for cleaning Data
Else
         Not valid
Train & Test = Split data in Train & Test (X_Train, Y_Train, X_Test, Y_Test)
If (Feature vector Fv & Target Variable Tv is valid)
          Data Convert into Train TS & Test TS
Else
          Error Occur
Model Selection = Select Model for Classification (LSTM)
LSTM = Apply fitting LSTM Model
TS = Compare Train & Test Set
   If (Train & Test TS is Accurate)
             Accuracy Result Show
Else
             Error Occur
CM = Classification Metrics
If (CM Process is valid)
        Show classification Report
Else
        Error Occur in Classification
End
```

Hybridization Process

The process of hybridizing Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) for a classification problem in network intrusion detection involves converting the sequential network traffic data into a format suitable for input into both CNN and LSTM models. This essentially involve representing the data as time-series sequences or as images if using CNNs. The architecture design entail designing the CNN architecture to extract spatial features from the input data where the CNN layers consist of convolutional layers followed by pooling layers to capture important patterns in the data. The LSTM architecture was then used to capture temporal dependencies in the sequential data as the LSTM layers are effective at retaining long-term dependencies in time-series data. In summary, the combination of the CNN and LSTM components into a unified architecture was achieved by feeding the output of the CNN component into the LSTM component, allowing the LSTM to further process the extracted features.

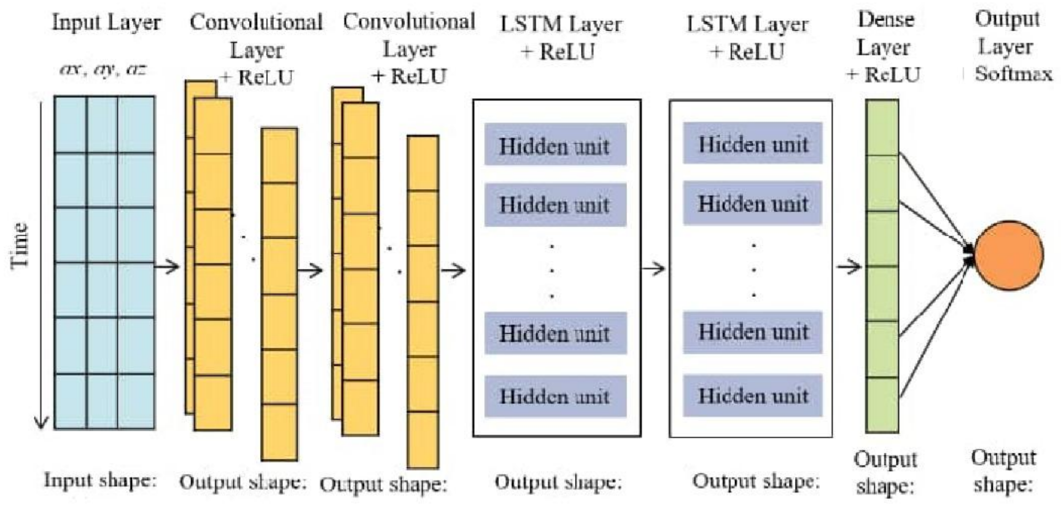


Figure 3.4: Process of Hybridization

Performance Evaluation Metrics

For assessing the efficiency level of the proposed model, the following metrics employ several features. These metrics are true positive (TP) or Recall (R), true negative (TN), false positive (FP) and false negative (FN).

True Positive Ratio (TPR) or Recall (R): TPR or Recall for a certain class, is the number of samples in the class that is correctly predicted, divided by the total number of samples in the class. Formula 3 shows this relation.

$$TPR = \frac{TP}{TP + FN} \dots\dots\dots\dots\dots\dots (3.3)$$

False positive rate (FPR) is measured for estimating the quantity of the attack data that is identified as normal data. It is calculated as follows:

$$FPR = \frac{FP}{FP + TN} \dots\dots\dots\dots\dots\dots (3.4)$$

False negative rate (FNR) is measured for estimating the quantity of the normal data that is identified as being attack data. It is calculated as follows:

$$FRN = \frac{FN}{FN + TP} \dots\dots\dots\dots\dots\dots (3.5)$$

Accuracy is represented in a percentage. It refers to the degree to which the instances are predicted correctly. It is calculated as follows:

$$Accuracy = \frac{TP + FN}{TP + FP + TN + FN} \dots\dots\dots\dots\dots\dots (3.6)$$

Precision is represented by the ratio of the number of decisions that are considered correct. It is represented in the TP divided by the sum of FP and TP. It is calculated as follows:

$$Precision = \frac{TP}{TP + FP} \dots\dots\dots\dots\dots\dots\dots\dots 3.7$$

RESULT AND DISCUSSION

Introduction

This chapter presents the results obtained from the practical exploration and implementation related to predicting network intrusion detection. The research methodology involves employing deep learning algorithms, specifically the LSTM and CNN algorithms. Consequently, the effectiveness of the constructed models undergoes a thorough validation process, evaluating performance metrics such as precision, recall, and F1-score.

Parameter Setting

The parameter settings for the proposed LSTM and CNN model are presented in Table 4.1. considering the batch size which determines the number of samples that are processed in each iteration during training. The batch size was set to 32 so as to allow the model update its weights more frequently with the potentials of leading to faster convergence. For the epoch, the maximum number of epochs defines the number of times the learning algorithm will work through the entire training dataset. In this case, 10 epochs are initially used for the LSTM and CNN algorithm, and 2 epochs was applied for the hybridized LSTM-CNN model. The optimizer applied is the RMSprop (Root Mean Square Propagation) which is an adaptive learning rate optimization algorithm that adjusts the learning rates of each parameter individually. It is suitable for deep learning tasks and can help converge faster and handle various types of data and models. The loss function utilized is the Mean Square Error (MSE) which measures the average squared difference between the actual and predicted values. Considering the activation function, the ReLU (Rectified Linear Unit), Tanh (Hyperbolic Tangent), and Hard-Sigmoid was used. The ReLU introduces non-linearity by replacing all negative values in the output with zero. It helps the model learn complex patterns, Tanh squashes the output to be between -1 and 1 and thus introduce non-linearity and handle vanishing gradient problems. The Hard-Sigmoid is a modified version of the sigmoid function that provides a faster approximation and is useful in reducing computation time.

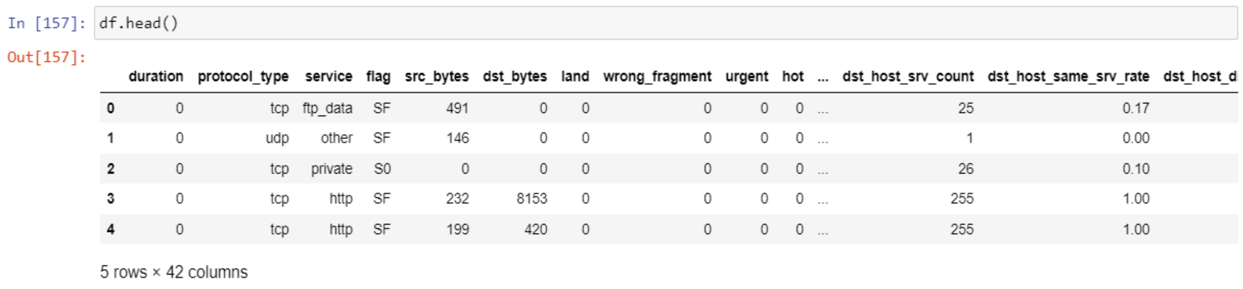
Table 4.1: Neural Network (LSTM-CNN) Model Parameter Setting

Parameter	**Value**
Batch Size	32
Max epochs	10, 2
Optimizer	Rmsprop
Loss	mean square error
Activation	ReLU/Tanh/hard-Sigmoid

Overall, these parameter settings reflect a combination that aims to strike a balance between effective learning, model convergence, and handling the characteristics of the data used in the network intrusion model. Fine-tuning and experimentation with these parameters may be necessary based on the specific characteristics of the dataset and the desired performance of the model.

Reading the Textual File

In the process of developing a network intrusion detection model, the crucial step of incorporating a dataset becomes apparent during training. Subsequently, the identification of intrusion incidents within the dataset was carried out using LSTM, CNN, and LSTM-CNN models. This experimental undertaking utilized the capabilities of the Pandas library to interact with the dataset. Pandas, an externally sourced Python library, is equipped with modules designed to simplify the ingestion of datasets in various file formats. In the context of this study, the dataset in question adopts the CSV (Comma Separated Values) file format. Notably, the Pandas library includes a 'read_csv' function proficient in assimilating and presenting the dataset attributes in a tabular structure, as illustrated in Figure 4.1. This functionality significantly contributes to the analytical framework of the investigation.



```
In [157]: df.head()
```

Out[157]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_d
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	25	0.17	
1	0	udp	other	SF	146	0	0	0	0	0	...	1	0.00	
2	0	tcp	private	S0	0	0	0	0	0	0	...	26	0.10	
3	0	tcp	http	SF	232	8153	0	0	0	0	...	255	1.00	
4	0	tcp	http	SF	199	420	0	0	0	0	...	255	1.00	

5 rows × 42 columns

Figure 4.2: depict records

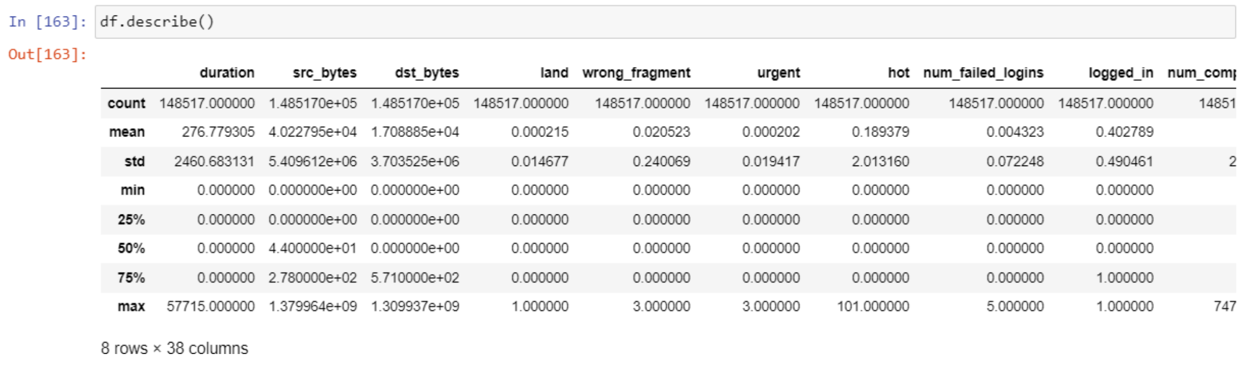
Data Exploration

This module, serving as a section for data exploration, facilitates the visualization of select features within the dataset. The utilization of visualizations empowers the scientific inquiry to discern inherent patterns and interrelationships among distinct variables encapsulated within the dataset.

Dataset Descriptive Statistics

Conducting thorough analytical research plays a crucial role in obtaining a comprehensive statistical representation of data properties, enabling the meticulous design and execution of optimal experiments. The tabulated presentation in Table 4.2 illustrates the scale data intervals covering 25, 50, and 75 percentiles, derived through the utilization of Pandas data frames. This tabular format encompasses essential statistical metrics, including count, mean, standard deviation, minimum, and maximum values. The count column enumerates instances of intrusion records within the dataset, totaling 148,517 intrusion records. The 'min' represents the minimum floating value within a given data column, 'std' denotes the standard deviation of the dataset specific to a labeled column, and 'max' articulates the maximum values observed in the corresponding column. Each column is associated with its distinctive values for the respective descriptive statistics, covering count, mean, standard deviation, minimum, maximum, and scale data intervals at 25, 50, and 75 percentiles, thereby contributing to a comprehensive analytical portrayal.

Table 4.2: Descriptive Statistics



```
In [163]: df.describe()
```

Out[163]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_comp
count	148517.000000	1.485170e+05	1.485170e+05	148517.000000	148517.000000	148517.000000	148517.000000	148517.000000	148517.000000	14851
mean	276.779305	4.022795e+04	1.708885e+04	0.000215	0.020523	0.000202	0.189379	0.004323	0.402789	
std	2460.683131	5.409612e+06	3.703525e+06	0.014677	0.240069	0.019417	2.013160	0.072248	0.490461	2
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	4.400000e+01	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	2.780000e+02	5.710000e+02	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	
max	57715.000000	1.379964e+09	1.309937e+09	1.000000	3.000000	3.000000	101.000000	5.000000	1.000000	747

8 rows × 38 columns

Percentage Split Technique

In training the LSTM, CNN, and LSTM-CNN (Hybrid) algorithms, this study divided the dataset into some training and test proportions, where the training set was used to train the models and the test set to validate the workings of the model. From the total of 148,517 intrusion records that were categorized into 2 classes seventy per cent was utilized to train the models whereas the remaining 30% was used to test the viability of the models.

4.5 Models Result

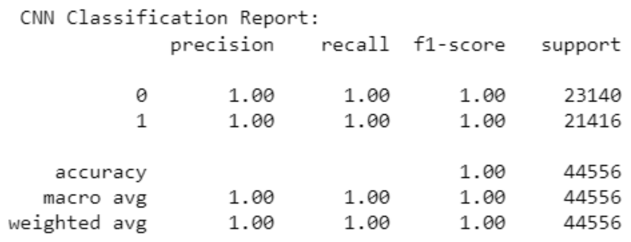
The table 4.3 presents the results of three different network intrusion models: CNN (Convolutional Neural Network), LSTM (Long Short-Term Memory), and LSTM-CNN hybrid. Each model underwent training for a specified number of epochs (training cycles) and utilized a batch size of 32 instances during each iteration. The chosen optimizer for all models was rmsprop. The accuracy percentages signify the models' performance in correctly classifying instances, with higher values indicating greater accuracy. The CNN model achieved an impressive accuracy of 99.98% after 10 epochs, demonstrating its effectiveness in discerning patterns within the network intrusion dataset. The LSTM model also performed commendably with an accuracy of 99.96% after 10 epochs, showcasing its proficiency in capturing long-term dependencies. The hybrid LSTM-CNN model, despite undergoing only 2 epochs, exhibited exceptional accuracy, reaching 100%. This outstanding result suggests that the combination of LSTM and CNN as an hybrid architectures synergistically contributed to robust and accurate predictions. Overall, these models showcase promising capabilities in the domain of network intrusion detection, with each exhibiting high accuracy levels under the specified training configurations.

Table 4.3: Network Intrusion Models Result

Models	Epoch	Batch	Optimizer	Accuracy (%)
CNN	10	32	Rmsprop	99.98
LSTM	10	32	Rmsprop	99.96
LSTM-CNN	2	32	Rmsprop	100

Result Evaluation (Classification Report)

The CNN classification report provides a detailed evaluation of the performance of the Convolutional Neural Network (CNN) in the context of network intrusion detection as shown in figure 4.4. The report includes precision, recall, and F1-score metrics for each class (0 and 1). For Class 0, which represents instances of non-intrusions, the CNN achieved perfect precision, recall, and F1-score, all equal to 1.00. This indicates that the model correctly identified and classified all instances of non-intrusions without any false positives or false negatives. The support value of 23,140 denotes the total number of instances belonging to Class 0 in the dataset. Similarly, for Class 1, corresponding to instances of intrusions, the CNN achieved perfect precision, recall, and F1-score, all equal to 1.00. This indicates that the model accurately identified and classified all instances of intrusions without any misclassifications. The support value of 21,416 denotes the total number of instances belonging to Class 1 in the dataset. The overall accuracy of the CNN is 1.00, indicating that it achieved perfect classification accuracy on the entire dataset, which consists of 44,556 instances.



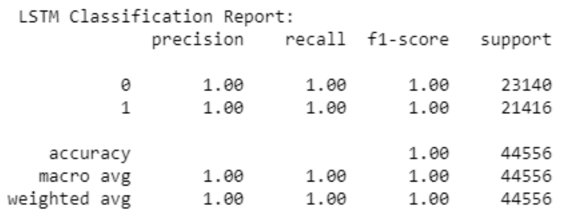
```
CNN Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     23140
           1       1.00      1.00      1.00     21416

    accuracy                           1.00     44556
   macro avg       1.00      1.00      1.00     44556
weighted avg       1.00      1.00      1.00     44556
```

Figure 4.4: CNN classification Report

The LSTM (Long Short-Term Memory) model for network intrusion detection also exhibits outstanding performance, as indicated by the classification report in Figure 4.5. In the classification report, precision, recall, and f1-score metrics are also provided for two classes (0 and 1) along with overall accuracy. Hence, in this case, both classes (0 and 1) demonstrate perfect precision, recall, and f1-score, each achieving a value of 1.00. This implies that the LSTM model achieved flawless predictions for both normal (class 0) and intrusive (class 1) instances in the dataset.



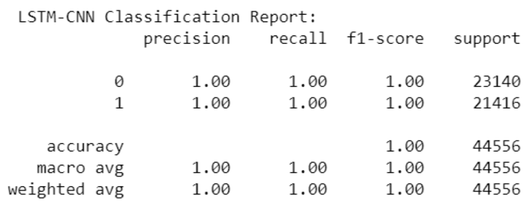
```
LSTM Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     23140
           1       1.00      1.00      1.00     21416

    accuracy                           1.00     44556
   macro avg       1.00      1.00      1.00     44556
weighted avg       1.00      1.00      1.00     44556
```

Figure 4.5: LSTM classification Report

The classification report for the LSTM-CNN network intrusion detection model demonstrates exceptional performance as shown in Figure 4.6. The precision, recall, and f1-score metrics for both classes (0 and 1) are all perfect, each achieving a value of 1.00. This indicates that the model made no false positive or false negative predictions for either class, achieving complete accuracy in its classifications. The accuracy of the model, as indicated by the overall accuracy metric, is also 1.00, reinforcing the model's ability to correctly classify instances from both classes.



```
LSTM-CNN Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     23140
           1       1.00      1.00      1.00     21416

    accuracy                           1.00     44556
   macro avg       1.00      1.00      1.00     44556
weighted avg       1.00      1.00      1.00     44556
```

Figure 4.6: LSTM-CNN classification Report

Result Evaluation (Confusion Matrix)

The confusion matrix provided reflects the performance of a Convolutional Neural Network (CNN) in a classification task as shown in figure 4.7. In this matrix, the top-left quadrant (23137) represents the True Negatives (TN), indicating instances where the CNN correctly classified non-intrusive network data as non-intrusive. The top-right quadrant (3) signifies False Positives (FP), denoting cases where non-intrusive data was mistakenly classified as intrusive. The bottom-left quadrant (5) represents False Negatives (FN), indicating instances where intrusive data was misclassified as non-intrusive. Finally, the bottom-right quadrant (21411) stands for True Positives (TP), depicting the correct classification of intrusive network data as intrusive. These values contribute to various evaluation metrics such as precision, recall, and accuracy, providing insights into the CNN model's effectiveness in distinguishing between intrusive and non-intrusive network activities.

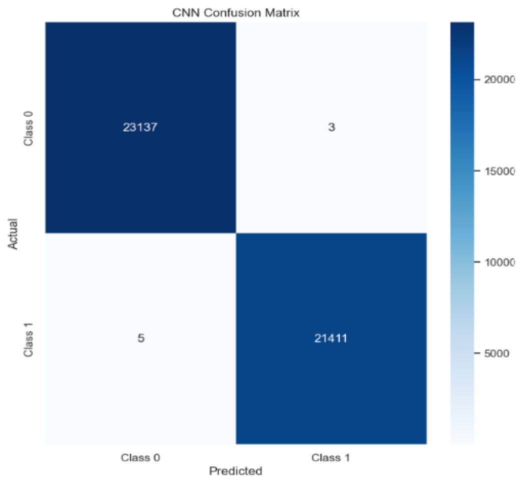


Figure 4.7: CNN Confusion Matrix Report

The confusion matrix for the LSTM supports the exceptional performance of the LSTM network intrusion model classification report providing a detailed breakdown of predicted and actual class labels as shown in Figure 4.8. The values on the diagonal represent correct predictions, while off-diagonal elements signify misclassifications. In this confusion matrix, the model made only 20 errors out of the total 44,556 instances, with 10 instances misclassified as class 0 and 10 as class 1. The overwhelming majority of predictions align with the actual classes, resulting in an accuracy of 100%. This remarkable accuracy, along with perfect precision, recall, and f1-score for both classes, underscores the LSTM model's robust ability to accurately classify instances of network intrusion, making it a highly effective tool for intrusion detection.

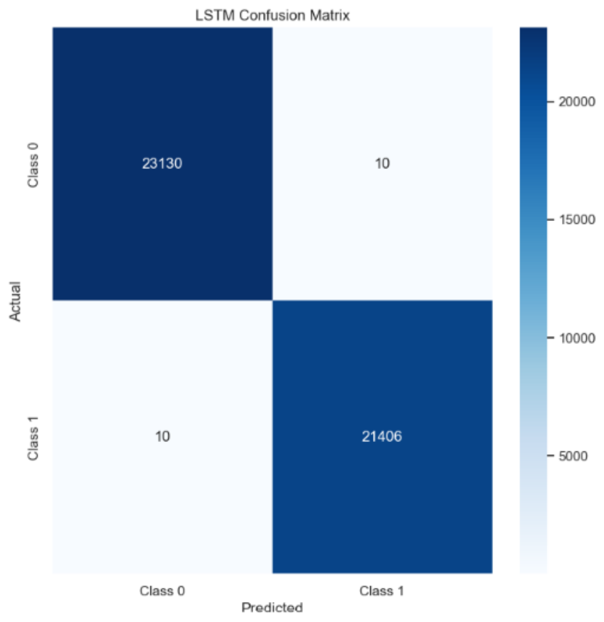


Figure 4.8: LSTM Confusion Matrix Report

The confusion matrix for the hybrid model (LSTM-CNN) as shown in figure 4.9 further supports findings from the hybrid LSTM-CNN classification report, with no misclassifications evident. Class 0, representing non-intrusive instances, has 23,140 true positive predictions, and Class 1, representing intrusive instances, has 21,416 true positive predictions. Both classes have zero false positives or false negatives. In summary, the LSTM-CNN model exhibits outstanding accuracy and precision in detecting network intrusions, achieving a perfect score across all performance metrics.

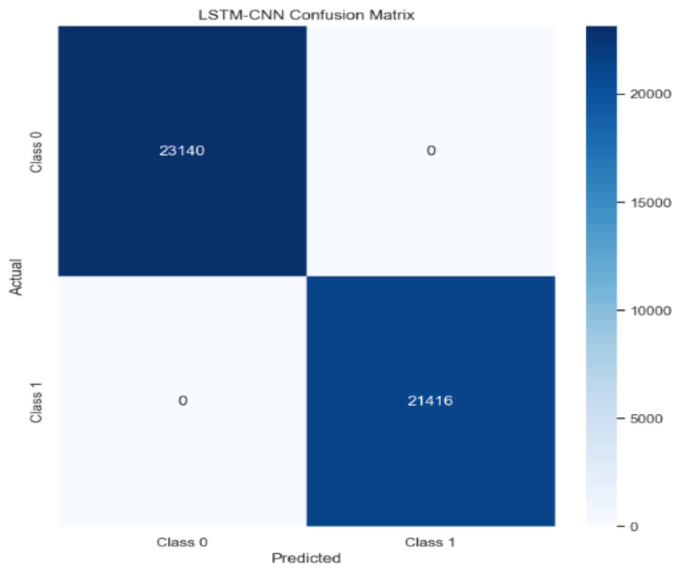


Figure 4.9: LSTM-CNN (Hybrid) Confusion Matrix Report

Result Comparison

The table 4.4 presents a comparison of the results obtained from various studies on network intrusion detection. [10] achieved an accuracy of 79.10% using a Sparse Autoencoder in conjunction with Soft-max Regression on the NSL-KDD dataset. [12] employed a Recurrent Neural Network (RNN) on the same NSL-KDD dataset and attained a slightly lower accuracy of 75.75%. Priyadarshini and Barik (2019) utilized LSTM on the FOG dataset and obtained a notably high accuracy of 98.88%.[20] introduced an Epigenetic Algorithm-Based Detection Technique, achieving an impressive detection rate of up to 98% without specifying a dataset. [16] employed Rough-Set Theory, Back-propagation Neural Network (BPNN), and Discrete variant of Cuttlefish Algorithm (D-CFA) on KDD99 and UNSW-NB15 datasets, achieving an accuracy above 84% using Rough-Set Theory. [17] developed an IDS utilizing Artificial Intelligence Methods, claiming high efficiency and accuracy without specifying exact performance metrics. In comparison, the current study, conducted in 2024, implemented a hybrid CNN-LSTM model on the NSL-KDD dataset, achieving an exceptionally high accuracy of 99.99%, outperforming all other approaches in the literature. This demonstrates a significant advancement in the field of network intrusion detection, suggesting that the CNN-LSTM hybrid model is highly effective in accurately identifying and classifying network intrusions.

Table 4.4: Result Comparison with other work on Network Intrusion

Authors	Year	Dataset	Algorithm	Best Algorithm	Score of Best Algorithm
Javaid et al. (2016)	2016	NSL-KDD	Sparse Autoencoder, Soft-max Regression	Sparse Autoencoder	79.10%
Yin et al. (2017)	2017	NSL-KDD	Recurrent Neural Network (RNN)	RNN	75.75%
Ezzarii et al. (2020)	2020	N/A	Epigenetic Algorithm-Based Detection Technique	EGA	Up to 98%
Khairul et al. (2020)	2020	KDD99, UNSW-NB15	Rough-Set Theory, BPNN, D-CFA	Rough-Set Theory	Above 84%
Abdiyeva-Aliyeva et al. (2021)	2021	N/A	Artificial Intelligence Methods	Scale-Hybrid-IDS-AlertNet	High efficiency and accuracy
Current Study	2024	NSL-KDD	CNN, LSTM, CNN-LSTM	CNN-LSTM	99.99

SUMMARY, CONCLUSION, AND RECOMMENDATION.

Summary

In the contemporary digital landscape, the escalating frequency and sophistication of cyber threats pose a formidable challenge to the security of networked systems as identified by the survey conducted by this study. The need for Network Intrusion Detection Models (NIDMs) arises from the imperative to fortify these systems against unauthorized access, data breaches, and other malicious activities. NIDMs play a pivotal role in proactively identifying and mitigating potential threats by analyzing network traffic patterns, anomalous behaviors, and known attack signatures. Their deployment is essential for safeguarding sensitive information, ensuring data integrity, and maintaining the overall resilience of interconnected infrastructures. As cyber threats evolve, the continuous development and enhancement of NIDMs become paramount to stay ahead of adversaries and uphold the security and trustworthiness of digital networks.

To address these concerns, this study devised a systematic three-phase methodology for developing network intrusion detection models. The initial phase involved meticulous data preparation, encompassing the removal of unwanted characters, handling missing values, and feature selection using a correlation matrix. Subsequently, the cleansed and scaled dataset was fed into machine learning algorithms, specifically LSTM, CNN, and their hybrid, maintaining a 70:30 training-to-test ratio. The models were evaluated using precision, recall, accuracy, and f1-score metrics as the third phase of the methodology. The implementation of this research was facilitated through the use of the Python programming language, supported by key third-party libraries such as NumPy, Matplotlib, Pandas, TensorFlow, and Sklearn. This sequential methodology aimed to provide a structured and effective framework for the development and evaluation of network intrusion detection models.

Conclusion

In conclusion, the conducted experiments for network intrusion detection using different models, namely Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and a hybrid LSTM-CNN, have yielded promising results. The CNN model achieved an impressive accuracy of 99.98% after 10 epochs, showcasing its effectiveness in classifying network activities. Similarly, the LSTM model achieved a high accuracy of 99.96% with the same number of epochs and batch size. Notably, the hybrid LSTM-CNN model demonstrated exceptional performance, achieving a perfect accuracy of 100% after only 2 epochs. These results indicate the robustness and efficiency of the proposed hybrid model in detecting network intrusions. The use of the rmsprop optimizer in all models contributed to their convergence and overall success. The findings suggest that the hybrid LSTM-CNN model holds great potential for network intrusion detection, offering a compelling option for securing network systems against malicious activities.

Suggestions for Further Research

After an extensive experimental study and review of several kinds of literature relating to the developed network intrusion detection models, this study suggests future research as follows:

i. **Ensemble Approaches**: future studies can investigate the potential of ensemble methods by combining the strengths of different models, such as KNN, Multilayer Perceptron, etc. Ensemble techniques, like stacking or bagging, could potentially improve overall performance by leveraging the complementary strengths of diverse models.

ii. **Dynamic Adaptability:** future studies can also explore the development of intrusion detection systems that dynamically adapt to evolving network environments. This could involve the integration of reinforcement learning or other adaptive techniques to enhance the system's ability to recognize novel intrusion patterns.

REFERENCES

[1] Shiravani, A., Sadreddini, M. H., and Nahook, H. N. (2023). Network intrusion detection using data dimensions reduction techniques. *Journal of Big Data, 10*(1), 1-25. https://doi.org/10.1186/s40537-023-00697-5.

[2] Tharewal, S., Ashfaque, M. W., Banu, S. S., Uma, P., Hassen, S. M., & Shabaz, M. (2022). Intrusion detection system for industrial Internet of Things based on deep reinforcement learning. *Wireless Communications and Mobile Computing*, *2022*, 1-8.

[3] Naveed, M., Arif, F., Usman, S. M., Anwar, A., Hadjouni, M., Elmannai, H., ... & Umar, F. (2022). A Deep Learning-Based Framework for Feature Extraction and Classification of Intrusion Detection in Networks. *Wireless Communications and Mobile Computing*, *20-22*.

[4] Santhosh Kumar, S. V. N., Selvi, M., & Kannan, A. (2023). A comprehensive survey on machine learning-based intrusion detection systems for secure communication in the Internet of Things. *Computational Intelligence and Neuroscience*, *20-23*.

[5] Lan, J., Lu, J. Z., Wan, G. G., Wang, Y. Y., Huang, C. Y., Zhang, S. B., ... & Ma, J. N. (2022). E-minBatch GraphSAGE: An Industrial Internet Attack Detection Model. *Security and Communication Networks*, 2022.

[6] Bhati, N.S., and Khari, M. (2020). A new ensemble-based approach for intrusion detection system using voting. *J Intell Fuzzy Syst*., 42(2),969–79.

[7] Su, T., Sun, H., Zhu, J., Wang, S., and Li, Y. (2020). BAT: deep learning methods on network intrusion detection using NSL-KDD dataset. *IEEE Access*, 8, 29575–85.

[8] Jianjian, D., Yang, T., and Feiyue, Y. (2018). A novel intrusion detection system based on IABRBFSVM for wireless sensor networks. *Procedia Comput Sci.*,13–21.

[9] Al-Safi, AHS., Hani, ZIR., and Abdul-Zahra, M.M. (2021). Using a hybrid algorithm and feature selection for network anomaly intrusion detection. *J Mech Eng Res Dev.*, 44(4):253–62.

[10] Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A deep learning approach for network intrusion detection systems. *Information and Communications Technologies*, 21-26.

[11] Ioannou, C., Vassiliou V., and Sergiou, C. (2017). An Intrusion Detection System for Wireless Sensor Networks. *International Conference on Telecommunications (ICT)*, 2017, 1-5, DOI: 10.1109/ICT.2017.7998271.

[12] Yin, C., Zhu, Y., Fei, J., He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*. 2017, 5, 21954-21961.

[13] Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2016). Deep learning approach for network intrusion detection in software-defined networking. *International Conference on Wireless Networks and Mobile Communications IEEE*, 258-263.

[14] Ashfaq, R. A. R., Wang, X.-Z., Huang, J. Z., Abbas, H., & He, Y.-L. (2017). Fuzziness-based semi-supervised learning approach for the intrusion detection system. *Information Sciences*, 378, 484–497.

[15] Shone, N., Ngoc, TN., Phai, V.D., and Shi, Q. (2018). A deep learning approach to network intrusion detection. *IEEE Trans Emerg Top Comput Intel*, 2(1):41-50.

[16] Khairul, A.Z.A., Salwani, A., and Mohammed, F.E. (2020). An analysis of the KDD99 and UNSW-NB15 Datasets for the intrusion Detection System. *Symmetry*, *12*(10), 1666.

[17] Abdiyeva-Aliyeva, G., Hematyar, M., and Bakan, S. (2021). Development of a System for the Detection and Prevention of Cyber Attacks Using Artificial Intelligence Methods. *Global Conference for Advancement in Technology (GCAT),* 1-5.

[18] Balasubramaniam, S., Vijesh Joe, C., Sivakumar, T. A., Prasanth, A., Satheesh Kumar, K., Kavitha, V., and Dhanaraj, R. K. (2023). Optimization Enabled Deep Learning-Based DDoS Attack Detection in Cloud Computing. *International Journal of Intelligent Systems, 2023.*

[19] Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation*, *31*(7), 1235-1270.

[20] Ezzarii, M., Ghazi, H. E., Ghazi, H. E., and Bouanani, F. E. (2020). Epigenetic Algorithm-Based Detection Technique for Network Attacks. IEEE Access, 8, 199482-199491.