




Journal of Soft Computing and Artificial Intelligence

Journal homepage: <https://dergipark.org.tr/en/pub/jscai>

International
Open Access 

Volume 05
Issue 02

December, 2024

Research Article

Cybersecurity in the Internet of Things: the Detection of the Types of Upcoming Digital Information by Using Classification Techniques

Dima Raed Abu Khalil¹ , Yousef Abuzir² 

^{1,2} Faculty of Technology and Applied Sciences, Al-Quds Open University, Ramallah, Palestine

ARTICLE INFO

Article history:

Received **October 30, 2024**

Revised **November 22, 2024**

Accepted **November 25, 2024**

Keywords:

Network Attack Detection

Internet of Things (IoT)

Machine Learning

Logistic Regression

Linear Discriminant

Analysis (LDA)

Deep Learning

ABSTRACT

This study addresses the critical challenge of Cyber-attacks detection (CAD) in the Internet of Things (IoT) environment, specifically focusing on the classification of non-malicious and malicious network traffic. The primary objective is to enhance the accuracy and reliability of detection mechanisms through the implementation of advanced machine learning models, particularly the hybrid CNN-GRU-LSTM model. The study utilizes the SYN DoS dataset from the Kitsune Network Attack Dataset to train and evaluate various models, including Linear Discriminant Analysis (LDA), Logistic Regression, and the CNN-GRU-LSTM model. The methodology includes a comprehensive performance analysis of each model, employing metrics such as accuracy, precision, recall, and F1-score. The results reveal that both LDA and Logistic Regression achieved perfect accuracy (1.00), while the CNN-GRU-LSTM model exhibited an accuracy of 0.998. Additionally, the CNN-GRU-LSTM model demonstrated a high area under the curve (AUC) value of 0.8559, indicating strong discriminatory power. The study further employs SHAP (SHapley Additive exPlanations) for model interpretability, allowing for a detailed analysis of feature importance and insights into model behavior. In conclusion, the hybrid CNN-GRU-LSTM model offers a promising approach for effective network attack detection while providing a basis for future improvements in real-time applications and the exploration of additional datasets.

1. Introduction

In the era of digital transformation, network security has become a critical concern for organizations worldwide. With the increasing reliance on networked systems, the frequency and sophistication of cyberattacks have escalated, posing significant threats to data integrity, privacy, and operational stability [1,2]. Network attacks, including Distributed Denial of Service (DDoS) and

Man-in-the-Middle (MitM) attacks, can disrupt services and compromise sensitive information, making it imperative to develop robust methods for detecting and classifying these threats effectively [3, 4].

Traditional network attack detection methods, often reliant on signature-based detection, struggle to keep pace with the evolving nature of cyber threats. Attackers continuously develop new methods to

* Corresponding author

e-mail: yabuzir@qou.edu

DOI: 10.55195/jscai.1576195

bypass signature-based defenses, highlighting the limitations of these traditional approaches. False positives and negatives further hamper the effectiveness of these methods, leading to wasted resources and potential security breaches [5] [6] [7].

Deep learning (DL), a subfield of artificial intelligence, offers a promising solution for network attack detection [9]. DL algorithms are effective at identifying complex patterns within vast amounts of dataset. By examining network traffic data, DL models have the ability learn to distinguish between normal and malicious network behavior, offering a more adaptable and robust approach to network attack detection [10] [11] [12].

This research explores the potential of deep learning (DL) and machine learning (ML) techniques for accurately predicting and detecting SYN DoS attacks. We hypothesize that deep learning models can outperform traditional methods due to their superior pattern recognition capabilities [11]. We will investigate the effectiveness of two machine learning and a deep learning hybrid CNN-GRU-LSTM model – for identifying SYN DoS attacks within the Kitsune SYN DoS dataset (<https://www.kaggle.com/datasets/ymirsky/network-attack-dataset-kitsune/data>). Our evaluation will compare the performance of these machine and deep learning model, emphasizing key metrics such as accuracy, precision, recall, and F1-score [12] [13].

The Problem Question is How can we accurately classify and detect various types of network attacks using machine and deep learning techniques, and what are the most effective models for distinguishing between harmful and benign network traffic?

The motivation behind this research stems from the growing need for advanced and efficient solutions to enhance network security. Traditional methods of network attack detection often struggle with high false-positive rates and limited adaptability to new attack vectors. Machine learning offers the potential to improve detection accuracy and adapt to evolving threats by learning from historical data. By exploring and comparing various machine learning models, this research aims to identify the most effective approaches for classifying network attacks, thereby contributing to the development of more resilient cybersecurity systems.

The novelty of this study lies in the integration of CNN, GRU, and LSTM into a single hybrid model

that addresses the complexities of network traffic data. Previous studies have largely relied on individual models, such as CNNs for feature extraction or LSTMs for sequence learning, but have not explored the synergy between these architectures in the context of network attack detection.

This study makes several key contributions to the field of network security and machine learning:

- **Comprehensive Evaluation of Models:** The study assesses and compares the performance of Linear Discriminant Analysis (LDA), Logistic Regression, and a hybrid CNN-GRU-LSTM model within the framework of network attack classification. A comprehensive evaluation of this hybrid approach, demonstrating its superior accuracy and ability to handle class imbalance in detecting both malicious and benign network instances.
- **Detailed Analysis of Network Attack Data:** By using the Kitsune Network Attack Dataset, the study provides an in-depth analysis of various attack types, including SYN DoS, and demonstrates how machine learning models can be applied to detect and classify these attacks.
- **Model Interpretability:** The use of SHAP (SHapley Additive exPlanations) to explain the models' predictions provides valuable information about the factors influencing the classification decisions, enhancing the transparency and trustworthiness of the machine learning models.
- **The hybrid model that leverages the strengths of CNN for spatial feature extraction, GRU for short-term temporal dependencies, and LSTM for capturing long-term temporal dependencies in network traffic.**
- **The integration of advanced optimization techniques such as dropout regularization and early stopping to avoid overfitting and ensure model generalization on unseen data.**

This hybrid model provides a novel framework for intrusion detection systems, improving both accuracy and computational efficiency compared to traditional and single-model approaches.

This study examines the development of machine learning for network attack detection. It begins by reviewing existing methods and datasets before delving into model development using techniques like LDA, logistic regression, and hybrid CNN-GRU-LSTM. Model performance is evaluated using

standard metrics, and their effectiveness is analyzed using interpretability tools like SHAP. The study concludes by discussing the implications of the findings for network security and outlining directions for future research.

2. Literature Review

Researchers are increasingly interested in using deep learning (DL) to create online network attack detection. This is because machine learning (ML) and DL techniques have been shown to be effective in identifying cyberattacks launched from compromised Internet of Things (IoT) devices [14], [15].

One challenge with traditional ML-based network attack detections is that they require a lot of labelled data for training. To address this, a new approach called Decentralized and Online Federated Learning Intrusion Detection (DOF-ID) has been proposed. This system allows collaborating devices to share information and improve intrusion detection performance across the network [16] [17].

Another promising technique is Deep Transfer Learning (DTL), a type of DL that can be used to enhance intrusion detection within industrial control systems. DTL allows the system to gain insights from data in one domain (such as general network traffic) and apply insights to a different domain (such as industrial control systems) [18] [19]. This can improve detection accuracy and help mitigate threats more effectively. Overall, these studies show that advanced ML and DL techniques have a lot of potential for improving the capabilities of online network intrusion detection systems.

In the paper by Hussain et al. (2023), the authors reviewed various intrusion detection models and the threats posed to IoT systems by compromised devices, emphasizing the use of ML and DL techniques as effective defensive measures [15]. Mert et al. (2023) proposed the DOF-ID architecture, which enhances intrusion detection by allowing IDSs to learn from both local and remote data while maintaining data privacy, showing significant performance improvements on Kitsune and Bot-IoT datasets [20]. Kheddar et al. (2023) provided a comprehensive review of using deep transfer learning (DTL) in industrial control networks for intrusion detection, highlighting improvements in detection accuracy and the use of multiple datasets and

evaluation metrics like accuracy and false alarm rate [18].

Wasnik and Chavhan (2023) tested different DL algorithms on public malware data. They found these models work well, but need frequent updates to stay ahead of changing attack patterns. The authors propose a specific deep neural network (DNN) that can adapt to dynamic network behavior. They suggest constantly improving these models and linking them to real-time monitoring for proactive threat detection [20]. Ogundokun et al. (2023) systematically reviewed the application of ML and DL algorithms in IDS, analyzing various classifiers, datasets, and frameworks used from 2016 to 2021, and offering insights into recent advancements and challenges [22].

Krishna et al. (2020) focused on building an IDS that can also prevent attacks (DOS, Probe, R2L, U2R). They used a Multi-Layer Perceptron (MLP) deep learning model on the KDDCup99 dataset and achieved high accuracy against different attack types. Their system combines detection and prevention, proving effective in real-time situations. The authors recommend further development of the prevention mechanisms and testing the system across diverse network environments [23].

The study of (Fadel et al., 2022) proposed the HDLIDP framework, combining signature-based and deep learning techniques to improve DDoS attack detection and prevention in SDNs, demonstrating significant accuracy improvements through experiments on traditional and SDN datasets [24].

The research by (Alghamdi, 2022) proposes a hybrid intrusion detection model (PO-CFNN) for securing Internet of Things (IoT) devices. It uses a unique optimization technique and achieves high performance on test datasets. The author suggests adapting the model for more complex IoT scenarios and improving its efficiency [25].

The review by (Monani et al., 2023) explores different ways to analyze cyber threats using data, like predicting denial-of-service attacks. The study suggests combining these methods for stronger defenses [26].

Deep Learning for Network Security: The survey of (Auwal, 2022) highlights how deep learning can improve network intrusion detection compared to traditional methods. It identifies areas for further research, like exploring techniques that don't require

as much labeled data [27].

The study of (Hnamte & Hussain, 2023) proposes a deep learning model using convolutional neural networks to detect and classify network intrusions. It shows promising results, but needs testing with more diverse data to ensure real-world effectiveness [28].

Alabdulatif and Rizvi (2022) addressed the improvement of Kitsune Network Intrusion Detection (NID) using machine learning techniques. They evaluated various tree algorithm variants on Kitsune datasets, ultimately recommending the Fine Tree algorithm for better performance. The main metrics used were effectiveness and efficiency. The results indicated that the Fine Tree algorithm outperformed other tree variants in terms of improving Kitsune NID's accuracy and reliability [29].

The study by Malliga et al. (2022) examines the efficacy of deep learning techniques in identifying DoS/DDoS attacks. They concluded that deep learning is capable of managing these evolving threats [30]. Sujatha et al. (2023) examined the application of deep reinforcement learning (DRL) for network intrusion detection [31]. They reported that

their DQL model achieve high degree and impressive accuracy in identifying intrusion.

Mohammed et al. (2023) in their review on machine learning and deep learning strategies for DDoS detection in Software-Defined Networking (SDN) frameworks. Their results reveal highlight a growing interest in utilizing these techniques, with challenges related to datasets [32]. In a similar vein, Omarov et al. (2022) explore current techniques for detecting network intrusions in Internet of Things (IoT) scenarios. They point out the lack of computational models and formal justification of attacks as key challenges in this area [33].

Researchers are exploring various Machine Learning (ML) and Deep Learning (DL) techniques to improve cyberattack detection. These techniques address challenges like data privacy, adaptability, and evolving threats. Studies recommend ongoing model refinement, using advanced algorithms, and testing in real-world scenarios. Overall, these advancements in ML and DL are crucial for building robust and adaptable cyberattack detection systems to combat cyber threats.

Table 1 Summary for some researches in applying ML and DL in network attack detection

| Reference | Problem | ML/DL Techniques | Evaluation Metrics | Main Results |
|--------------------------------------|--|---|-------------------------------------|--|
| Hussain et al. (2023) | IoT-based cyber-attacks due to device proliferation | ML, DL | Not specified | Effective control against IoT-originated attacks |
| Mert et al. (2023) | Limited applicability of ML-based IDSs due to private local data | Federated learning | Accuracy, computation time | Improved intrusion detection performance across nodes |
| Kheddar et al. (2023) | Protecting industrial control systems from various threats | Deep Transfer Learning | Accuracy, F-score, false alarm rate | Enhanced IDS performance with scarce labeled data |
| Ogundokun et al. (2023) | Lack of comprehensive studies on ML for IDS | ML, DL algorithms | Not specified | Insights into advancements and challenges from 2016-2021 |
| Fadel et al. (2022) | DDoS attacks on SDN controllers | Hybrid Deep Learning | Classification accuracy | Significant improvement in detection accuracy |
| Alabdulatif and Rizvi (2022) | Improving Kitsune NID | Variants of Tree algorithms (Simple Tree, Medium Tree, Coarse Tree, RUS Boosted, Bagged Tree) | Confusion Matrix, Speed, Accuracy | Fine Tree algorithm outperformed others |
| Malliga, Nandhini, Kogilavani (2022) | Detecting DoS/DDoS attacks | Deep learning models | Various performance metrics | Deep learning models have improved detection capabilities but need further enhancement |

| | | | | |
|-------------------------|-------------------------------|------------------------------------|---|--|
| Sujatha et al. (2023) | Network intrusion detection | Deep Q-Learning (DQL) | Accuracy, recall rate, precision | DQL model achieved 91.4% accuracy, outperforming other models |
| Bahashwan et al. (2023) | Detecting DDoS attacks in SDN | ML, DL, hybrid approaches | Evaluation based on Various performance metrics (Accuracy) and unrealistic datasets | Ensemble, hybrid, and single ML-DL approaches are most used but need improvement |
| Omarov et al. (2022) | Network intrusion detection | Taxonomy of detection technologies | Evaluation of advanced research topics | Highlights need for computational models and formal attack justification |

3. Methodology

In this study, we implemented a systematic and robust methodology to address the challenges posed by network attack detection using machine learning techniques. Our approach involves several key stages, from data acquisition and preprocessing to model training, evaluation, and interpretability. The

goal is to develop effective and interpretable models that can accurately classify network traffic as benign or malicious. Figure 1, presents a conceptual diagram illustrating the methodology for network attack detection research. The diagram highlights the workflow from data acquisition through model evaluation and interpretability.

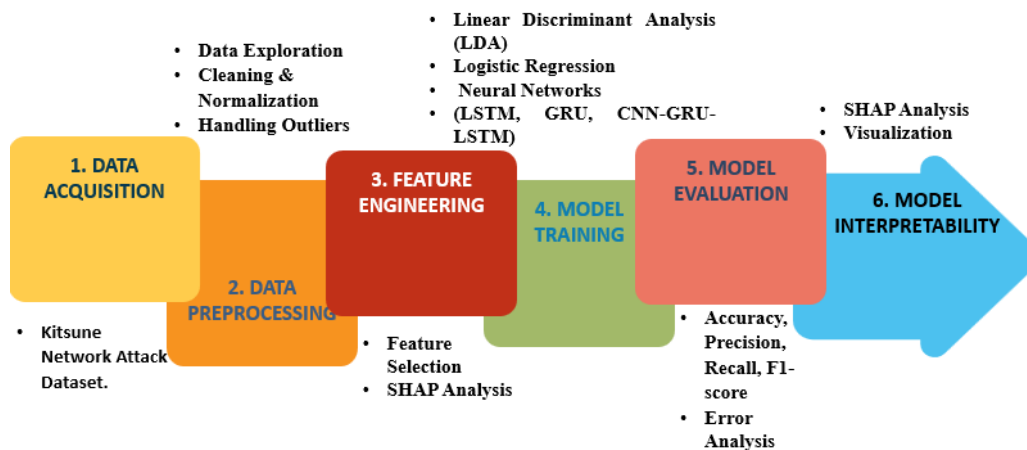


Figure 1 A conceptual diagram illustrating the methodology

Data Acquisition: The first step involves sourcing the Kitsune Network Attack Dataset, specifically the SYN DoS dataset. It provides a rich set of features related to various types of network attacks. This dataset serves as the foundation for our study, offering diverse and representative samples for model training and evaluation.

Data Preprocessing: Once the data is acquired, it undergoes extensive preprocessing. This stage includes data exploration to understand the structure and quality of the dataset, followed by cleaning and normalization to prepare the data for model training. Outlier detection and handling are also performed to ensure the data's integrity and improve model performance.

Feature Engineering: Feature engineering is a

crucial step where we select and refine the features used in model training. This step applies techniques such as SHAP (SHapley Additive exPlanations) to assess feature significance and comprehend the impact of different features on model predictions. This step ensures that the models leverage the most relevant information for accurate classification.

Model Training: We then train a variety of machine learning models to address the classification task. The models include Linear Discriminant Analysis (LDA), Logistic Regression, and a hybrid CNN-GRU-LSTM model. Each model is trained and fine-tuned to optimize its performance in detecting network attacks.

Model Evaluation: The trained models are evaluated using several metrics to assess their

performance. Metrics such as accuracy, precision, recall, and F1-score are calculated to provide a comprehensive view of each model's effectiveness. Error analysis is also conducted to identify any patterns in misclassification and areas for improvement.

Model Interpretability: To ensure that the models are not only effective but also interpretable, we use SHAP analysis to explain model predictions. This step involves visualizing SHAP values to understand how different features influence the model's decisions, providing transparency and trust in the model's outputs.

The methodology outlined here integrates these steps into a coherent process aimed at developing robust and interpretable models for network attack detection. By systematically addressing each phase of the study, we aim to enhance the reliability and practical applicability of machine learning solutions in network security.

3.1 Dataset and Preprocessing

3.1.1. Description of the Kitsune Network Attack Dataset

Dataset

The Kitsune Network Attack Dataset is a comprehensive dataset designed to facilitate the analysis and classification of various network attacks. It includes network traffic data captured from a simulated environment where different types of network attacks were introduced. The dataset is hosted on Kaggle and provides CSV files with detailed information about network traffic, including both benign and malicious activities.

The dataset encompasses a wide range of attack types, such as:

- ARP MitM (Address Resolution Protocol Man-in-the-Middle)
- SYN DoS (SYN Denial of Service)
- Active Wiretap

Each type of attack is represented by specific CSV files containing attributes related to the network traffic during the attack. Key attributes typically include time-related features, packet counts, and

other metrics essential for understanding the nature and impact of the attacks.

The research utilized the SYN DoS dataset from the Kitsune Network Attack Dataset to analyze and evaluate network attack detection systems. This dataset specifically focuses on SYN flood attacks, a common type of denial-of-service attack that targets the TCP handshake process. By leveraging the SYN DoS dataset, the study aims to develop and test detection algorithms that can effectively identify and mitigate such attacks, thereby enhancing network security measures.

3.1.2. Data Exploration and Preprocessing Steps

Data Inspection: Initial exploration of the dataset involves using methods such as `df.info()`, `df.head()` (Figure 2), and `df.describe()` to understand the structure, size, and summary statistics of the dataset. The output in Figure 2 the initial five rows and first 115 columns of the SYN DoS dataset, are presented, as well as relevant information about the DataFrame. Figure 3, shows a summary statistics table for the DataFrame `df`. By default, it calculates the count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, and maximum values for each numeric column in the DataFrame. The output shows summary statistics for all 115 columns in the DataFrame. The first column, Count, shows the number of non-null values in each column. The Mean column shows the average value for each column, and the Standard column shows the standard deviation. The lowest column displays the minimum value, and the 25% column displays the 25th percentile. The 50% column displays the average, and the 75% column displays the 75th percentile. Finally, the Maximum column displays the maximum value for each column. Here we notice that the count values for all columns are the same, which indicates that there are no missing values in the DataFrame. In addition, the mean values for columns 107-115 are very small, indicating that these columns may contain mostly zero values.

These methods provide insights into data types, missing values, and the basic distribution of feature values.

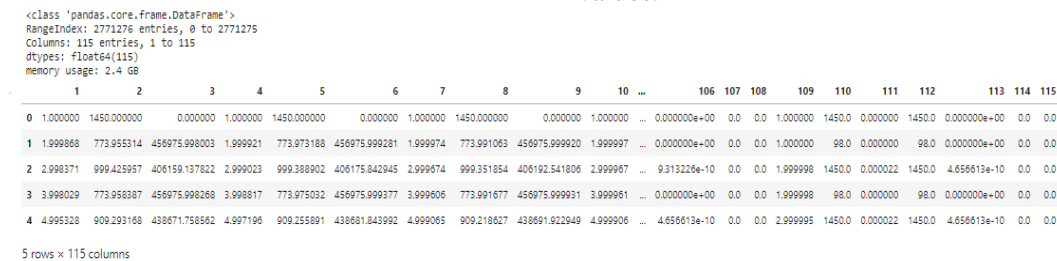


Figure 2 Output of `df.info()` and `df.head()`.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| count | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 |
| mean | 6.211946e+01 | 8.024928e+02 | 3.053892e+05 | 9.963715e+01 | 8.032211e+02 | 3.055215e+05 | 2.913213e+02 | 8.041999e+02 | 3.056160e+05 | 2.881085e+03 |
| std | 2.558334e+01 | 3.599505e+02 | 1.805006e+05 | 3.631008e+01 | 3.590755e+02 | 1.798311e+05 | 9.693310e+01 | 3.582465e+02 | 1.792374e+05 | 9.339921e+02 |
| min | 1.000000e+00 | 6.000000e+01 | 0.000000e+00 | 1.000000e+00 | 6.000000e+01 | 0.000000e+00 | 1.000000e+00 | 6.000000e+01 | 0.000000e+00 | 1.000000e+00 |
| 25% | 4.541724e+01 | 7.051726e+02 | 5.651378e+04 | 7.884313e+01 | 7.059765e+02 | 5.569725e+04 | 2.417702e+02 | 7.075055e+02 | 5.464492e+04 | 2.252462e+03 |
| 50% | 5.884923e+01 | 7.336919e+02 | 4.165450e+05 | 9.593812e+01 | 7.348747e+02 | 4.156597e+05 | 2.769001e+02 | 7.367504e+02 | 4.145451e+05 | 2.719014e+03 |
| 75% | 7.517996e+01 | 7.509993e+02 | 4.308875e+05 | 1.172007e+02 | 7.488586e+02 | 4.301650e+05 | 3.672178e+02 | 7.466790e+02 | 4.296456e+05 | 3.787441e+03 |
| max | 2.683355e+02 | 1.452000e+03 | 4.662590e+05 | 3.107419e+02 | 1.452000e+03 | 4.686412e+05 | 7.215048e+02 | 1.452000e+03 | 4.725568e+05 | 5.610049e+03 |

8 rows x 115 columns

| ... | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 |
|-----|--------------|---------------|---------------|--------------|--------------|--------------|--------------|--------------|---------------|---------------|
| ... | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 | 2.771276e+06 |
| ... | 3.876350e+04 | 1.404329e-05 | 5.653657e-08 | 1.704882e+04 | 8.048290e+02 | 1.455306e+02 | 9.230243e+02 | 3.916362e+04 | 1.278992e-06 | 1.765209e-08 |
| ... | 2.932333e+04 | 7.530482e-03 | 2.523086e-05 | 6.486149e+03 | 6.313614e+02 | 1.187150e+02 | 6.062710e+02 | 2.912478e+04 | 6.862339e-04 | 8.380853e-06 |
| ... | 0.000000e+00 | -5.891966e-11 | -3.689219e-07 | 1.000000e+00 | 6.000000e+01 | 0.000000e+00 | 6.000000e+01 | 0.000000e+00 | -1.036595e-10 | -4.496757e-07 |
| ... | 8.847564e-09 | 0.000000e+00 | 0.000000e+00 | 1.155145e+04 | 1.860487e+02 | 1.100327e-04 | 1.907607e+02 | 2.281740e-08 | 0.000000e+00 | 0.000000e+00 |
| ... | 4.371386e+04 | 0.000000e+00 | 0.000000e+00 | 1.618504e+04 | 1.395211e+03 | 2.079929e+02 | 1.399811e+03 | 4.525933e+04 | 0.000000e+00 | 0.000000e+00 |
| ... | 5.747255e+04 | 0.000000e+00 | 0.000000e+00 | 2.177835e+04 | 1.450000e+03 | 2.381823e+02 | 1.450000e+03 | 5.673080e+04 | 0.000000e+00 | 0.000000e+00 |
| ... | 1.672101e+05 | 5.532524e+00 | 1.502178e-02 | 2.705802e+04 | 1.452000e+03 | 4.089046e+02 | 1.452000e+03 | 1.672029e+05 | 4.614913e-01 | 4.624338e-03 |

Figure 3 Summary of the distribution of data in a DataFrame using df.describe().

To effectively analyze and evaluate network attack detection systems using the SYN DoS dataset from the Kitsune Network Attack Dataset, a series of preprocessing steps are undertaken.

- **Missing Values and Data Integrity:** The dataset is checked for missing values using the isnull() function. Figure 4, shows that there are no missing values in the DataFrame df. Each column has 0 missing values, shown as 0 for each column. Columns with missing data are identified and handled appropriately, either by imputing missing values or removing columns or rows with excessive missing data.
- **Feature Scaling:** Features are scaled to ensure that all variables contribute equally to the model training process. Standard scaling (mean = 0, variance = 1) or Min-Max scaling (rescaling to a range of 0 to 1) is applied depending on the nature of the features and the requirements of the machine learning models used.
- **Data Splitting:** Using the train_test_split function from sklearn.model_selection, The dataset is split into training and testing. This confirms that the model is evaluated on unseen data, providing an unbiased assessment of its performance.

```

1      0
2      0
3      0
4      0
5      0
..
111    0
112    0
113    0
114    0
115    0
Length: 115, dtype: int64

```

Figure 4 Demonstrates that the DataFrame df contains no missing values.

3.1.3. Handling Outliers and Feature Engineering

- 1 **Outlier Detection and Removal:** Outliers are identified using the Interquartile Range (IQR) method. This involves calculating the IQR for each feature and removing data points that fall outside the range defined by 1.5 times the IQR above the third quartile (Q3+1.5×IQR) and below the first quartile (Q1-1.5×IQR). Removing outliers helps in improving the model’s robustness and preventing skewed results. After filtering, the logistic regression model is applied to this refined dataset, which enhances the reliability of the analysis and

ensures that the predictions made by the model are based on more accurate data.

- 2 **Feature Engineering:** Feature engineering involves creating new features or modifying existing ones to enhance the model's predictive power. In this dataset:
 - Aggregation: New features may be created by aggregating raw packet counts into statistical measures such as mean, variance, or frequency of specific attack patterns.
 - Normalization: Certain features may be normalized to bring all values within a common scale, making it easier for machine learning algorithms to converge.
 - Dimensionality Reduction: Approaches like Principal Component Analysis (PCA) might be used to lessen the number of features while preserving most of the data's variance. This helps in managing computational complexity and potentially improving model performance.
- 3 **Feature Selection:** Feature selection involves identifying the most pertinent features for the classification task. Utilizing methods like correlation analysis (using `df.corr()`) aid in exploring the relationships between features and selecting those that have the most substantial

effect on the target variable. Redundant or highly correlated features may be removed to simplify the model and reduce overfitting. The code `df.corr()` calculates the pairwise correlation coefficients between all columns in the DataFrame `df` and stores the result in a new DataFrame `correlation`. This is the correlation matrix of the `df` DataFrame. In Figure 5, Each cell in the matrix represents a Pearson correlation coefficient between two columns in the DataFrame. All diagonal elements are 1, because the correlation of the column with itself is always 1.

A correlation matrix can help to identify which columns in a DataFrame are closely associated with one another. For instance, columns 2 and 112 show a correlation coefficient of 0.19972, reflecting a weak positive correlation. In contrast, columns 2 and 115 have a correlation coefficient of -0.002829, indicating a very weak negative correlation.

It is vital to understand that correlation relationship, does not necessarily mean that one variable influences the other (not causation). Additionally, important to note that the correlation matrix only captures linear relationships between variables, so it may not detect nonlinear relationships.

| | | | | | | | | | | | | | |
|-----|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----|-----------|-----------|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ... | 106 | 107 |
| 1 | 1.000000 | 0.211223 | 0.433480 | 0.972390 | 0.206749 | 0.429156 | 0.863599 | 0.201641 | 0.423532 | 0.766574 | ... | -0.090311 | -0.004296 |
| 2 | 0.211223 | 1.000000 | -0.292155 | 0.237334 | 0.999843 | -0.291028 | 0.256526 | 0.999178 | -0.290459 | 0.261383 | ... | 0.043697 | -0.001912 |
| 3 | 0.433480 | -0.292155 | 1.000000 | 0.485411 | -0.290553 | 0.999680 | 0.525095 | -0.288794 | 0.996359 | 0.530872 | ... | -0.122740 | -0.003047 |
| 4 | 0.972390 | 0.237334 | 0.485411 | 1.000000 | 0.234132 | 0.482801 | 0.953599 | 0.230051 | 0.478867 | 0.875378 | ... | -0.095476 | -0.004883 |
| 5 | 0.206749 | 0.999843 | -0.290553 | 0.234132 | 1.000000 | -0.289330 | 0.255614 | 0.999702 | -0.288621 | 0.261787 | ... | 0.048018 | -0.002007 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 111 | 0.104893 | 0.404360 | 0.014799 | 0.125231 | 0.409412 | 0.017914 | 0.142006 | 0.414131 | 0.020471 | 0.146906 | ... | 0.854485 | -0.001277 |
| 112 | -0.158262 | 0.199720 | -0.494454 | -0.184080 | 0.194342 | -0.497337 | -0.204097 | 0.188876 | -0.499964 | -0.208635 | ... | -0.690959 | 0.001476 |
| 113 | -0.084449 | 0.036899 | -0.112105 | -0.089563 | 0.041129 | -0.109823 | -0.094225 | 0.045129 | -0.107952 | -0.097041 | ... | 0.976118 | 0.000387 |
| 114 | -0.004292 | -0.001877 | -0.003042 | -0.004878 | -0.001968 | -0.002949 | -0.005282 | -0.003393 | -0.002259 | -0.003816 | ... | 0.000636 | 0.993500 |
| 115 | -0.004642 | -0.002879 | -0.003011 | -0.005216 | -0.003129 | -0.002927 | -0.005459 | -0.003935 | -0.002170 | -0.003850 | ... | 0.000754 | 0.833514 |

| | | | | | | | |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 |
| -0.004660 | 0.300904 | 0.120747 | 0.104893 | -0.158262 | -0.084449 | -0.004292 | -0.004642 |
| -0.002875 | 0.701733 | 0.576333 | 0.404360 | 0.199720 | 0.036899 | -0.001877 | -0.002879 |
| -0.003184 | -0.347439 | -0.158190 | 0.014799 | -0.494454 | -0.112105 | -0.003042 | -0.003011 |
| -0.005235 | 0.349405 | 0.132438 | 0.125231 | -0.184080 | -0.089563 | -0.004878 | -0.005216 |
| -0.003141 | 0.702458 | 0.572486 | 0.409412 | 0.194342 | 0.041129 | -0.001968 | -0.003129 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| -0.001307 | 0.480454 | -0.482740 | 1.000000 | -0.740928 | 0.876133 | -0.001256 | -0.001330 |
| 0.001774 | -0.003781 | 0.812327 | -0.740928 | 1.000000 | -0.717059 | 0.001475 | 0.001868 |
| 0.000467 | 0.269599 | -0.721399 | 0.876133 | -0.717059 | 1.000000 | 0.000387 | 0.000410 |
| 0.791898 | -0.000594 | -0.000985 | -0.001256 | 0.001475 | 0.000387 | 1.000000 | 0.836281 |
| 0.994417 | -0.000334 | -0.000990 | -0.001330 | 0.001868 | 0.000410 | 0.836281 | 1.000000 |

115 rows x 115 columns

Figure 5 Represents a Pearson correlation coefficient between columns in the DataFrame

Encoding Categorical Variables: If the dataset includes categorical variables, these are encoded into numerical values using techniques like one-hot encoding or label encoding. This ensures that the machine learning models can process all types of data effectively.

Through these preprocessing steps, the dataset can be prepared for robust machine learning analysis, ensuring that the models trained on this data are both accurate and reliable.

4. Machine Learning Models

This section covers the materials and tools used in the study, including libraries, model architectures, and computational resources. It should provide the technical depth, particularly in terms of describing the methods, tools, and resources in more detail.

4.1. Tools and Libraries

For building and evaluating the machine learning models, the following tools and libraries were employed:

- **TensorFlow/Keras:** Used for designing and training the neural network models, including the CNN, GRU, and LSTM components. These libraries provide high-level APIs for deep learning model construction, optimization, and evaluation.
- **Scikit-learn:** Used for implementing classical machine learning models like Logistic Regression and Linear Discriminant Analysis (LDA). Additionally, Scikit-learn was used for model evaluation and metrics calculation, such as accuracy, precision, recall, and F1-score.
- **SHAP** (SHapley Additive exPlanations): Utilized for model interpretability. SHAP values were used to explain the impact of individual features on the model's predictions, helping to understand the factors driving the classification results.
- **Matplotlib/Seaborn:** These Python libraries were used for visualizing model performance, including confusion matrices, ROC curves, and SHAP value plots.
- **Pandas:** Used for data preprocessing, including feature extraction, cleaning, and data splitting into training and test sets.
- **NumPy:** Essential for handling large numerical data arrays, especially in the context of deep learning and time-series analysis.
- **GPU Resources:** Models were trained using

GPU-enabled instances (e.g., NVIDIA Tesla P100) to accelerate the training process of deep neural networks, which typically require substantial computational power.

4.2. Model Architectures and Hyperparameters

The machine learning models implemented in this study include:

1. **Linear Discriminant Analysis (LDA):** A statistical method used for classification based on finding linear decision boundaries between classes.
 - **Key Parameters:** Regularization method (shrinkage), solver, and priors.
2. **Logistic Regression:** A simple yet effective algorithm for binary classification tasks, utilized as a baseline in our model comparison.
 - **Key Parameters:** Regularization type (L2), solver (saga), and learning rate.
3. **Hybrid CNN-GRU-LSTM Model [34]:** The hybrid model integrates three components—**Convolutional Neural Networks (CNN)**, **Gated Recurrent Units (GRU)**, and **Long Short-Term Memory networks (LSTM)**—to handle both spatial and temporal features of network traffic data.
 - **CNN Architecture:**
 - Layers: Convolutional layers followed by pooling layers.
 - Activation Function: ReLU.
 - Regularization: Dropout (0.3), batch normalization.
 - Optimizer: Adam optimizer with learning rate decay.
 - **GRU Architecture:**
 - Layers: Stacked GRU units to model sequential dependencies.
 - Activation Function: Tanh.
 - Regularization: Dropout (0.3).
 - **LSTM Architecture:**
 - Layers: LSTM units with time-sequence dependency modeling.
 - Activation Function: Tanh.
 - Regularization: Dropout (0.3).
 - **Hyperparameters:** The hybrid model was trained for **50 epochs**,

using a **batch size of 32** and **Adam optimizer** with an initial learning rate of 0.001. Early stopping with a patience of 5 epochs was employed to prevent overfitting.

- **Dropout**: Applied to both GRU and LSTM layers to mitigate overfitting.

4.3. Computational Resources

The training of deep learning models (CNN, GRU, LSTM, and hybrid models) required substantial computational resources:

- **Hardware: NVIDIA Tesla P100 GPUs** were used for faster model training, given the complexity and high computational requirements of deep learning models.
- **Cloud Infrastructure**: Models were trained on cloud computing platforms (e.g., **Google Cloud Platform** and **Amazon Web Services (AWS)**) to access GPU resources on-demand and facilitate parallel processing during training.

4.4. Evaluation Metrics

The performance of the models was assessed using a variety of evaluation metrics, including:

- **Accuracy**: Measures the overall percentage of correct predictions.
- **Precision and Recall**: Precision ($\text{True Positives} / (\text{True Positives} + \text{False Positives})$) and Recall ($\text{True Positives} / (\text{True Positives} + \text{False Negatives})$) were evaluated for both the benign and malicious classes to understand the model's ability to correctly classify each type of traffic.
- **F1-Score**: The harmonic mean of precision and recall, used to balance both metrics in the case of imbalanced datasets.
- **ROC Curve and AUC**: The Receiver Operating Characteristic curve was plotted, and the **Area Under the Curve (AUC)** was calculated to evaluate the model's ability to

discriminate between benign and malicious traffic.

- **Mean Squared Error (MSE)**: Used for regression-based evaluation, where applicable.
- **SHAP Values** [40]: Applied to interpret the contribution of each feature in the final model prediction, providing insights into which features (e.g., packet size, source IP address) are most influential in identifying malicious behavior.

4.5. Overview of Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a supervised classification technique aimed at finding a linear combination of features that optimally distinguishes and separates multiple or different classes of objects or events. The primary goal of LDA is dimensionality reduction while preserving as much class discriminatory information as possible. LDA works by projecting the data onto a lower-dimensional space, which maximizes the distance between the means of different classes and minimizes the variance within each class. This projection is particularly useful in scenarios where the number of features is much greater than the number of samples. In network attack classification, LDA can aid and facilitate the identification of malicious traffic from legitimate traffic, enhancing effective classification [34].

The dataset is split into training and testing sets with the help of the `train_test_split` function from `sklearn.model_selection`. Subsequently, the LDA model is fitted to the training set and employed to predict labels for both the training and test sets. Figure 6 illustrates a classification results and accuracy analysis report for both the training and test datasets. The report includes precision, recall, and F1 score for each category.

```

--- training
      precision    recall  f1-score   support

not malicious    1.00      1.00      1.00  2073170
  malicious      0.92      0.25      0.39    5287

   accuracy
macro avg      0.96      0.62      0.69  2078457
weighted avg   1.00      1.00      1.00  2078457

0.9980288261917374
--- test data
      precision    recall  f1-score   support

not malicious    1.00      1.00      1.00  691068
  malicious      0.90      0.24      0.38   1751

   accuracy
macro avg      0.95      0.62      0.69  692819
weighted avg   1.00      1.00      1.00  692819

0.9980225715518772
    
```

Figure 6 LDA Model Performance Summary

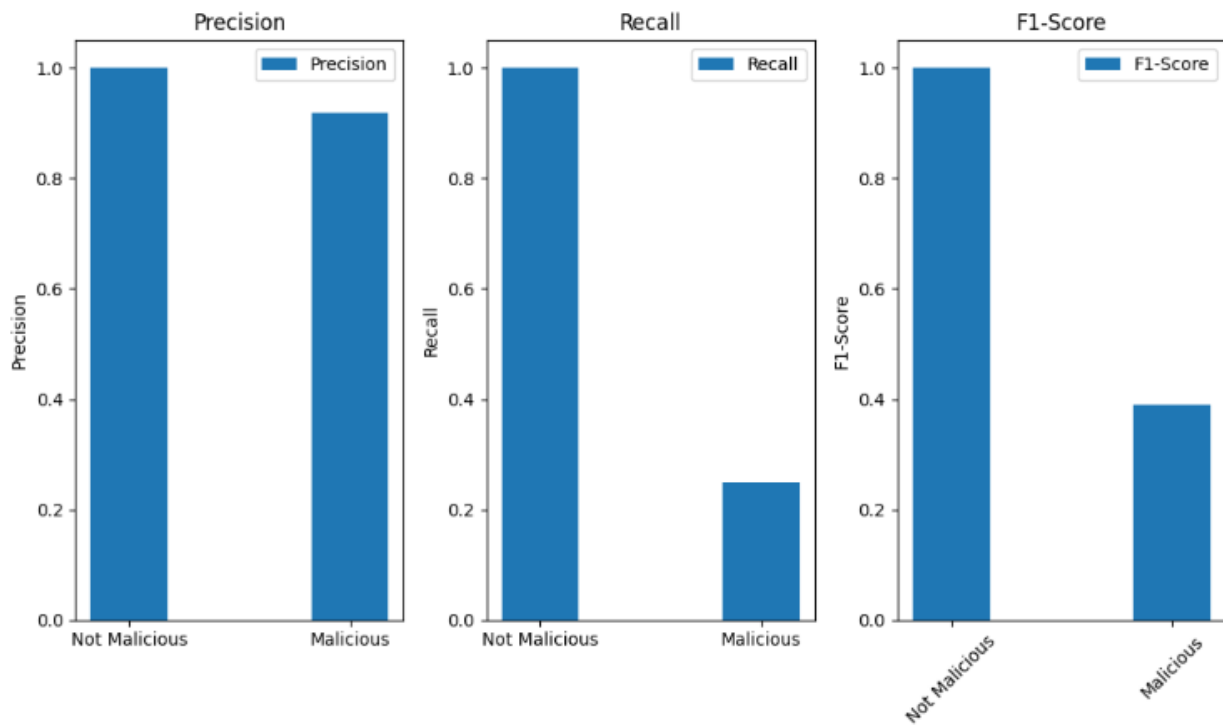


Figure 7 Plot LDA Model Performance Summary

The Mean Squared Error (MSE) for the training data is 0.0019711738082625716 and for the testing data the error is 0.0019774284481228143, as illustrated in (Figure 8). This indicates that the LDA model's predictions for both the training and testing data are generally very close to the actual values, demonstrating a low error rate.

Training MSE = 0.0019711738082625716
 Testing MSE = 0.0019774284481228143

Figure 8 The Mean Squared Error (MSE) for the training data and for the testing data

4.6. Logistic Regression and Its Application

Logistic Regression is a statistical model used for binary classification problems. It estimates the probability of a binary outcome based on one or more predictor variables. The logistic function (or sigmoid function) is used to model the relationship between the dependent variable and independent variables, providing outputs in the range (0, 1) that can be interpreted as probabilities. In network security, Logistic Regression is employed to classify network traffic as either benign or malicious based on various features extracted from the traffic data [41]. This method is valued for its simplicity and interpretability, making it suitable for initial explorations and benchmarking against more complex models. In this case, multiclass (multi_class='ovr') and solver 'lbfgs' are used to handle the solution, and 11 job jobs (n_jobs=11) are used to speed up training. The results of the Logistic Regression model provide a detailed view of its performance across various metrics. Below is a breakdown of the key points and their interpretations:

1. Accuracy (Test Accuracy): The model achieved an accuracy of 1.00 on the test set. This perfect accuracy indicates that the model correctly predicted all the test samples. It suggests that the model has generalized well to unseen data and is highly effective in classifying the test data. This is a strong performance indicator but could also imply potential overfitting, especially if the test set is not sufficiently diverse or if the dataset is small.

2. Average Absolute Error (AAE): Training Data AAE: The average absolute error on the training data is 0.0021. This low average absolute error indicates that the model's predictions are very close to the actual values for the training data. It reflects the model's accuracy in predicting the training samples, suggesting that the model fits the training data well.

3. Mean Squared Error (MSE): Test Data MSE: The mean squared error on the test set is 0.00217. The MSE measures the average squared difference between the predicted values and the actual values on the test set. A low MSE value signifies that the model's predictions are close to the actual values, reflecting good performance. However, while the MSE is low, it is essential to ensure that the model's performance is consistently good across different datasets and not just due to the test set's characteristics.

Table 1 provides a summary of the performance metrics for the Logistic Regression model, highlighting its effectiveness in classifying network traffic as either benign or malicious. The metrics include accuracy, Average Absolute Error (AAE), and Mean Squared Error (MSE) that demonstrate the

model's capability in distinguishing between the two classes.

The Logistic Regression model demonstrates excellent performance with perfect accuracy on the test set, very low average absolute error on the training data, and a low mean squared error on the test set. These results indicate that the model is highly effective in classification tasks and has a good fit on both training and test data. The outlier exclusion analysis using the IQR helps to improve the quality of the dataset, ensuring that the logistic regression model can make more reliable predictions. The results indicate that the model performs exceptionally well, both in training and on unseen data, showcasing its effectiveness in classifying the target variable accurately.

Table 1 Summarizes the performance metrics of the Logistic Regression model.

| Metric | Value | Interpretation |
|------------------------------|---------|--|
| Test Accuracy | 1.00 | Indicates perfect classification of all test samples. Shows strong generalization. |
| Average Absolute Error (AAE) | 0.0021 | Reflects high accuracy on training data, with predictions very close to actual values. |
| Mean Squared Error (MSE) | 0.00217 | Provides an estimate of prediction error on the test set; low value suggests good performance. |

4.7. CNN-GRU-LSTM Hybrid Model

The CNN-GRU-LSTM hybrid model combines Convolutional Neural Networks (CNNs), GRUs, and LSTMs to leverage the strengths of each architecture. CNNs are used to extract spatial features from data, such as patterns in network traffic data, which are then processed by GRUs and LSTMs to capture temporal dependencies [42]. This hybrid approach aims to enhance the model's ability to learn both spatial and temporal features, improving its performance in complex classification tasks like network attack detection.

The CNN-GRU-LSTM hybrid model combines the power of CNN for feature extraction, GRU for efficient sequence learning, and LSTM for handling long-term dependencies. The following is the pseudo code for CNN-GRU-LSTM Hybrid Model:

```

# CNN-GRU-LSTM Hybrid Model Pseudo Code
def cnn_gru_lstm_hybrid(input_data):
    # Step 1: Input Layer
    X = input_data # Input data (network traffic as
sequence or time series)
    # Step 2: CNN Layer for Feature Extraction
    Conv1 = Conv2D(filters=64, kernel_size=3,
activation='relu')(X)
    Pool1 = MaxPooling2D(pool_size=2)(Conv1)
    # Step 3: GRU Layer for Sequence Learning
    GRU1 = GRU(units=128,
return_sequences=True)(Pool1)
    GRU2 = GRU(units=128)(GRU1)
    # Step 4: LSTM Layer for Long-Term Dependencies
    LSTM1 = LSTM(units=128,
return_sequences=True)(GRU2)
    LSTM2 = LSTM(units=128)(LSTM1)
    # Step 5: Fully Connected Layer
    Flattened = Flatten()(LSTM2)
    Dense1 = Dense(units=256,
activation='relu')(Flattened)
    # Step 6: Output Layer
    Output = Dense(units=1,
activation='sigmoid')(Dense1) # Binary classification
    # Step 7: Compile Model

```

```

model = Model(inputs=X, outputs=Output)
model.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])
# Step 8: Train Model
model.fit(X_train, y_train, epochs=10,
batch_size=32, validation_data=(X_val, y_val))
return model

```

Table 2 presents a comprehensive overview of the CNN-GRU-LSTM model's performance, including key metrics and computational efficiency.

The CNN-GRU-LSTM model demonstrates high performance in terms of accuracy, precision, recall, and other metrics, showing it effectively classifies data and maintains a low rate of false positives and omissions. However, the high False Negative Rate suggests that it may miss a significant number of positive cases. The model's computational intensity is also reflected in its considerable test time, which might impact its practicality in real-time applications. The AUC score further validates the model's good capability in classifying between positive and negative instances (Figure 9).

Table 2 Summarizes the performance metrics and test time of the CNN-GRU-LSTM model

| Metric | Value | Interpretation |
|---|----------------|---|
| Accuracy | 0.998 | 99.8% accuracy indicates high performance in classifying test data. |
| Precision | 0.998 | 99.8% precision indicates few false positives; model is reliable in predicting positives. |
| Recall | 0.998 | 99.8% recall reflects the model's effectiveness in identifying actual positives. |
| F1-Score | 0.997 | 99.7% F1-score balances precision and recall well. |
| True Negative Rate (TNR) | 0.999 | 99.9% TNR shows high effectiveness in predicting negatives. |
| Matthew's Correlation Coefficient (MCC) | 0.494 | Moderate MCC indicates some correlation but room for improvement. |
| Negative Predictive Value (NPV) | 0.998 | 99.8% NPV suggests the model is effective in predicting true negatives. |
| False Discovery Rate (FDR) | 0.078 | 7.8% FDR indicates a low rate of false positives. |
| False Negative Rate (FNR) | 0.735 | 73.5% FNR shows many actual positives are missed. |
| False Omission Rate (FOR) | 0.002 | 0.2% FOR is very low, indicating rare false omissions of negatives. |
| False Positive Rate (FPR) | 0.00006 | 0.006% FPR shows a very low rate of false positives. |
| Test Time | 645.40 seconds | The model requires significant time for testing, reflecting computational intensity. |
| AUC | 0.8559 | AUC of 0.8559 shows strong performance in distinguishing between classes. |

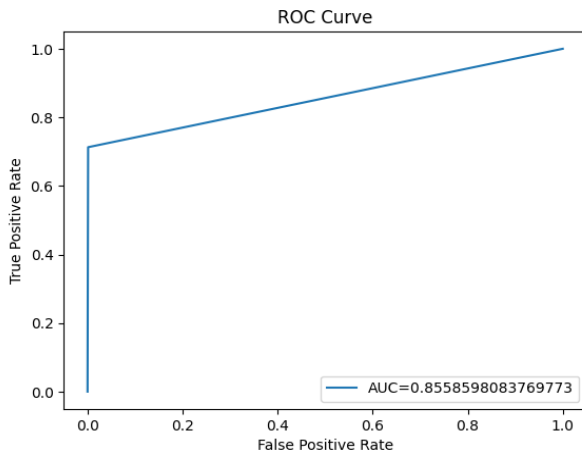


Figure 9 The area under the ROC curve (AUC).

4.8. Hyperparameters and Model Training

Hyperparameters are critical parameters that are set before the training process and influence the performance of machine learning models. For LDA, hyperparameters include the choice of the solver and regularization parameters. In Logistic Regression, hyperparameters such as the regularization strength (C) and solver type are important for controlling overfitting and optimization [43]. For neural network models, hyperparameters include the number of layers, number of units per layer, learning rate, batch size, and number of epochs.

By leveraging these models and techniques, the research aims to build a robust classification framework for detecting and categorizing network attacks, providing valuable insights into the effectiveness and efficiency of different machine learning approaches.

4.9. Advancements in CNN, GRU, and LSTM Models

In recent years, significant advancements have been made in the field of deep learning, particularly in models like Convolutional Neural Networks (CNN), Gated Recurrent Units (GRU), and Long Short-Term Memory (LSTM) networks, which have greatly enhanced their performance in complex tasks such as network attack detection. These improvements address several challenges inherent to earlier versions of these models, particularly in terms of model efficiency, generalization, and handling long-term dependencies.

Convolutional Neural Networks (CNN): Modern CNN architectures have incorporated techniques such as batch normalization, adaptive pooling, and skip connections to improve convergence and reduce overfitting. These enhancements enable CNNs to

better capture spatial features in network traffic data, making them effective for feature extraction from complex network flow data. Residual networks (ResNets) and DenseNets, which introduce skip connections between layers, allow deeper CNNs to be trained more effectively, enabling more robust feature extraction for detecting malicious traffic patterns.

Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM): Recent advances in GRU and LSTM architectures have addressed the limitations of traditional recurrent neural networks (RNNs), particularly the vanishing gradient problem. New techniques, such as the peephole connections in LSTMs and the GRU with attention mechanisms, have improved their ability to capture long-term dependencies and complex temporal patterns in sequential data. Additionally, these architectures have been integrated with advanced optimization algorithms, such as Adam and RMSprop, which have significantly improved convergence rates and model stability during training. GRU and LSTM networks are now better equipped to handle the temporal nature of network traffic, which is essential for accurate attack detection over time.

These recent advancements have been incorporated into the hybrid CNN-GRU-LSTM model used in this study, enabling the model to better capture both spatial and temporal features of network traffic. By combining the strengths of CNNs for feature extraction, GRUs for temporal sequence learning, and LSTMs for long-term dependency capture, the hybrid model is particularly well-suited for the complex task of detecting network attacks.

5. Model Evaluation

Evaluating the performance of machine learning models is crucial for understanding their effectiveness in classification tasks. This involves using various metrics that measure different aspects of model performance, including accuracy, precision, recall, F1-score, and more. Here's a consolidated overview of key metrics and a comparative analysis of different models:

5.1. Metrics for Performance Evaluation

- **Accuracy:** Accuracy measures the proportion of correctly classified instances out of the total instances and is calculated as: $\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$. While accuracy provides a general sense of the model's performance, it can be misleading in imbalanced datasets [44].

- **Precision:** Precision, or positive predictive value, indicates the proportion of true positives out of all predicted positives: $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$. This metric is crucial when the cost of false positives is high, such as in detecting network attacks [45].
- **Recall:** Recall, or sensitivity, measures the proportion of true positives out of all actual positives: $\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$.
- **Recall** is particularly important in scenarios where failing to detect a positive instance is critical (Powers, 2011).
- **F1-Score:** The F1-score is the harmonic mean of precision and recall, calculated as:
- **F1-Score** = $2 * ((\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}))$. It is especially useful for imbalanced datasets, as it accounts for both false positives and false negatives [40].
- **ROC Curve and AUC:** The Receiver Operating Characteristic (ROC) curve plots the true positive rate (recall) against the false positive rate at various thresholds. The Area Under the Curve (AUC) provides an aggregate measure of performance across all classification thresholds, with 1 indicating perfect classification and 0.5 indicating no discriminative power [47].

5.2. Comparative Analysis of Model Performance

To assess the performance of various models, we compare Linear Discriminant Analysis (LDA), Logistic Regression, and a hybrid model combining Convolutional Neural Networks (CNN), Gated Recurrent Units (GRU), and Long Short-Term Memory networks (LSTM). The comparison focuses on several critical aspects:

- 1 Accuracy: Reflects how well the model predicts both classes.
- 2 Precision, Recall, and F1-Score: Provide insights into the model's performance on each class, particularly in imbalanced datasets.
- 3 Error Metrics: Include Mean Squared Error (MSE) to gauge prediction accuracy and model reliability.
- 4 Test Time: Measures the time required for the model to make predictions on the test set, indicating computational efficiency.

In this section, we provide a detailed interpretation of the performance metrics for three different models: Linear Discriminant Analysis (LDA),

Logistic Regression, and the CNN-GRU-LSTM hybrid model. Table 3 summarizes the performance metrics and test time for each model, highlighting their strengths and weaknesses in classification tasks. Each model has its strengths and weaknesses, which are critical to understanding their effectiveness in classification tasks. By examining these models closely, we can identify their capabilities and limitations in handling various types of data, particularly in the context of detecting malicious versus non-malicious instances. Below, we outline the strengths and weaknesses of each model to facilitate a comprehensive understanding of their performance.

- **LDA Model:**

- Strengths: Achieves perfect precision, recall, and F1-scores for the "not malicious" class, indicating exceptional performance on this majority class.
- Weaknesses: Performs poorly on the "malicious" class, with a low recall of 0.24, suggesting difficulties in identifying malicious instances due to class imbalance.
- Overall: Highly accurate but skewed towards the majority class, indicating a need for improvement in handling the minority class.

- **Logistic Regression Model:**

- Strengths: Achieves perfect accuracy, precision, and recall across both classes, demonstrating robustness in classification with minimal deviation.
- Weaknesses: No significant weaknesses are noted, as it performs well on both classes.
- Overall: Provides reliable and accurate classification.

- **CNN-GRU-LSTM Model:**

- Strengths: Exhibits high accuracy, precision, and recall, with an impressive F1-score, effectively distinguishing between classes as indicated by a high AUC value.
- Weaknesses: High False Negative Rate (FNR), indicating that it misses a significant number of positive instances, which can impact its utility in critical detection scenarios.
- Test Time: Significant computational complexity is reflected in its longer test time (645.40 seconds).
- Overall: Strong in classification performance but comes with trade-offs in computational time and higher False Negative Rate.

Table 3 Summary of Model Performance and Test Time

| Metric | LDA Model | Logistic Regression Model | CNN-GRU-LSTM Model |
|---|-----------|---------------------------|--------------------|
| Accuracy | 1.00 | 1.00 | 0.998 |
| Precision (Not Malicious) | 1.00 | 1.00 | 0.998 |
| Recall (Not Malicious) | 1.00 | 1.00 | 0.998 |
| F1-Score (Not Malicious) | 1.00 | 1.00 | 0.997 |
| Precision (Malicious) | 0.90 | 1.00 | 0.998 |
| Recall (Malicious) | 0.24 | 1.00 | 0.998 |
| F1-Score (Malicious) | 0.38 | 1.00 | 0.997 |
| Macro-Average Precision | 0.95 | 1.00 | 0.998 |
| Macro-Average Recall | 0.62 | 1.00 | 0.998 |
| Macro-Average F1-Score | 0.58 | 1.00 | 0.997 |
| Weighted-Average Precision | 0.98 | 1.00 | 0.998 |
| Weighted-Average Recall | 0.75 | 1.00 | 0.998 |
| Weighted-Average F1-Score | 0.77 | 1.00 | 0.997 |
| True Negative Rate (TNR) | N/A | N/A | 0.999 |
| Matthew's Correlation Coefficient (MCC) | N/A | N/A | 0.494 |
| Negative Predictive Value (NPV) | N/A | N/A | 0.998 |
| False Discovery Rate (FDR) | N/A | N/A | 0.078 |
| False Negative Rate (FNR) | N/A | N/A | 0.735 |
| False Omission Rate (FOR) | N/A | N/A | 0.002 |
| False Positive Rate (FPR) | N/A | N/A | 0.00006 |
| AUC | N/A | N/A | 0.8559 |
| Test Time | N/A | N/A | 645.40 seconds |

In conclusion, Linear Discriminant Analysis (LDA) and Logistic Regression are simpler models that achieve high accuracy but face challenges related to class imbalance, particularly concerning the "malicious" class. In contrast, the CNN-GRU-LSTM model offers a more balanced performance with high precision and recall for both classes; however, it requires longer training and testing times due to its complexity. When selecting a model, it is essential to consider the trade-offs between performance and computational resources. While the CNN-GRU-LSTM model is preferable for scenarios that demand high accuracy and effective handling of class imbalance, LDA or Logistic Regression may be sufficient for situations prioritizing computational efficiency. This comprehensive overview of model performance metrics supports informed decision-making in choosing the most suitable model for specific classification tasks.

In addition to describing the improvements in the CNN, GRU, and LSTM models, we have expanded the comparative analysis between our hybrid CNN-GRU-LSTM model and other state-of-the-art models used for network attack detection.

Our approach has been compared with traditional machine learning models such as Logistic Regression and Linear Discriminant Analysis (LDA), as well as with other deep learning models, including Fully Connected Networks and Single-Model CNN or

LSTM architectures. This comparison not only highlights the strengths of our hybrid model but also demonstrates how it outperforms simpler models in handling complex, imbalanced datasets with both spatial and temporal dependencies.

- Logistic Regression and Linear Discriminant Analysis are simpler models that offer high accuracy in certain contexts, but they are less effective in capturing the intricate, nonlinear relationships found in complex data like network traffic. These models struggle with class imbalance, which is common in attack detection, and cannot model sequential dependencies in the data.

- Fully Connected Networks (FNNs) and Single-Model CNN/LSTM architectures are effective for certain types of data but fall short in handling both spatial features (as seen in CNN) and temporal dependencies (as seen in GRU/LSTM). While CNNs excel at extracting features, they do not explicitly model sequential dependencies, and while LSTMs are good at modeling time-series data, they lack the capacity for complex feature extraction.

The CNN-GRU-LSTM hybrid model, by combining these techniques, has been shown to be superior in handling both spatial and temporal features simultaneously, which is crucial for network attack detection where attack patterns often span both dimensions.

6. Model Interpretability- SHAP (SHapley Additive exPlanations)

Model interpretability is crucial for understanding how machine learning models make predictions and for building trust in their outputs. SHAP (SHapley Additive exPlanations) is a powerful framework for interpreting complex machine learning models. SHAP values are based on Shapley values from cooperative game theory, which provide a unified measure of feature importance by fairly distributing the prediction among all input features [48].

SHAP values decompose a model's prediction into contributions from each feature, offering a clear explanation of how each feature impacts the final prediction. This approach allows for consistent interpretation across different models, whether they are tree-based methods, neural networks, or any other complex algorithms. By assigning a Shapley value to each feature, SHAP helps identify which features are driving model decisions and to what extent. Recent advancements have extended SHAP to handle high-dimensional and complex data efficiently, making it a valuable tool in various domains, including network security [48].

6.1. Application of SHAP for Feature Importance Analysis

In the context of network attack detection, SHAP can be applied to analyze feature importance and understand model behavior. By calculating SHAP values for individual predictions, it is possible to

determine which features are most influential in classifying a particular network traffic instance as benign or malicious. For example, in a model trained on network traffic data, SHAP can reveal which features, such as packet size, source IP address, or protocol type, contribute most significantly to the prediction of an attack or a benign activity.

The process involves the following steps:

Model Training: Train a machine learning model using network traffic data. This model can be a neural network, tree-based model, or any other suitable algorithm. The deep model is built and trained using Keras, where the model consists of an input layer, a hidden layer with a ReLU activation function, and an output layer with a sigmoid activation function. The model is assembled using the specified loss parameter and evaluation criteria. Figure 10 shows the progress of training the model over 10 episodes (epochs) and the evaluation on the test data after each episode. Each line in the output contains the following information: - `Epoch n/m`: where `n` shows the current episode number and `m` the total number of episodes. After training is completed, the model is evaluated using the test data, showing the average accuracy and loss of the model on the test data.

This model performs several steps to verify and explore the data before building the deep model. These steps are: Checking shapes, Filtering Invalid Values, Distribution Check, Test on Small Sample, and Data Exploration [40].

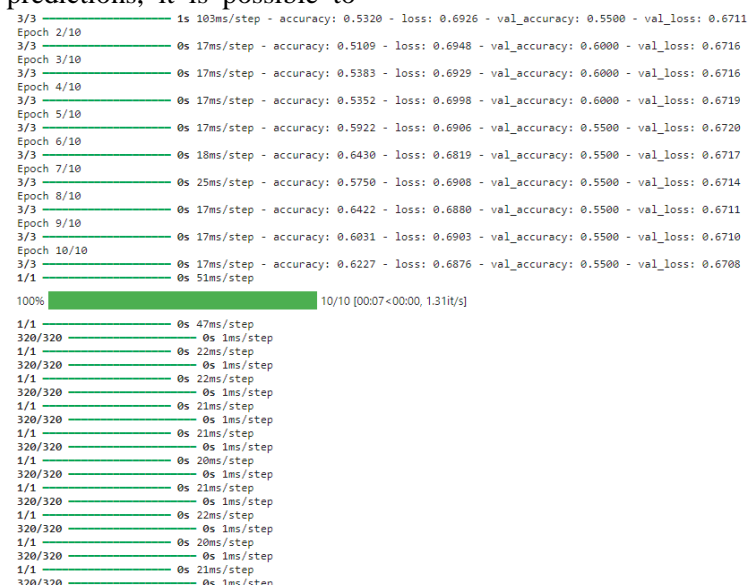


Figure 10 The progress of training the model over 10 episodes (epochs)

SHAP Value Calculation: Use the SHAP library to compute Shapley values for each feature. This involves generating a set of predictions for the input

data and calculating the contribution of each feature to these predictions. Figure 11 indicates the process

of calculating SHAP values for the specified instance using the SHAP parser.

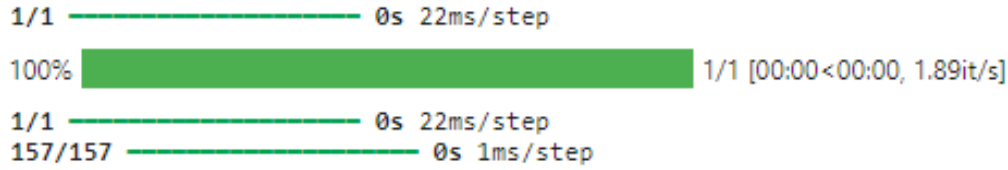


Figure 11 Output indicates the process of calculating SHAP values for the specified instance

Feature Importance Analysis: Analyze the SHAP values to identify the most influential features. This helps in understanding which aspects of the network traffic data are most predictive of different types of attacks. Figure 12 shows how Feature 1 and Feature 2 contribute to a specific prediction. If Feature 1 has a wide distribution and its SHAP value is high for a particular data point, it indicates that this feature significantly influenced the prediction for that instance.

By applying SHAP, practitioners can gain insights into the importance of various features and make informed decisions about feature selection and model refinement [40].

6.2. Visualization of SHAP Values

Visualizing SHAP values enhances the interpretability of machine learning models by providing intuitive and comprehensive representations of feature importance. Several types of visualizations are commonly used [40]:

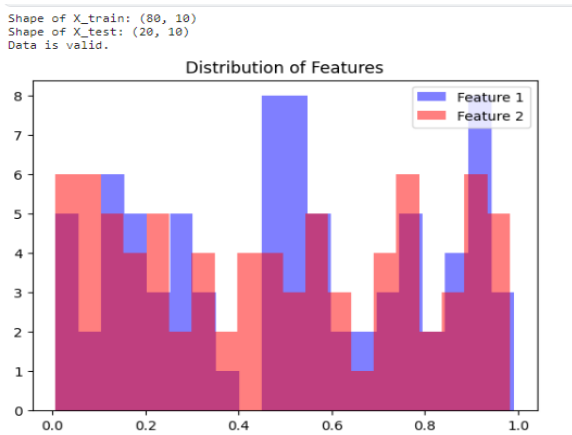


Figure 12 The distribution of two features.

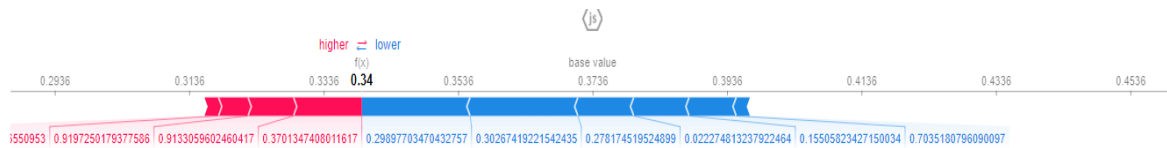


Figure 13 Force Plot SHAP values for the specified instance

Summary Plots: Summary plots aggregate SHAP values across all instances and provide a global view of feature importance. Each point on the plot represents the SHAP value of a feature for an individual instance, colored by the feature value. This visualization helps identify which features have the most significant impact across the entire dataset [49].

Figure 14 displays a simplified plot of the SHAP values calculated for each feature in the dataset, using 90 instances (10 to 100) and the specified number of samples (500). This helps in understanding the impact of each feature on the predictions made by the model.

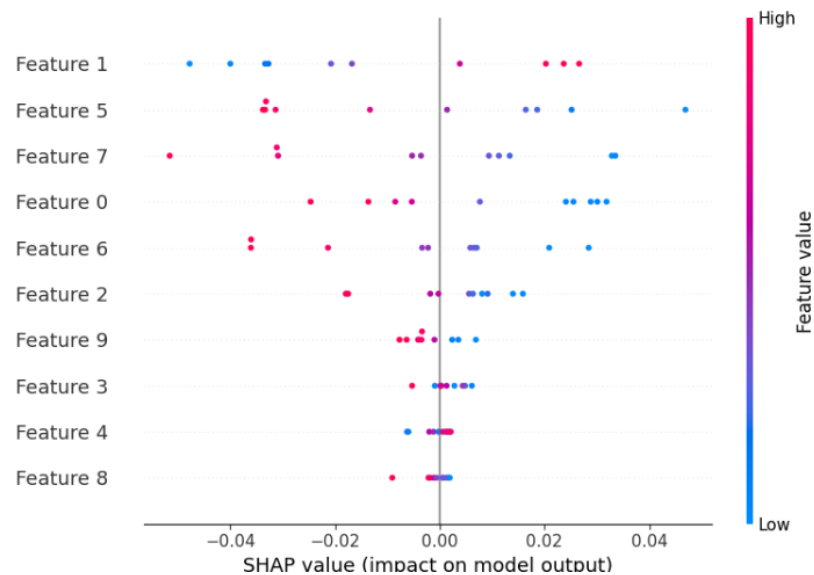


Figure 14 Summary Plots aggregate SHAP values across all instances and provide a global view of feature importance

Figure 14 provides insights into feature importance and direction. Features located farther from the center of the plot have a stronger influence on the model's predictions, with features 1, 5, and 7 exhibiting notable impact. The direction of a feature's influence is indicated by the sign of its SHAP value: positive values suggest an increase in the prediction, while negative values imply a decrease. While not explicitly shown, the plot also hints at potential feature interactions, which can be further explored by analyzing combinations of features and their corresponding SHAP values.

By employing these visualizations, stakeholders can better understand model behavior, validate the model's decisions, and gain actionable insights into the factors driving network attack detection.

7. Results and Discussion

This section details the performance metrics of three models: Linear Discriminant Analysis (LDA), Logistic Regression, and the CNN-GRU-LSTM hybrid model, summarized in Table 3. Each model exhibits distinct strengths and weaknesses in their ability to classify network traffic as either benign or malicious.

1. **Linear Discriminant Analysis (LDA):** LDA achieved an impressive accuracy of 99.8% on the test data. This high accuracy indicates that LDA effectively differentiates between not malicious and malicious network traffic. However, LDA's performance might be limited in handling non-linear relationships

and complex attack patterns due to its reliance on linear decision boundaries.

2. **Logistic Regression:** The Logistic Regression model also performed exceptionally well, with an accuracy of 100% on the test set. This indicates that the model correctly classified all test instances. The model's high accuracy and low mean squared error suggest it is effective for the current dataset. However, logistic regression might not capture complex relationships in the data as well as more sophisticated models.
3. **CNN-GRU-LSTM model:** Exhibited high accuracy (0.998) and precision (0.998), with impressive recall (0.998) for the "malicious" class. However, it showed a high false negative rate (0.735), indicating that it misses a significant number of positive instances. Despite these limitations, the model achieved a strong AUC of 0.8559, reflecting its effectiveness in distinguishing between benign and malicious traffic. The computational complexity of this model is evident in its longer test time of 645.40 seconds.
4. **Error Metrics:** The models generally exhibited low mean squared errors and high F1-scores, reflecting their ability to balance precision and recall effectively. Notably, the CNN-GRU-LSTM model showed a well-rounded performance with high accuracy, precision, recall, and F1-score, but also revealed areas for improvement, such as the False Negative Rate (FNR) and False Discovery Rate (FDR).

In summary, while the LDA and Logistic Regression models demonstrate high accuracy, they struggle with class imbalance, particularly concerning malicious instances. In contrast, the CNN-GRU-LSTM model balances performance with complexity, making it preferable for scenarios demanding high accuracy, despite its longer processing times.

5. 8. Conclusions and Future Work

In conclusion, this research contributes valuable insights into the application of machine learning models for network attack detection and sets the stage for future advancements in the field. The recommendations and potential improvements outlined provide a roadmap for enhancing the effectiveness, efficiency, and interpretability of network security solutions.

This study presents a robust framework for network attack detection by integrating advanced machine learning techniques and model interpretability tools. The hybrid CNN-GRU-LSTM model demonstrated strong performance with an accuracy of 0.998 and effectively capturing complex patterns in network traffic data. The utilization of the SYN DoS dataset from the Kitsune Network Attack Dataset provided a solid foundation for our analysis and model training, ensuring that our findings are grounded in realistic and relevant data. Furthermore, the application of SHAP values not only enhanced the interpretability of the model but also offered valuable insights into feature importance, thereby building trust in the model's predictions.

Looking ahead, future work can explore several avenues to further improve network attack detection systems. One potential direction is the incorporation of additional datasets to enhance the model's generalizability and robustness against diverse attack vectors. Moreover, experimenting with ensemble methods that combine the strengths of multiple algorithms could yield even better performance. Another important aspect is the exploration of real-time detection capabilities, enabling the model to operate in live network environments where prompt response to threats is crucial.

Additionally, ongoing research into explainable AI (XAI) techniques can enhance our understanding of model decisions, allowing practitioners to interpret complex interactions between features more effectively. Lastly, developing user-friendly visualization tools for SHAP outputs could assist security analysts in quickly identifying critical indicators of attacks, facilitating faster decision-

making processes. Through these future endeavors, we aim to advance the field of network security and contribute to creating safer digital environments.

References

- [1]. Barry, B., Chan, H. A. Barry, B., Chan, H. (2010), Intrusion Detection Systems, In: Stavroulakis, P., Stamp, M. (eds): Handbook of Information and Communication Security pp193-205, SpringerLink. DOI:10.1007/978-3-642-04117-4_10.
- [2]. Ashiku L. and Dagli C.H. (2021). Network Intrusion Detection System using Deep Learning, *Procedia Computer Science* 2021, 185(1):239-247
- [3]. Gottapu S. R. and Krishna S. P. (2023), A Novel Approach for Detection of DoS / DDoS Attack in Network Environment using Ensemble Machine Learning Model. *International Journal on Recent and Innovation Trends in Computing and Communication* 11(9):244-253. DOI: 10.17762/ijritcc.v11i9.8340 ISBN: 2321-8169
- [4]. Gottapu S. R. and Krishna S. P. (2024), Exploring a novel framework for DoS/DDoS attack detection and simulation in contemporary networks, *January 2024i-manager's Journal on Software Engineering* 18(3):43. DOI:10.26634/jse.18.3.20596
- [5]. Inuwa, M. M., & Das, R. (2024). A comparative analysis of various machine learning methods for anomaly detection in cyber-attacks on IoT networks. *Internet of Things*, 26, 101162. <https://doi.org/10.1016/j.iot.2024.101162>
- [6]. Becerra-Suarez, F.L., Tuesta-Monteza, V.A., Mejia-Cabrera, H.I., Arcila-Diaz, J. (2024). Performance Evaluation of Deep Learning Models for Classifying Cybersecurity Attacks in IoT Networks. *Informatics* 2024, 11, 32. <https://doi.org/10.3390/informatics11020032>
- [7]. Liu, H.; Lang, B. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Appl. Sci.* 2019, 9, 4396. <https://doi.org/10.3390/app9204396>
- [8]. Amutha S., Kavitha R., Srinivasan R. and Kavitha M., "Secure network intrusion detection system using NID-RNN based Deep Learning," 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 2022, pp. 1-5, doi: 10.1109/ACCAI53970.2022.9752526.
- [9]. Liao, H., Murah, M. Z., Hasan, M. K., Aman, A. H. M., Fang, J., Hu, X., & Khan, A. U. R. (2024). A Survey of Deep Learning Technologies for Intrusion Detection in Internet of Things. *IEEE Access*. vol.12, pp.4745-4761, 2024.
- [10]. Tossou, S., Qorib, M., & Kacem, T. (2023, October). Anomaly Based Intrusion Detection System:

- A Deep Learning Approach. In 2023 International Symposium on Networks, Computers and Communications (ISNCC) (pp. 1-6). IEEE, Doha, Qatar, 2023, pp. 1-6, doi: 10.1109/ISNCC58260.2023.10323740.
- [11]. Rani, S., & Kumar, S. (2023, May). Unleashing the Power of Machine and Deep Learning for Advanced Network Intrusion Detection: An Analysis and Exploration. In 2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI) IEEE, Chennai, India, 2023, pp. 1-9, doi: 10.1109/ACCAI58221.2023.10200892.
- [12]. Pandathara A. (2023). A Comprehensive Examination of Literature Exploring the Implementation of Machine Learning to Network Security's Intrusion Detection Systems. International Journal of Advanced Research in Science, Communication and Technology, doi: 10.48175/ijarsct-8605
- [13]. Hussain, A., Sharif, H., Rehman, F., Kirn, H., Sadiq, A., Khan, M. S., Riaz, A., Ali, C. N., & Chandio, A. H. (2023). A systematic review of intrusion detection systems in internet of things using ML and DL. In 2023 4th International Conference on Computing, Mathematics and Engineering Technologies (iCoMET) (pp. 1-5). IEEE. <https://doi.org/10.1109/iCoMET57998.2023.10099142>
- [14]. Nakip M., Gül B. C., Gelenbe E. (2023). Decentralized Online Federated G-Network Learning for Lightweight Intrusion Detection. In 2023 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE, 2023, pp. 1-8. DOI:10.1109/MASCOTS59514.2023.10387644
- [15]. Zhu, S., Xu, X., Zhao, J., & Xiao, F. (2024). Lkd-stnn: A lightweight malicious traffic detection method for internet of things based on knowledge distillation. IEEE Internet of Things Journal, vol. 11, no. 4, pp. 6438-6453, 15 Feb.15, 2024.
- [16]. Kheddar, H., Himeur, Y., & Awad, A. I. (2023). Deep transfer learning for intrusion detection in industrial control networks: A comprehensive review. Journal of Network and Computer Applications, 220, 103760. <https://doi.org/10.1016/j.jnca.2023.103760>
- [17]. Sunil, C. K., Reddy, S., Kanber, S. G., Sandeep, V. R., & Patil, N. (2023). Comparative analysis of intrusion detection system using ML and DL techniques. In Hybrid Intelligent Systems (pp. 736-745). Springer, Cham. https://doi.org/10.1007/978-3-031-27409-1_67
- [18]. Mert, Nakip., Baran, Can, Gül., Erol, Gelenbe. (2023). Decentralized Online Federated G-Network Learning for Lightweight Intrusion Detection. arXiv.org, doi: 10.48550/arXiv.2306.13029
- [19]. Wasnik P. and Chavhan N., "A Review Paper on Designing Intelligent Intrusion Detection System Using Deep Learning," 2023 11th International Conference on Emerging Trends in Engineering & Technology - Signal and Information Processing (ICETET - SIP), Nagpur, India, 2023, pp. 1-6, doi: 10.1109/ICETET-SIP58143.2023.10151563.
- [20]. Ogundokun R. O., Basil U., Babatunde A. N., Abdulahi A. T., Adenike A. R. and Adebisi A. A., "Intrusion Detection Systems Based on Machine Learning Approaches: A Systematic Review," 2023 International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG), Omu-Aran, Nigeria, 2023, pp. 01-04, doi: 10.1109/SEB-SDG57117.2023.10124506.
- [21]. Krishna, A., Lal, A., Mathewkutty, A. J., Jacob, D. S., & Hari, M. (2020, July). Intrusion detection and prevention system using deep learning. In 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC) (pp. 273-278). IEEE.
- [22]. Fadel, M. M., El-Ghamrawy, S. M., Ali-Eldin, A. M., Hassan, M. K., & El-Desoky, A. I. (2022). HDLIDP: A Hybrid Deep Learning Intrusion Detection and Prevention Framework. Computers, Materials & Continua, 73(2).
- [23]. Alghamdi, Mohammed I., A Hybrid Model for Intrusion Detection in IoT Applications, Wireless Communications and Mobile Computing, 2022, 4553502, 9 pages, 2022. <https://doi.org/10.1155/2022/4553502>
- [24]. Monani A. Bhusnale O. Borade K. Madali R. (2023). Analysing Cyber Threats: A Comprehensive Literature Review on Data-Driven Approaches, International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), Volume 9, Issue 3, pp.188-193, May-June-2023. Available at doi : <https://doi.org/10.32628/CSEIT2390351>
- [25]. Auwal, Sani, Iliyasu. (2022). A Survey of Network Intrusion Detection Techniques Using Deep Learning. International Journal of Engineering Research in Computer Science and Engineering, doi: 10.36647/ijercse/09.08.art017
- [26]. [28] Hnamte V., Hussain J. (2023). Network Intrusion Detection using Deep Convolution Neural Network. 4th International Conference for Emerging Technology (INCET). doi: 10.1109/INCET57972.2023.10170202
- [27]. Alabdulatif, A., & Rizvi, S.S.H. (2022). Machine learning approach for improvement in kitsune NID. Intelligent Automation & Soft Computing, 32(2), 827-840. <https://doi.org/10.32604/iasc.2022.021879>

- [28]. Malliga, S., Nandhini, P. S., & Kogilavani, S. V. (2022). A comprehensive review of deep learning techniques for the detection of (distributed) denial of service attacks. *Information Technology and Control*, doi: 10.5755/j01.itc.51.1.29595
- [29]. Sujatha, V., Prasanna, K. L., Niharika, K., Charishma, V., & Sai, K. B. (2023). Network intrusion detection using deep reinforcement learning. 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), 1146-1150. <https://doi.org/10.1109/ICCMC56507.2023.10083673>
- [30]. Mohammed, A., Bahashwan, A.A., Manickam, S., Al-Amiedy, T.A., Aladaileh, M.A., & Hasbullah, I.H. (2023). A systematic literature review on machine learning and deep learning approaches for detecting DDoS attacks in software-defined networking. *Sensors*, 23(9), 4441. <https://doi.org/10.3390/s23094441>
- [31]. Omarov, B., Asqar, M., Sadybekov, R., Koishiyeva, T., Bazarbayeva, A., & Uxikbayev, Y. (2022). IoT network intrusion detection: A brief review. 2022 International Conference on Smart Information Systems and Technologies (SIST), 1-5. <https://doi.org/10.1109/SIST54437.2022.9945763>
- [32]. Tahreem, M., Andleeb, I., Hussain, B. Z., & Hameed, A. (2022, December). Machine learning-based Android intrusion detection systems. Paper presented at the International Conference on Data Intensive Applications & Their Challenges (Computatia X), Jaipur, India. Aligarh Muslim University, University of Windsor, Texas A&M University.
- [33]. Gonaygunta, H. (2023). Machine learning algorithms for detection of cyber threats using logistic regression. *International Journal of Smart Sensor and Adhoc Network*, 3(4), 36-42. <https://doi.org/10.47893/IJSSAN.2023.1229>
- [34]. Pandey, G., Kumar, A. K., & Jha, M. (2024). Human activity recognition using CNN-LSTM-GRU model. *International Research Journal on Advanced Engineering Hub (IRJAEH)*, 2(04), 889-894. <https://doi.org/10.47392/IRJAEH.2024.012>
- [35]. Bhattarai, A., Gyawali, U., Verma, A., & Ranga, V. (2024). Improving intrusion detection in a software-defined network using hybrid CNN and Bi-LSTM. *Proceedings of the 2024 IEEE International Conference on Artificial Intelligence and Computational Applications (ICAAIC)*. <https://doi.org/10.1109/icaaic60222.2024.10575090>
- [36]. Abdulhakim, A., & Ilyas, M. (2024). Deep learning for smart grid intrusion detection: A hybrid CNN-LSTM-based model. *International Journal of Artificial Intelligence & Applications (IJAAI)*, 15(3), 1-10. <https://doi.org/10.5121/ijaia.2024.15301>
- [37]. Al-Aql, N. (2024). Hybrid RNN-LSTM networks for enhanced intrusion detection in vehicle CAN systems. *Journal of Electrical Systems*, 33(1), 1-8. <https://doi.org/10.52783/jes.3318>
- [38]. Poornachander, V., Kumar, K. S., & Jagadish, S. (2024). DDoS attack intrusion detection system with CNN and LSTM hybridization. *Proceedings of the 2nd International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, Coimbatore, India, 1-6. <https://doi.org/10.1109/ICSCSS60660.2024.10625330>
- [39]. Lv, H., & Ding, Y. (2024). A hybrid intrusion detection system with K-means and CNN+LSTM. *EAI Endorsed Transactions on Scalable Information Systems*, 11(6). <https://doi.org/10.4108/eetsis.5667>
- [40]. Abu Khalil, D., & Abuzir, Y. (n.d.). Detecting and Analyzing Network Attacks: A Time-Series Analysis Using the Kitsune Dataset. *Journal of Emerging Computer Technologies*, 5(1), 9-23. <https://doi.org/10.57020/ject.1563146>
- [41]. Gür, Y. E. (2024). Comparative Analysis of Deep Learning Models for Silver Price Prediction: CNN, LSTM, GRU and Hybrid Approach. *Akdeniz İİBF Dergisi*, 24(1), 1-13. <https://doi.org/10.25294/aiiibfd.1404173>
- [42]. Scikit-learn. (2021). *scikit-learn: Machine Learning in Python*. Retrieved from <https://scikit-learn.org/>
- [43]. Hayel, R., Hindi, K. M., Hosny, M. I., & Alharbi, R. (2024). A selective LVQ algorithm for improving instance reduction techniques and its application for text classification. *Journal of Intelligent & Fuzzy Systems*. <https://doi.org/10.3233/JIFS-235290>
- [44]. Davis, J., & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*.
- [45]. Van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworth-Heinemann.
- [46]. Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 861-874.
- [47]. Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017)*, 4765-4774.
- [48]. Lundberg, S. M., Erion, G., & Lee, S. I. (2020). Explainable AI for Trees: From Local Explanations to Global Understanding. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAccT 2020)*, 418-429.