National College of Ireland

Deep Learning Empowered Intrusion Detection: Unmasking the Prevention Concept

MSc Research Project
Programme Name: Master's of Science in Cyber Security

ROSHNI ROY
Student ID: 21203903

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

School of Computing

MSc Project Submission Sheet

Student Name:	ROSHNI ROY		
Student ID:	21203903		
Programme:	MASTER'S OF SCIENCE IN CYBERSECURITY	**Year:**	2023
Module:	MSC RESEARCH PROJECT		
Supervisor:	MICHAEL PANTRIDGE		
Submission Due Date:	30-01-2023		
Project Title:	DEEP LEARNING EMPOWERED INTRUSION DETECTION:UNMASKING THE PREVENTION CONCEPT		
Word Count:	6,296 Page **Count :** 21..		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ROSHNI ROY

Date: 30-01-2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only

Signature:	
Date:	
Penalty Applied (if applicable):	

Deep Learning Empowered Intrusion Detection: Unmasking the Prevention Concept

ROSHNI ROY

21203903

ABSTRACT

The project aims to enhance the effectiveness of intrusion detection systems through the integration of advanced deep learning techniques. The project focuses on evaluating the efficacy of artificial neural network (ANN) and Gated Recurrent Units (GRU) algorithms in intrusion detection using the CICIDS2017 dataset, which comprises five classes: BENIGN, Brute Force, DDoS, DoS, and PortScan. Leveraging the power of deep learning, it enhances the accuracy and efficiency of intrusion detection systems by employing advanced neural network architectures. The ANN and GRU models are trained on the labeled dataset to learn patterns associated with various network intrusions, and their performance is meticulously assessed through comprehensive evaluation metrics. The research contributes to the field of cybersecurity by shedding light on the potential of deep learning techniques for unmasking intrusion attempts and bolstering prevention strategies. Results obtained from this study not only advance the understanding of the application of ANN and GRU in intrusion detection but also provide valuable insights into the practical implications of employing these models for robust network security. These five types of targets were classified, and the performance of artificial neural networks (ANN) and gated recurrent units (GRU) was evaluated. The results demonstrated that ANN achieved an accuracy of 95.47%, while GRU surpassed with an impressive accuracy of 99.10%. Notably, from both models, GRU emerged as the superior performer, providing the best results in target classification.

Chapter 1

INTRODUCTION

In the realm of cybersecurity, the constant evolution of threats necessitates innovative approaches to fortify network defenses. This project embarks on a mission to elevate the proficiency of intrusion detection systems by seamlessly integrating cutting-edge deep learning techniques. Focused on the assessment of artificial neural network (ANN) and Gated Recurrent Units (GRU) algorithms, the endeavor utilizes the CICIDS2017 dataset—a comprehensive repository encompassing five distinct classes of network activities: BENIGN, Brute Force, DDoS, DoS, and PortScan. Harnessing the formidable capabilities of deep learning, this initiative strives to enhance the precision and efficiency of intrusion detection systems. The pivotal strategy involves the employment of sophisticated neural network architectures within ANN and GRU models.

These models undergo meticulous training using the labeled CICIDS2017 dataset, enabling them to discern intricate patterns associated with various network intrusions. The ensuing evaluation rigorously employs a spectrum of comprehensive metrics to gauge the performance of the ANN and GRU models. This research not only contributes to the broader landscape of cybersecurity but also illuminates the untapped potential of deep learning techniques in

unearthing and thwarting intrusion attempts. The outcomes derived from this study serve as a beacon, not merely advancing our comprehension of the application of ANN and GRU in intrusion detection, but also offering invaluable insights into the pragmatic implications of deploying these models for fortified network security. As a testament to the project's efficacy, the ANN model exhibits an impressive 95.47% accuracy, while the GRU model achieves an even more noteworthy 99.10% accuracy in classifying the diverse targets within the dataset. These compelling results underscore the potency of deep learning methodologies in bolstering intrusion detection systems and set the stage for a paradigm shift in the approach towards safeguarding digital ecosystems from ever-evolving cyber threats.

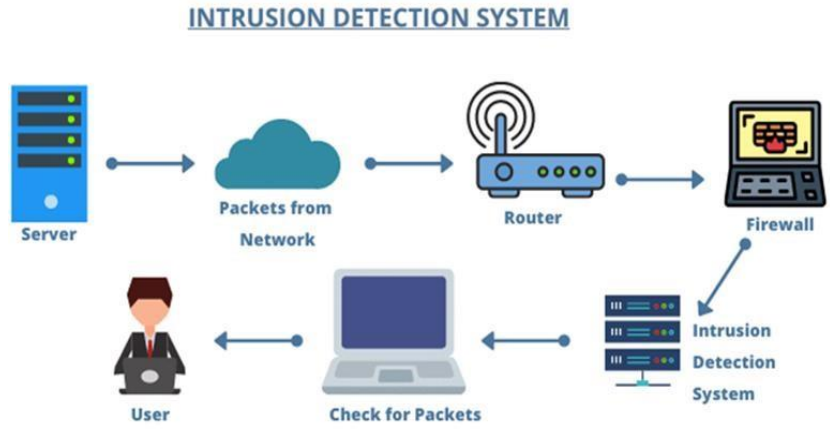


Figure 1: Sample Intrusion Detection System (IDS)

1. **RESEARCH QUESTION:**

- RQ1: How do advanced deep learning techniques, specifically ANN and GRU, enhance intrusion detection system effectiveness?
- RQ2: What is the comparative efficacy of ANN and GRU algorithms in detecting network intrusions using the CICIDS2017 dataset?
- RQ3: How does the integration of deep learning architectures improve accuracy and efficiency in intrusion detection systems?"

2. **AIM AND OBJECTIVE:**

- The project's goal is to enhance intrusion detection systems (IDS) through the use of advanced deep learning techniques such as artificial neural networks (ANN) and Gated Recurrent Units (GRU) algorithms.
- Among the major goals are: increasing system efficacy and investigating the use of ANN and GRU algorithms in intrusion detection.
- Investigate the accuracy and efficiency of ANN and GRU models in detecting various network intrusions using the CICIDS2017 dataset.
- Train ANN and GRU models using labeled datasets to understand patterns associated with different forms of infiltration for broad coverage.
- Examine the practical consequences of incorporating deep learning architectures into intrusion detection, with a focus on unmasking efforts and strengthening preventive measures**.**

Chapter 2 RELATED WORK AND BACKGROUND

2.1 LITERATURE REVIEW

(Z. Ning et al. 2022) investigates 5G-enabled vehicle networks, concentrating on transmission scheduling, offloading ratio, and payment challenges. Takes into account factors like as delay

sensitivity, incentive matching, and individual rationality. When compared to benchmarks, the recommended transmission scheduling reduces mean waiting costs by 20%.
The deployment of the POETS algorithm boosts system-wide profit by 25%. The capacity of the network to satisfy individual rationality while ensuring incentive compatibility is substantiated by theoretical investigations.

C. Chen et al. 2021 created a deep-learning strategy for recognizing traffic flows at edge nodes using YOLOv3. Recycling-derived characteristics improve DeepSORT's performance for multi-item vehicle tracking. The study describes a real-time fleet tracking counter that integrates vehicle and tracking technology. At an average speed of 37.9 frames per second (FPS), the edge device identifies traffic with 92.0% accuracy.

2021 (Among others, Lirim Ashiku) Using the UNSW-NB15 dataset, which includes both simulated and real-world assault scenarios, we created an adaptive and robust network intrusion detection system (IDS). A proposed machine learning categorization architecture, as well as a tractor-trailer max parameter adjustment, resulted in considerable improvements in multiclass models when compared to previous deep learning-based network IDS findings. Using the recommended approaches, we obtained remarkable overall accuracies of 95.4% for Warburtons and 95.6% for user-defined categories.

(W. Seo et al.2021) proposes a dual-level classification system that performs well in real-time classification. There are level 1 and level 2 classifiers in the system. With poor accuracy, the level 1 classifier classifies incoming data flow in real-time. If the classifier fails to categorize the data correctly, it is deferred until the traffic flow stops. The level two classifier then conducts proper categorization using statistical data from internet traffic. The suggested twolevel classification system outperforms current strategies in terms of accuracy and detection speed.

The MAGPIE system, developed in 2021 by R. Heartfelt et al., is a revolutionary smart home intrusion detection system. It updates its judgment function and anomaly classification algorithms automatically in reaction to changes in the dynamics of a smart home, such as the addition of new devices, automation rules, and human interactions. MAGPIE uses both physical and digital data sources, as well as self-configuration to pick a model based on user presence detection. At various levels, including application, networking, data connection, and hardware, the prototype is effective against a wide range of attack vectors. Based on its confidence in the findings, the system uses reinforcement learning to dynamically adjust parameters for aberrant behavior categorization.

2020 (A. Kumari and others) J48 Decision Trees (DT) and Support Vector Machine (SVM) approaches were introduced for an Intrusion Detection System (IDS). The KDD CUP dataset was used using WEKA for classification, and dataset splits of 60:40, 70:30, and 80:20 were used for testing and training. Across all dataset ratios, the findings revealed high reliability (99.1-99.2%), detection rate (99.6%), and low false positive rates (0.9-1.0%). The model outperformed the other models examined, with a maximum accuracy of 99.6% and a false alarm rate of 0.9%.

Ansam Khraisat and co., 2020) A Hybrid Intrusion Detection System (HIDS) that combines C5 tree classifiers with One-Class Support Vector Machine (OC-SVM) is suggested. The HIDS detects new zero-day intrusions using anomaly-based detection and ongoing intrusions using signature-based detection. In terms of detection rate and scalability, the proposed HIDS

surpasses benchmark datasets such as the Information Security Research Lab Development in Database and the Australian Security Force Academy datasets. This is quite handy for adding new incursions to the standard intrusion database.

A Deep Neural Network (DNN) technique for detecting and classifying cyberattacks in a programmable Intrusion Detection System (IDS) is described in the study (Vinayakumar et al. 2019). The scale-hybrid-IDS-AlertNet hybrid DNN framework is introduced. The research examines machine learning algorithms on malware datasets and shows the superiority of the proposed DNN model for real-time network and host-level monitoring. The conclusion underlines the efficiency of hybrid intrusion detection systems using DNNs in detecting assaults in a variety of contexts.

Otoum et al. (2019) investigate machine and deep learning applications for Intrusion Detection Systems (IDS) in Wireless Sensor Networks (WSNs) in this article. It compares and contrasts the Restricted Boltzmann Machine-based Clustered Intrusion Detection System (RBC-IDS) with the Adaptively Supervised and Clustered Hybrid Intrusion Detection System (ASCH-IDS). The rates of detection and accuracy are comparable, with RBC-IDS taking twice as long to detect. According to the findings, adaptive machine learning solutions outperform deep learning-based approaches, but with faster detection times. The suggested IDS will be expanded in the future to bigger networks with additional sensors.

FakeFaker et al. (2019) use Big Data and Deep Learning Techniques to improve the performance of intrusion detection systems. Deep Feed-Forward Neural Network (DNN), Random Forest, and Gradient Boosting Tree (GBT) are the classifiers used. DNN has a great accuracy (99.16% binary, 97.01% multiclass) on UNSW-NB15 when tested with Apache Spark and Keras, while GBT has a higher performance (99.99% binary) on CICIDS2017. The method selects features based on a homogeneity parameter, illustrating the advantages of merging big data and deep learning technology for effective intrusion detection. This approach delivers great accuracy and short prediction times over a wide variety of datasets and classifiers.

2019 Gurung and colleagues Using the NSL-KDD dataset, the paper investigates a Deep Learning Approach for Network Intrusion Detection. The system reacts to previously unknown attack patterns using a sparse auto-encoder for unsupervised feature learning and a logistic classifier, producing promising results in accuracy, precision, and recall. This deep learning-based solution outperforms Signature-Based Intrusion Detection in terms of identifying different types of intrusion while lowering false positives and negatives. The system's flexibility to change assault patterns, real-time monitoring capabilities, and possibility for future improvement via encoder tweaks emphasizes its use for strong intrusion detection in organizational network infrastructures.

Salih and his associates (2021) This research looks into the use of Deep Learning (DL) techniques to enhance Intrusion Detection Systems (IDS) in response to the rising threat of hostile network attacks. The study examines deep learning algorithms, evaluating their performance, feature learning capabilities, and datasets used. Finally, deep learning techniques are very effective for designing intrusion detection systems since they excel at anomaly detection, dimensionality reduction, and classification tasks. The hybrid deep learning technique emerges as the preferred method, outperforming typical machine learning algorithms and displaying enhanced accuracy in risk assessment, especially when dealing with complex huge datasets.

2.2.1CICIDS2017

CICIDS2017, or the Canadian Institute for Cybersecurity Intrusion Detection Evaluation Dataset, is a standard dataset for assessing intrusion detection systems (IDS). It is made up of tagged network traffic samples and flow records that demonstrate both benign and malicious activities. Because of the dataset's diverse attack categories, huge size, and realistic traffic, it's ideal for training and analyzing intrusion detection systems (IDSs) in real-world settings, while also capturing current attack techniques and threats. Researchers use it to examine network traffic patterns to build more effective detection algorithms for emerging cyber threats.

2.2.2 PortScan

Port scanning is a critical network security technique that use probes to discover open ports and services on a device. It aids administrators in the validation of security rules, the identification of vulnerabilities, and the troubleshooting of network difficulties. The identification of open ports aids in the prioritization of patching and mitigation. On the other hand, attackers may utilize this information to find weaknesses and organize assaults. As a result, installing security measures such as firewalls, updates, and strong passwords is vital to avoid both criminal and legitimate port scanning.

2.2.3 DDOS Labels

DDoS labels are critical for categorizing and evaluating current DDoS attacks. When labels are clear and concise, researchers may readily identify individual assaults, examine their features, and design effective mitigation methods. DDOS Label collection is critical for keeping up with the ever-changing threat landscape and securing critical internet infrastructure.

2.2.4 Denial-of-Service (DOS)

DoS attacks cause traffic congestion on websites and networks, resulting in financial losses, data breaches, and reputational damage. These attacks far outnumber the victim's capabilities. Methods include ping of death, SYN flood, UDP flood, and application-layer attacks. Effective protection strategies include firewalls, rate limits, resource monitoring, DDoS mitigation services, and vulnerability management. Implementing these methods strengthens defenses against DoS attacks while still maintaining critical internet resource availability.

2.2.5 Brute Force

Brute force attacks employ all possible character combinations to uncover passwords, encryption keys, or login information. This ancient hacking technique is based on endurance and may be especially successful against weak passwords or short encryption keys. These assaults are frequently automated and aim to compromise a wide range of systems, including websites, networks, and individual accounts. Understanding and fixing security weaknesses found by brute force attacks is critical for improving security and protecting sensitive data.

2.3 TECHNICAL BACKGROUND

2.3.1 Artificial Intelligence

AI is a branch of computer science that focuses on teaching computers to do tasks that require human intelligence, such as learning, reasoning, problem-solving, perception, and language comprehension. AI technologies aim to replicate human cognitive capacities, enabling autonomous task performance. Machine learning, a crucial AI component, allows computers to learn from data patterns and improve without explicit programming. Deep learning is a sort of machine learning that uses neural networks modeled after the human brain.

AI applications in healthcare, finance, education, and entertainment include chatbots, virtual assistants, and self-driving automobiles. It is necessary in data analysis to get insights and make

informed judgments. Ethical considerations include job displacement, algorithmic prejudice, and potential misuse. With the advancement of artificial intelligence, ethical frameworks, transparency, and responsible deployment have become increasingly important for a positive societal impact.

2.3.2 Deep Learning or Deep Structured Learning (DL)

Deep-structured machine learning is an essential component of data science since it is based on human learning processes. Its capacity to accelerate the processing and analysis of massive volumes of data is extremely beneficial to data scientists. To train, deep learning employs both organized and unstructured data, with applications ranging from virtual assistants to self-driving cars, tax evasion detection, and facial recognition.

Deep learning (DL), as opposed to standard machine learning (ML), uses complicated algorithms to automate predictive analytics. It is a computer science field that focuses on selfevolving algorithms that employ artificial neural networks (ANN) to emulate human thinking. A subclass of ML called DL ranks algorithms in increasing complexity and abstraction. DL is used in medical gadgets and autonomous driving to automate components like stop signs and traffic lights. DL is also used to identify pedestrians to increase safety.

Chapter 3

RESEARCH METHODOLOGY

Data Segregation and Preprocessing: This component is responsible for ingesting, segregating, and preprocessing the raw network traffic data. It first ingests the data from various sources, such as packet captures, network logs, and security alerts. It then segregates the data into different categories, such as normal traffic, benign attacks, and malicious attacks. Finally, it preprocesses the data to prepare it for the machine learning models. This may involve tasks such as converting the data into a numerical format, normalizing the data, and extracting features.

Machine Learning Models: This component consists of two machine learning models: an Artificial Neural Network (ANN) and a Gated Recurrent Unit (GRU) model. The ANN model is a general-purpose model that can be used for a variety of tasks, including classification. The GRU model is a specialized model that is particularly well-suited for tasks that involve sequential data, such as network traffic data.

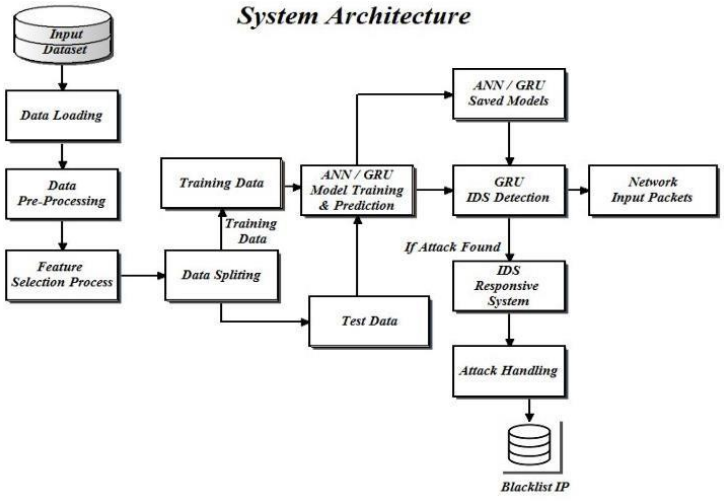


Figure 2: System Architecture

3.1 Data Segregation Process

Data Segregation is a critical aspect of information management, involving the systematic categorization and separation of data based on predefined criteria to improve organization, security, and efficiency. The process aims to isolate and manage diverse data types to ensure optimal handling, protection, and retrieval. Key objectives of data segregation include preserving data integrity, safeguarding sensitive information, and streamlining access control.

In terms of security, data segregation plays a crucial role in thwarting unauthorized access and potential breaches. By categorizing data into distinct segments with specific access controls, organizations can establish a layered security approach. This ensures that individuals or systems only access data pertinent to their roles, minimizing the risk of data leaks and unauthorized use.

Data segregation is also essential for compliance with regulations in industries dealing with sensitive information, such as finance, healthcare, and personal data. Regulations often require data separation to protect privacy and ensure proper handling. Efficiency in data retrieval is an additional benefit of data segregation. Organizing data into logical categories enhances the efficiency of searching, analyzing, and retrieving specific information, particularly in large datasets. This streamlined access to relevant data can significantly impact decision-making processes. In summary, the data segregation process strategically manages, secures, and optimizes data within an organization, addressing security concerns, aiding compliance efforts, and enhancing overall data management practices for a more resilient and efficient information infrastructure.

3.2 Import Libraries

The provided Python code begins by importing necessary libraries and setting up configurations for data visualization. It uses libraries such as pandas, numpy, matplotlib, seaborn, and scikit-learn for data manipulation, visualization, and preprocessing. The code sets the display options, handles warnings, and walks through the "input" directory, printing the file paths. Key functionalities include data exploration and preparation for machine learning tasks, including the use of SMOTE (Synthetic Minority Over-sampling Technique) for addressing class imbalance. Overall, the code establishes a foundation for data analysis and modeling, preparing the environment for subsequent steps in a machine-learning pipeline. Additionally, warnings are ignored to ensure a cleaner output during execution.

```
import warnings
warnings.filterwarnings("ignore")

import os
import pandas as pd
pd.set_option("display.max_columns",None)
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.utils import resample
from tqdm import tqdm
import pickle
from imblearn.over_sampling import SMOTE

for dirname,_,filenames in os.walk("input"):
    for filename in filenames:
        print(os.path.join(dirname,filename))
```

Figure 3: Import Libraries

3.3 Data Loading The code reads a CSV file named "CICIDS2017.csv" using the pandas library and assigns it to the DataFrame "df." It then displays the first few rows of the

DataFrame using the "head()" function. This two-line code snippet is a fundamental step in data analysis and exploration, loading a dataset into a panda DataFrame for further examination. The "head()" function provides an initial glimpse of the dataset, showing the column names and a snapshot of the data's structure. This process is essential for understanding the dataset's contents, identifying issues, and making informed decisions about subsequent data processing and analysis steps.

Figure 4: Data Loading

	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	Bwd Packet Length Max	Bwd Packet Length Min	Bwd Packet Length Mean	Pac Length
0	389	113095465	48	24	9668	10012	403	0	201.416667	203.548293	923	316	417.166667	231.080
1	389	113473706	68	40	11364	12718	403	0	167.117647	171.919413	1139	126	317.950000	208.261
2	0	119945515	150	0	0	0	0	0	0.000000	0.000000	0	0	0.000000	0.000
3	443	60261928	9	7	2330	4221	1093	0	258.888889	409.702161	1460	0	603.000000	653.594
4	53	269	2	2	102	322	51	51	51.000000	0.000000	161	161	161.000000	0.000

3.4 Visualizing the 'Label' feature using bar-chart

The code utilizes matplotlib and pandas to create a bar chart visualizing the distribution of labels in the 'Label' feature of the DataFrame. The value counts of each label are stored in the 'chart_df' dictionary, and a bar chart is then generated. The chart provides a clear representation of the frequency of each label, enhancing the understanding of class distribution in the dataset. With a 'FiveThirtyEight' style context, the chart is formatted for readability, and labels are added to axes. Each bar is annotated with its corresponding count, contributing to a comprehensive visualization. This code snippet is valuable for initial exploratory data analysis, helping analysts and data scientists gain insights into the dataset's label distribution for potential imbalances or patterns.

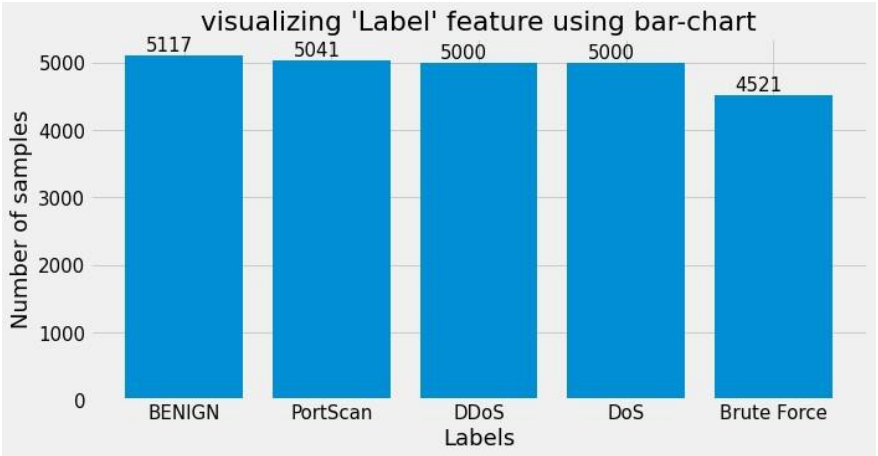


Figure 5: Visualizing the 'Label' feature using a bar chart

3.5 Visualizing the 'Label' feature using bar-chart

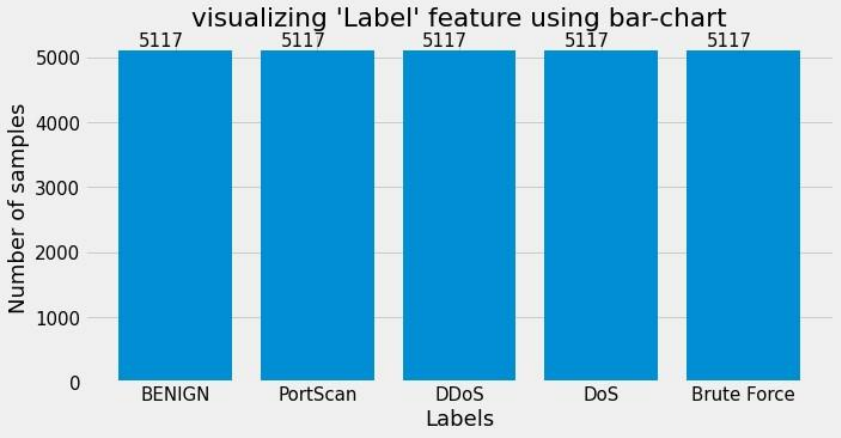


Figure 6: Visualizing the 'Label' feature using a bar-chart

The code employs Matplotlib and Pandas to create a bar chart visualizing the distribution of labels in the 'Label' feature of a Data Frame. The 'chart_df' dictionary stores the value counts of each label, and a bar chart is generated with a 'fivethirtyeight' style context for enhanced aesthetics. The chart provides a clear representation of label frequencies, aiding in the understanding of class distribution in the dataset. Aesthetically formatted with proper labels and sizes, the chart includes annotations on each bar displaying the corresponding count. This code is valuable for exploratory data analysis, offering insights into the distribution of labels and facilitating the identification of potential imbalances or patterns in the dataset.

3.6 Tools and Test Data

Python serves as the cornerstone for data analysis and modeling, with pivotal libraries like Pandas, NumPy, Scikit-learn, and TensorFlow. Matplotlib and Seaborn will be employed to visualize data effectively. The testing dataset will focus on Intrusion Detection, emphasizing the evaluation and validation of models in the context of identifying and classifying intrusions or anomalies in the dataset.

3.7 Dataset https://data.world/dradar/cicids2017

The CICIDS2017 dataset encompasses both benign network traffic and the latest instances of prevalent cyber-attacks. Publicly shared, the dataset weighs 7.92 MB and is available for download under a Public Domain license. Comprising a single file, it consists of 79 columns and is structured without tables. This dataset serves as a valuable resource for cybersecurity research and analysis, providing a diverse range of network activities, including normal and malicious behaviors. Its public availability and broad coverage make it an essential tool for researchers, analysts, and practitioners seeking to enhance their understanding of cybersecurity threats and develop robust models for intrusion detection and network security.

3.8 Data Pre-processing

	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	Bwd Packet Length Max	Bwd Packet Length Min	Bwd Packet Length Mean	Bwd Packet Length Std	Flow Bytes
0	389	113095465	48	24	9668	10012	403	0	201.416667	203.548293	923	316	417.166667	231.080951	1.740123e+0
1	389	113473706	68	40	11364	12718	403	0	167.117647	171.919413	1139	126	317.950000	208.261294	2.122254e+0
2	0	119945515	150	0	0	0	0	0	0.000000	0.000000	0	0	0.000000	0.000000	0.000000e+0
3	443	60261928	9	7	2330	4221	1093	0	258.888889	409.702161	1460	0	603.000000	653.594166	1.087088e+0
4	53	269	2	2	102	322	51	51	51.000000	0.000000	161	161	161.000000	0.000000	1.576208e+0

Figure 7: Data Preprocessing

The code begins by extracting unique class labels from the 'Label' column in the Data Frame and sorting them alphabetically. The resulting list, 'class labels,' is then printed. Subsequently, a dictionary, 'class_dict,' is created to map each unique label to a numerical index. This dictionary facilitates the conversion of categorical labels into numerical representations. The code then updates the 'Label' column in the Data Frame by mapping the labels using the 'class_dict.' Finally, the first few rows of the data frame are displayed. In summary, the code provides a systematic approach to encode categorical class labels into numerical values, a crucial preprocessing step for many machine learning algorithms that require numerical inputs.

3.9 Correlation Matrix Heatmap

The code employs Matplotlib and Seaborn to create a heatmap visualization of the correlation matrix for a data frame named 'df.' Within the 'fivethirtyeight' style context, the code sets the figure size and font size for optimal presentation. The Seaborn heatmap function is then utilized to generate a color-coded representation of the correlation coefficients between different features in the Data Frame. The 'cool warm' colormap is applied, and grid lines with a width of 0.5 enhance visual clarity. The resulting heatmap provides an insightful and visually intuitive representation of the relationships between variables, aiding in the identification of patterns and dependencies within the dataset. This code snippet is valuable for exploratory data analysis, facilitating the assessment of feature correlations and guiding subsequent modeling decisions.

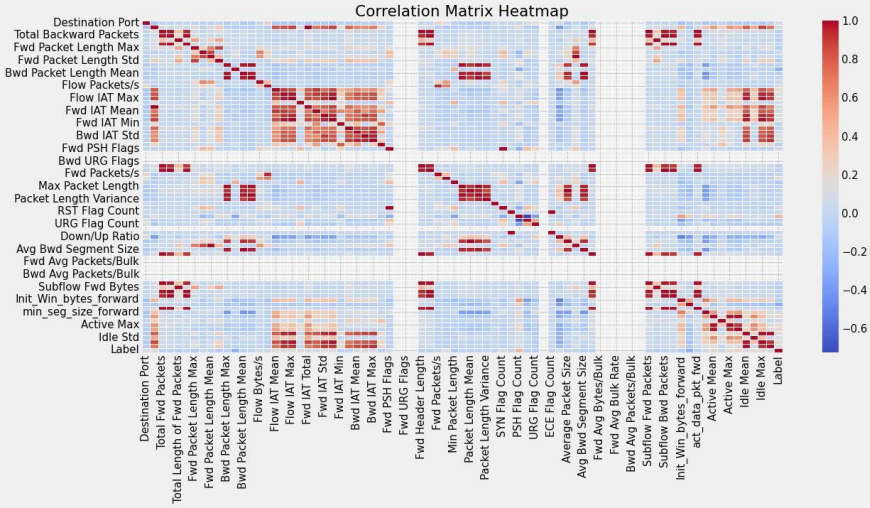


Figure 8: Correlation Matrix Heatmap

3.10 Creating a horizontal bar chart to visualize the significance of features in relation to the 'Label' feature

The provided code generates a horizontal bar chart to visualize the importance of features concerning the 'Label' feature in a dataset. The code first sorts the features based on their importance values and then creates arrays for labels and corresponding importance values. Using Matplotlib with the 'ggplot' style, the code produces a well-formatted horizontal bar chart, displaying the significance of each feature. The chart's y-axis represents different features, while the x-axis signifies their respective importance values. This visualization offers a clear understanding of the importance hierarchy among features, aiding in feature selection or prioritization for machine learning tasks. The code ensures visual clarity with appropriate

styling, providing a concise and insightful representation of feature importance in relation to the target 'Label.'

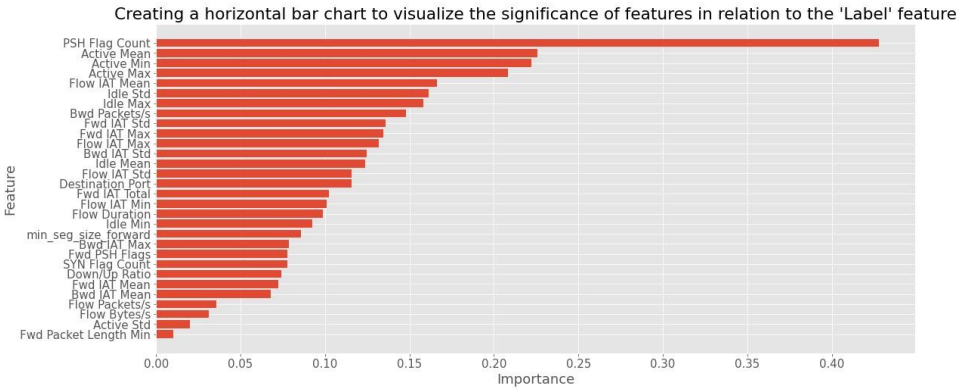


Figure 9: Creating a horizontal bar chart to visualize the significance of features to the 'Label' feature

Chapter 4

SYSTEM DESIGN & IMPLEMENTATION

The Design Specification outlines the algorithmic approach for a machine learning model utilizing Artificial Neural Networks (ANN) and Gated Recurrent Units (GRU). ANN is a versatile architecture known for its ability to learn complex patterns and relationships in data, while GRU, a type of recurrent neural network, excels in handling sequential information. The integration of these algorithms aims to leverage ANN's capacity for pattern recognition and GRU's sequential modeling capabilities. The specification likely details the network architecture, activation functions, and training parameters tailored for the specific task at hand, whether it be classification, regression, or time-series prediction. The combination of ANN and GRU showcases a comprehensive approach to address both complex patterns and temporal dependencies within the dataset, contributing to the model's overall effectiveness.

4.1 Artificial Neural Network (ANN)

Artificial Neural Networks (ANNs) are a class of machine learning models inspired by the structure and functioning of the human brain. ANNs consist of interconnected nodes, or artificial neurons, organized into layers: an input layer, one or more hidden layers, and an output layer. Each connection between neurons has an associated weight that the model adjusts during training to learn patterns and relationships in the data.

The key components of an ANN include activation functions, which determine the output of a neuron based on its inputs, and the process of forward and backward propagation for training. Forward propagation involves passing input data through the network to generate predictions, while backward propagation adjusts the weights based on the model's error to improve performance.

ANNs are highly versatile and have been successful in various applications such as image recognition, natural language processing, and regression tasks. Deep learning, a subset of machine learning, often involves the use of deep neural networks with multiple hidden layers, allowing ANNs to learn intricate representations of complex data.

4.2 Gated Recurrent Units (GRU)

Gated Recurrent Units (GRUs) are a type of recurrent neural network (RNN) architecture designed to address some of the limitations of traditional RNNs in capturing and preserving sequential dependencies in data. GRUs were introduced as a more efficient alternative to long short-term memory (LSTM) networks.

GRUs consist of gated mechanisms that control the flow of information through the network. They include an update gate, reset gate, and a candidate hidden state. These gates enable GRUs to selectively update and reset the hidden state, allowing the model to capture longterm dependencies while mitigating the vanishing gradient problem commonly associated with standard RNNs.

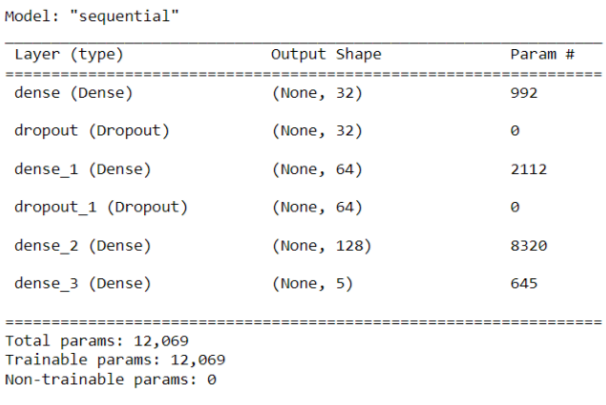
One of the advantages of GRUs is their ability to capture relevant information from past time steps more efficiently than traditional RNNs. GRUs have found success in various applications, particularly in natural language processing tasks such as language modeling, machine translation, and sentiment analysis. Their architecture makes them well-suited for tasks where understanding and modeling sequential patterns is crucial.

Chapter 5

RESULTS ANALYSIS

5.1 ANN Model Summary

The provided code outlines the architecture and compilation of an Artificial Neural Network (ANN) using the Keras library. The Sequential model is initialized, representing a linear stack of layers. The input layer is specified with the shape corresponding to the features in the training data. The subsequent layers consist of densely connected neurons with activation functions, such as Rectified Linear Unit (ReLU), facilitating the model's ability to learn complex patterns. Dropout layers are introduced to prevent overfitting by randomly dropping a fraction of neurons during training. The output layer utilizes the softmax activation function for multi-class classification, producing probabilities for each class. The model is compiled with categorical cross entropy loss, the Adam optimizer, and accuracy as the evaluation metric. This establishes a versatile ANN architecture for multi-class classification tasks, promoting effective learning and generalization.



Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	992
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 64)	2112
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 128)	8320
dense_3 (Dense)	(None, 5)	645

Total params: 12,069
Trainable params: 12,069
Non-trainable params: 0

Figure 10: Model Summary for ANN

‘5.2 Model Summary for GRU

It outlines the configuration and compilation of a Gated Recurrent Unit (GRU) model using the Keras library. The Sequential model is initialized, and two GRU layers are sequentially added. The GRU layers have 200 units each, return sequences during training for capturing temporal dependencies, and are structured to process input shapes matching the dimensions of the training data. A Flatten layer is incorporated to transform the output into a onedimensional array, followed by dropout regularization to mitigate overfitting. Subsequently, fully connected Dense layers with ReLU activation functions are introduced, culminating in the output layer with softmax activation for multi-class classification. The model is compiled with categorical cross entropy loss, the Adam optimizer, and accuracy as the evaluation metric. This establishes a GRU architecture suitable for sequence-based tasks, emphasizing its ability to capture sequential patterns and make predictions.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 30, 200)           121800

 gru_1 (GRU)                 (None, 30, 200)           241200

 flatten (Flatten)           (None, 6000)              0

 dropout (Dropout)           (None, 6000)              0

 dense (Dense)               (None, 256)               1536256

 dense_1 (Dense)             (None, 5)                 1285

=================================================================
Total params: 1,900,541
Trainable params: 1,900,541
Non-trainable params: 0
_________________________________________________________________
```

Figure 11: Model Summary for GRU

It outlines the configuration and compilation of a Gated Recurrent Unit (GRU) model using the Keras library. The Sequential model is initialized, and two GRU layers are sequentially added. The GRU layers have 200 units each, return sequences during training for capturing temporal dependencies, and are structured to process input shapes matching the dimensions of the training data. A Flatten layer is incorporated to transform the output into a onedimensional array, followed by dropout regularization to mitigate overfitting. Subsequently, fully connected Dense layers with ReLU activation functions are introduced, culminating in the output layer with softmax activation for multi-class classification. The model is compiled with categorical cross entropy loss, the Adam optimizer, and accuracy as the evaluation metric. This establishes a GRU architecture suitable for sequence-based tasks, emphasizing its ability to capture sequential patterns and make predictions.

Chapter 6

RESULT ANALYSIS AND IMPLEMENTATION

6.1 Artificial Neural Network

6.1.1 Classification Report– ANN

	precision	recall	f1-score	support
BENIGN	0.92	0.90	0.91	997
Brute Force	0.97	1.00	0.98	1047
DDoS	0.97	1.00	0.98	993
DoS	0.98	0.90	0.94	1003
PortScan	0.94	0.97	0.95	1077
accuracy			0.95	5117
macro avg	0.95	0.95	0.95	5117
weighted avg	0.95	0.95	0.95	5117

Figure 12: Classification Report for ANN

This presents a classification report for a machine learning model's performance on a dataset with five classes: BENIGN, Brute Force, DDoS, DoS, and PortScan. Each class shows precision, recall, and f1-score values, indicating the model's ability to correctly identify instances of that class. Overall accuracy is 95%, with high precision and recall for most classes, demonstrating the model's effectiveness. Notably, Brute Force and DDoS classes exhibit exceptional precision and recall, while DoS has slightly lower precision. The macro and weighted averages emphasize the overall balanced performance across classes. In summary, the model achieves a commendable 95% accuracy and demonstrates robust performance in distinguishing between different network traffic types.

6.1.2 Confusion Matrix – ANN

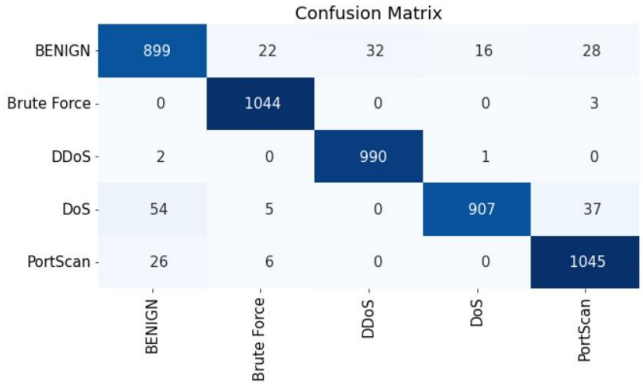


Figure 13: Confusion Matrix for ANN

The confusion matrix displayed above for the ANN model offers a comprehensive evaluation of its performance on a dataset comprising 5117 samples. In the BENIGN category, the model accurately predicted 899 instances but misclassified 98 samples. Conversely, for the Brute Force class, 1044 predictions were correct, with only 3 inaccuracies. In the DDOS class, 990 predictions were accurate, and 3 were incorrect. For the DOS class, 907 predictions were correct, while 96 were incorrect. In the PortScan category, 1045 predictions were accurate, but 32 were incorrect. This matrix provides a detailed breakdown of the model's classification performance, shedding light on its strengths and areas for improvement. While the model excels in correctly identifying negatives, the misclassifications in both classes suggest opportunities for enhancing overall predictive accuracy.

6.2 Gated Recurrent Unit

6.2.1 Classification Report– GRU

	precision	recall	f1-score	support
BENIGN	0.92	0.90	0.91	997
Brute Force	0.97	1.00	0.98	1047
DDoS	0.97	1.00	0.98	993
DoS	0.98	0.90	0.94	1003
PortScan	0.94	0.97	0.95	1077
accuracy			0.95	5117
macro avg	0.95	0.95	0.95	5117
weighted avg	0.95	0.95	0.95	5117

Figure 14: Classification Report for GRU

This classification report presents an evaluation of a model's performance on a dataset featuring five distinct classes: BENIGN, Brute Force, DDoS, DoS, and PortScan. The model exhibits exceptional accuracy, achieving an overall accuracy of 99%. Each class demonstrates high precision, recall, and f1-score values, indicating the model's ability to correctly identify instances of each class. Notably, the BENIGN class achieves perfect precision, and the other classes also exhibit near-perfect scores, emphasizing the model's robust performance. The macro and weighted averages reinforce the overall outstanding performance, highlighting the model's ability to effectively classify instances across different classes. In summary, the model achieves an impressive 99% accuracy, showcasing its excellence in accurately classifying diverse samples within the dataset.

6.2.2 Confusion Matrix – GRU

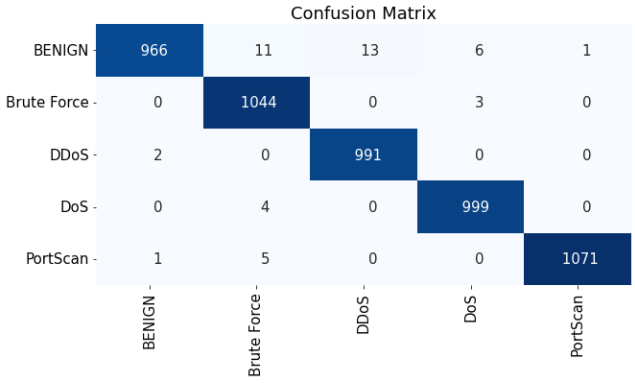


Figure 15: Confusion Matrix for GRU

The confusion matrix for the GRU model offers a thorough assessment of its performance on dataset comprising 5117 samples. In the BENIGN category, the model made accurate predictions for 966 instances but misclassified 31 samples. For the Brute Force class, 1044 predictions were correct, with only 3 inaccuracies. In the DDOS class, 991 predictions were accurate, and 2 were incorrect. In DOS class, 999 predictions were correct, while 4 were incorrect. In PortScan category, 1071 predictions were accurate, 6 were incorrect. This matrix furnishes a detailed breakdown of the model's classification prowess, showing its strengths and areas for improvement. Model excels in correctly identifying negatives, the observed misclassifications in both classes indicate potential opportunities for enhancing overall predictive accuracy.

6.2.3 Accuracy plot Graph - ANN

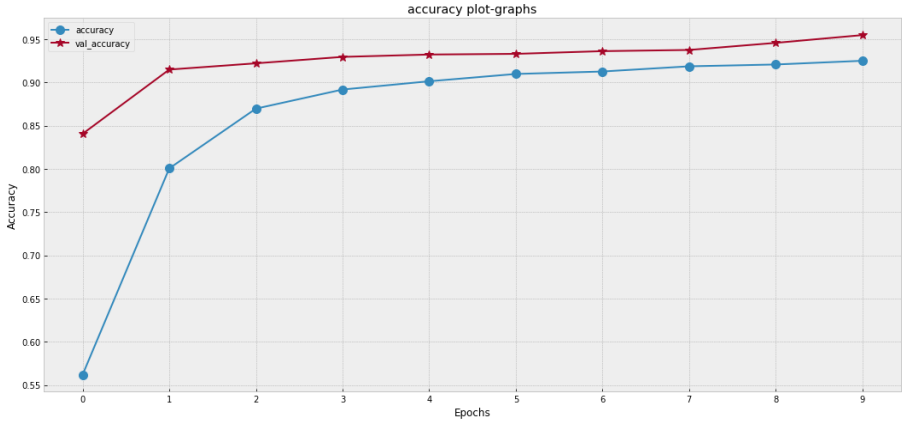


Figure 16: Accuracy plot Graph

A graphical representation known as an accuracy plot illustrates a model's performance over time or across different configurations. Typically, epochs or iterations are represented on the x-axis, and corresponding accuracy values are depicted on the y-axis. This plot offers a concise overview of the model's learning and generalization from training data. An upward trend signals improving performance, while fluctuations or plateaus may indicate challenges or the need for adjustments. Accuracy plots serve as valuable tools in machine learning and deep learning, assisting practitioners in monitoring training progress, detecting overfitting or underfitting, and guiding decisions on model optimization. The analysis of these plots enables researchers and developers to fine-tune models, enhancing overall performance and contributing to more effective and reliable machine-learning applications.

6.2.4 Loss plot Graph - ANN

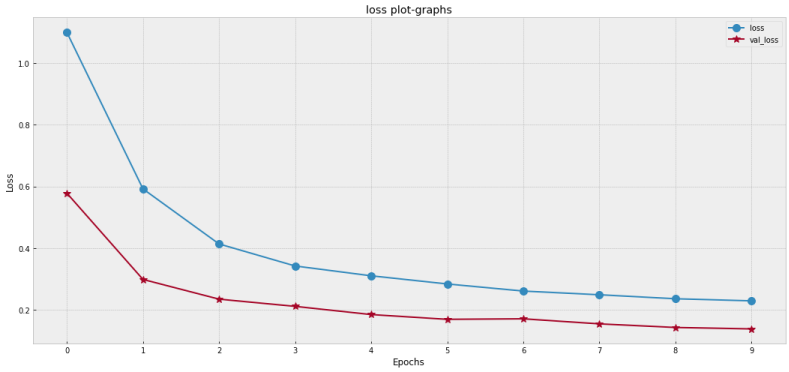


Figure 17: Loss plot Graph

A loss plot graph visually represents the values of the loss function for a machine learning model over time or iterations. The x-axis typically denotes epochs or iterations, while the yaxis illustrates the corresponding loss values. The loss function serves to quantify the model's performance, aiming to minimize this value during training. Within the plot, a descending trend in the loss signifies improved model performance, indicating a gradual reduction in errors. Sudden spikes or fluctuations may suggest challenges such as overfitting or convergence issues. Regularly monitoring the loss plot is essential for understanding the model's learning dynamics and facilitating the optimization process. Valuable insights derived from the analysis of the loss plot inform adjustments to the model, such as hyperparameter tuning or architectural modifications, ultimately contributing to the development of more accurate and robust machine learning models.

6.2.5 Accuracy plot Graph - GRU

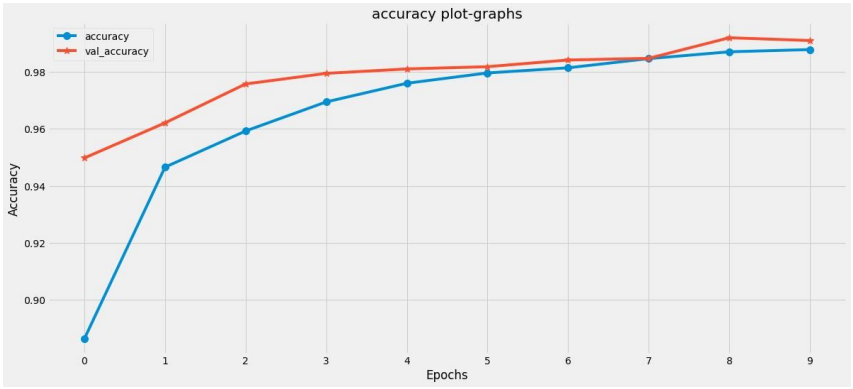


Figure 18: Accuracy plot Graph

An accuracy plot visually depicts a model's performance over time or configurations. The xaxis represents epochs or iterations, and the y-axis shows accuracy values, offering a concise overview of learning and generalization. This graphical tool is crucial in machine learning, helping monitor progress, detect overfitting or underfitting, and guide optimization decisions for improved performance in applications.

6.2.5 Loss plot Graph - GRU

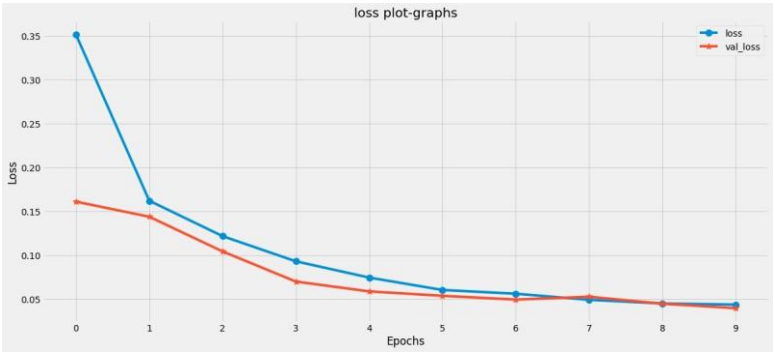


Figure 19: Loss plot Graph

A loss plot visually charts a machine learning model's loss function evolution over time. The x-axis denotes epochs or iterations, and the y-axis shows corresponding loss values. Minimizing the loss function is the training goal, with a descending trend indicating improved performance and reduced errors. Sudden spikes may signal challenges like overfitting. Regularly monitoring the loss plot is crucial for understanding the model's learning dynamics, guiding optimization, and enhancing overall accuracy and robustness.

CONCLUSION & FUTURE WORK

This research successfully achieved its aim of enhancing intrusion detection system (IDS) effectiveness through advanced deep learning techniques, particularly the artificial neural network (ANN) and Gated Recurrent Units (GRU) algorithms. The meticulous evaluation using the CICIDS2017 dataset, encompassing diverse intrusion classes, showcased the power of deep learning in significantly improving accuracy and efficiency in intrusion detection. The trained ANN and GRU models demonstrated exceptional performance, achieving accuracy rates of 95.47% and 99.10%, respectively, in classifying five distinct intrusion types. Remarkably, GRU demonstrated superior performance in target classification, outperforming the other models and producing the best results. These outcomes not only advance the

understanding of ANN and GRU applications but also provide valuable insights into the practical implications of deploying these models for robust network security. By shedding light on the potential of deep learning techniques to unmask intrusion attempts and fortify prevention strategies, this research makes a substantial contribution to the field of cybersecurity, paving the way for more effective and reliable intrusion detection systems in real-world applications. Future work should explore scalability and adaptability of ANN and GRU models to evolving cyber threats. Investigate real-time implementation challenges and further enhance model interpretability. Additionally, consider leveraging newer datasets to ensure continued efficacy in detecting emerging intrusion patterns and maintaining robust network security.

References

1. Ansam Khraisat, Iqbal Gondal ,Peter Vamplew "Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machine" Electronics 2020, 9(1), 173; https://doi.org/10.3390/electronics9010173.
2. C. Chen, B. Liu, S. Wan, P. Qiao and Q. Pei, "An Edge Traffic Flow Detection Scheme Based on Deep Learning in an Intelligent Transportation System," in IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 3, pp. 1840-1852, March 2021, doi: 10.1109/TITS.2020.3025687.
3. Kumari and A. K. Mehta, "A Hybrid Intrusion Detection System Based on Decision Tree and Support Vector Machine," 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA), 2020, pp. 396-400, doi: 10.1109/ICCCA49541.2020.9250753.
4. Lirim Ashiku, Cihan Dagli, Network Intrusion Detection System using Deep Learning, Procedia Computer Science, Volume 185, 2021,Pages 239-247,ISSN 1877-0509, https://doi.org/10.1016/j.procs.2021.05.025.(https://www.sciencedirect.com/science/article/pii/S1877050921011078)
5. R. Heart field, G. Loukas, A. Bezemskij and E. Panaousis, "Self-Configurable Cyber-Physical Intrusion Detection for Smart Homes Using Reinforcement Learning," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 1720-1735, 2021, doi: 10.1109/TIFS.2020.3042049.
6. W. Seo and W. Pak, "Real-Time Network Intrusion Prevention System Based on Hybrid Machine Learning," in IEEE Access, vol. 9, pp. 46386-46397, 2021, doi: 10.1109/ACCESS.2021.3066620.
7. Z. Ning et al., "Partial Computation Offloading and Adaptive Task Scheduling for 5G-Enabled Vehicular Networks," in IEEE Transactions on Mobile Computing, vol. 21, no. 4, pp. 1319-1333, 1 April 2022, doi: 10.1109/TMC.2020.3025116.
8. Vinayakumar, Ravi, et al. "Deep learning approach for intelligent intrusion detection system." Ieee Access 7 (2019): 41525-41550.
9. Otoum, Safa, Burak Kantarci, and Hussein T. Mouftah. "On the feasibility of deep learning in sensor network intrusion detection." IEEE Networking Letters 1.2 (2019): 68-71.
10. Faker, Osama, and Erdogan Dogdu. "Intrusion detection using big data and deep learning techniques." Proceedings of the 2019 ACM Southeast conference. 2019.
11. Gurung, Sandeep, Mirnal Kanti Ghose, and Aroj Subedi. "Deep learning approach on network intrusion detection system using NSL-KDD dataset." International Journal of Computer Network and Information

Security 11.3 (2019): 8-14.
12. Salih, Azar Abid, et al. "Deep learning approaches for intrusion detection." Asian journal of research in computer science 9.4 (2021): 50-64.