

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgavi-590018



A Project Report on

“BOOSTING NETWORK SECURITY: A COMPARATIVE ANALYSIS OF DEEP LEARNING TECHNIQUES FOR INTRUSION DETECTION”

Submitted in partial fulfillment of the requirement for award of degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

by

ADARSH PAWAR

1EP20CS003

MOHNISH REDDY G

1EP20CS055

MUKESHREDDY NAGIREDDYGARI

1EP20CS056

SHAIK ATHIQ REHAMAN

1EP20CS083

Under the guidance of

Dr. Heena Kousar

Assoc. Professor

Dept. of CSE, EPCET



Department of Computer Science and Engineering

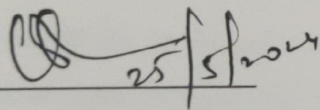
Approved by AICTE New Delhi | Affiliated to VTU, Belagavi,
Virgo Nagar, Bengaluru-560049



2023-2024

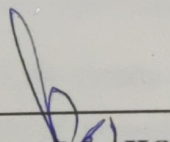
CERTIFICATE

This is to certify that the Project work entitled “**Boosting Network Security: A Comparative Analysis Of Deep Learning Techniques For Intrusion Detection**” is a bonafide work carried out by **ADARSH PAWAR [1EP20CS003]**, **MOHNISH REDDY G [1EP20CS055]**, **MUKESHREDDY NAGIREDDYGARI [1EP20CS056]**, and **SHAIK ATHIQ REHAMAN [1EP20CS083]**, in the partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgavi**, during the year **2023-2024**. It is certified that corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.



(Signature of the Guide)

Dr. Heena Kousar
Assoc. Prof., Dept. of CSE,
EPCET, Bengaluru

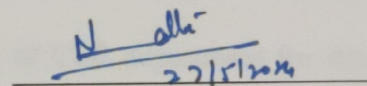


(Signature of the HOD)

Dr. I. Manimozhi
Professor, Dept. of CSE,
EPCET, Bengaluru

Professor & Head

Dept. of Computer Science Engineering
East Point College of Engineering & Technology
Bangalore - 560 049.



(Signature of the Principal)

Dr. Mrityunjaya V Latte
Professor & Principal
EPCET, Bengaluru

PRINCIPAL

**EAST POINT COLLEGE OF
ENGINEERING & TECHNOLOGY
BANGALORE- 560 049.**

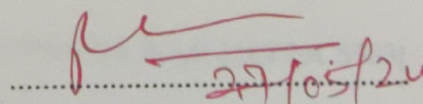
Signature with date

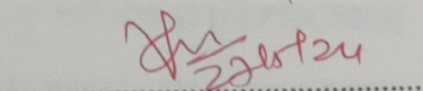
External Viva-Voce:

Name of the Examiners

1. Raja A

2. M. Jayaraman G





ACKNOWLEDGEMENT

First and foremost, we would like to express our sincere regards and thanks to **Management of East Point Group of Institutions, Bengaluru** for providing us an opportunity to work on the project Boosting Network Security: A Comparative Analysis of Deep Learning Techniques for Intrusion Detection.

We are grateful to **Dr. S Prakash, Senior Vice President, East Point Group of Institutions, Bengaluru** for his conscientious guidance and support that enabled us to complete this project.

We would like to express our gratitude to **Dr. Mrityunjaya V Latte, Principal, East Point College of Engineering and Technology, Bengaluru** for his constant support in completing the project work successfully.

We would like to express our sincere thanks to **Dr. I. Manimozhi**, Professor and Head of the Department of Computer Science and Engineering, EPCET for her valuable suggestions and encouragement to do our best in this project work.

We would like to express our gratitude towards our guide **Dr. Heena Kousar**, Associate Professor, Department of CSE and the project coordinator **Prof. Divya U H**, Assistant Professor, Department of CSE for their valuable guidance and constant supervision in completing the project successfully.

We would like to extend our thanks to all the faculty members of CSE department for their valuable inputs as reviewers during the course of the project work.

Finally, we would like to thank our parents and friends for their support and encouragement in successful completion of this project work.

ADARSH PAWAR [1EP20CS003]

MOHNISH REDDY G [1EP20CS055]

MUKESHREDDY NAGIREDDYGARI [1EP20CS056]

SHAIK ATHIQ REHAMAN [1EP20CS083]

ABSTRACT

This paper addresses demanding situations in Intrusion Detection Systems (IDS) by way of combining the Adaptive Synthetic Sampling (ADASYN) method with a break up-primarily based Resnet framework. ADASYN balances sample distribution, overcoming biases in the direction of large samples. Our method extracts multiscale functions, reduces interchannel redundancy, and enhances the version's capability using a smooth hobby operation. We recommend a Residual Neural Network (ResNN) version for intrusion detection, displaying massive enhancements in recognition accuracy and execution performance. Ongoing work targets to optimize performance further, highlighting the capacity for extra strong IDS solutions.

CONTENTS

Chapter No.	Description	Page No.
1	Introduction	1
	1.1 Background	1
	1.2 Problem Statement	2
	1.3 Existing System	2
	1.4 Proposed System	2
	1.5 Aim of the Project	3
	1.6 Objectives of the Project	3
	1.7 Summary	3
2	Literature Survey	4
3	Requirement Specification	9
	3.1 Hardware Requirements	9
	3.2 Software Requirements	9
	3.3 Tool Usage	10
	3.4 Functional Requirements	13
	3.5 Non-functional Requirements	14
	3.6 Summary	14
4	System Design	15
	4.1 System Architecture	15
	4.2 Data Flow Diagram	17
	4.3 UML Flow Diagrams	18
	4.3.1 Use Case diagram	18
	4.3.2 Sequence Diagram	19
	4. 4 Summary	21
5	System Implementation	22
	5.1 Modules and Components	22
	5.2 Residual Network	25
	5.3 Algorithms and Pseudo codes	29
	5.4 Summary	30
6	System Design	31

	6.1 Unit Testing	31
	6.2 Integration Testing	33
	6.3 System Testing	35
	6.4 Summary	36
7	Result and Analysis	37
	7.1 Snapshots	37
	7.2 Result Analysis	42
	7.3 Summary	43
8	Conclusion and Future Enhancement	44
	8.1 Conclusion	44
	8.2 Future Enhancement	44
	References	46
	Publication Certificate	50
	Plagiarism Certificate	52

LIST OF FIGURES

Figure No	Description	Page No
4.1	The overview of proposed methodology for Intrusion Detection	16
4.2	Level 0 Data Flow Diagram for classification of UNSW dataset	17
4.3	Level 1 Data Flow Diagram for extracting features from dataset	17
4.4	Use Case Diagram for Data Classification	18
4.5	Sequence Diagram for Data Processing	19
5.1	Undersampling and Oversampling for UNSW datasets	24
5.2	Illustration of Skip Connection in UNSW dataset	25
5.3	Skip Connection for Rectifying Error	26
5.4	Network Architecture of Residual Network	26
5.5	Flow Diagram of the Residual Network	27
7.1	Attacker before uploading Test Dataset to the Cloud	37
7.2	Attacker after uploaded Test Dataset to the Cloud	37
7.3	Test Data Uploaded to Cloud	38
7.4	Before Uploading UNSW Dataset	38
7.5	After Uploading UNSW Dataset	39
7.6	Feature Selection using ADASYN	39
7.7	Generating Training Model	40
7.8	Accuracy of NB Algorithm	40
7.9	Accuracy of ANN Algorithm	41
7.10	Accuracy of Proposed Algorithm	41
7.11	Downloading the Test Dataset and Detecting the Attack	42
7.12.	Comparing the Accuracy of Different Algorithms	42

LIST OF TABLES

Table No	Description	Page No
6.1	Unit Test Case-1	31
6.2	Unit Test Case-1	32
6.3	Unit Test Case-1	32
6.4	Integration Test Case-1	33
6.5	Integration Test Case-1	33
6.6	Integration Test Case-1	34
6.7	System Test Case-1	35
6.8	System Test Case-1	35
6.9	System Test Case-1	36

INTRODUCTION

Chapter 1

INTRODUCTION

1.1 Background

As 5G technology continues to expand, wireless networks are becoming increasingly vulnerable to attacks, leading to heightened concerns about network security [1]. Traditional security measures like firewalls and encryption are often insufficient in addressing modern threats such as Denial of Service (DoS) and probing attacks [2]. Intrusion Detection Systems (IDS) play a crucial role in detecting these threats by classifying network traffic based on acquired attributes, enabling early anomaly detection and preventive measures [3]. Conventional machine learning methods are commonly used for attack detection, involving stages like data gathering, feature selection, and classification [4]. Feature encoding strategies enhance classifier performance, with increased discretization generally improving model effectiveness [6].

Efforts to improve IDS efficiency have led to innovations like using Principal Component Analysis (PCA) for feature selection and optimizing feature sets with edge density algorithms [7], [10]. Classification techniques like decision trees, support vector machines, and artificial neural networks are widely employed for categorizing network traffic. Hybrid approaches, combining classifiers like SVM with k-means clustering, have shown promise in enhancing detection capabilities [16], [17]. Despite the effectiveness of traditional machine learning techniques, adapting to dynamic network environments remains a challenge.

Deep learning (DL) has emerged as a powerful tool in intrusion detection due to its ability to autonomously generate key features without complex feature engineering. Techniques like Particle Swarm Optimization (PSO) applied to Deep Belief Networks (DBN) have shown high accuracy in detecting attacks [18]. Recurrent neural networks and convolutional neural networks (CNNs) are also utilized for intrusion detection tasks, with CNNs often exhibiting higher accuracy compared to traditional DL methods [20], [21], [22]. Ongoing research aims to develop novel attack recognition methods using techniques like residual networks and data augmentation algorithms like ADASYN.

1.2 Problem Statement

- Traditional intrusion detection systems often struggle to keep pace with the evolving threat landscape, leading to vulnerabilities and breaches.
- Most traditional security systems lack the ability to detect network traffic that appears normal but harbours new, undetectable attack types.

1.3 Existing system

- The Existing model was evaluated on two datasets Bot-IoT dataset and NSL-KDD dataset.
- These results demonstrate that the Existing model is effective in detecting malicious traffic in cloud computing environments but it demonstrated the potential of using a small number of features by contrasting the results with those of other classifiers.
- The paper by Attou et al. (2023) proposes a cloud-based intrusion detection model based on RF and feature engineering that achieves high accuracy in detecting malicious traffic.

Limitations

Limited evaluation: The proposed model was only evaluated on two datasets, the BotIoT dataset and the NSL-KDD dataset. But recall is still not well enough using NSL-KDD and BOT-IOT.

1.4 Proposed system

- My proposed architecture, however, introduces the power of deep learning to overcome the limitation of machine learning algorithms.
- This shift from traditional ML to deep learning promises a future of more robust and sophisticated intrusion detection.
- Integration of oversampling techniques like ADASYN to handle imbalanced datasets, improving model performance and robustness.
- Utilization of deep learning techniques, including Convolutional Neural Networks (CNN) for feature extraction in the ADASYN-SPC-CNN model.

1.5 Aim of the project

- To revolutionize cloud security by building a cutting-edge, deep learning-powered intrusion detection system.
- This system will automatically learn from vast amounts of data, adapt to new threats in real-time, and provide transparent insights for informed security decisions.
- Our goal is to safeguard cloud environments with unparalleled accuracy, efficiency, and interpretability, paving the way for a more secure and resilient digital future.

1.6 Objectives of the Project

- Develop and deploy a deep learning-based intrusion detection system tailored for cloud environments.
- Evaluate the system's performance in detecting various types of intrusions, including known and zero-day attacks, and assess its ability to adapt to new threats.
- Investigate the scalability of the intrusion detection system to accommodate the dynamic and resource-intensive nature of cloud computing.
- Implementation of these technology in latest datasets.

1.7 Summary

The advent of 5G technology has raised concerns about network security due to increased vulnerability to Denial of Service (DoS) and probing attacks. Traditional intrusion detection systems (IDS) often struggle to navigate its evolving threats sustained and cannot detect new types of attacks hidden inside similar to normal network traffic. To address these challenges, a shift towards deep learning based intrusion is proposed detection systems so, using techniques such as ADASYN for feature extraction to deal with imbalanced data with the aim of ensuring unparalleled accuracy and flexibility in cloud environments that can be real-time security threat detection and optimization s refined. Configuring an IDS and customizing cloud security.

LITERATURE SURVEY

Chapter 2

LITERATURE SURVEY

2.1 Cloud-Based Intrusion Detection Approach Using Machine Learning Techniques [DOI: 10.26599/BDMA.2022.9020038]

Description: Cloud computing (CC) is a novel technology that has made it easier to access network and computer resources on demand such as storage and data management services. In addition, it aims to strengthen systems and make them useful. Regardless of these advantages, cloud providers suffer from many security limits. Particularly, the security of resources and services represents a real challenge for cloud technologies. For this reason, a set of solutions have been implemented to improve cloud security by monitoring resources, services, and networks, then detect attacks. Actually, intrusion detection system (IDS) is an enhanced mechanism used to control traffic within networks and detect abnormal activities. This paper presents a cloud-based intrusion detection model based on random forest (RF) and feature engineering. Specifically, the RF classifier is obtained and integrated to enhance accuracy (ACC) of the proposed detection model. The proposed model approach has been evaluated and validated on two datasets and gives 98.3% ACC and 99.99% ACC using Bot-IoT and NSL-KDD datasets, respectively. Consequently, the obtained results present good performances in terms of ACC, precision, and recall when compared to the recent related works.

Limitations

- Requires large training data
- Requires ongoing maintenance

2.2 A Deep Learning based Multi-Agent System for Intrusion Detection

Description: Intrusion detection systems play an important role in preventing attacks which have been increase rapidly due to the dependence on network and Internet connectivity. Deep learning algorithms are promising techniques, which have been used in many classification problems. In the same way, multi-agent systems become a new useful approach in intrusion detection field. In this paper, author propose a deep learning-based multi-agent system for intrusion detection which combines the desired features of multi-agent system approach with the precision of deep learning algorithms. Therefore, author created a number of autonomous, intelligent and adaptive agents that implanted three

algorithms, namely autoencoder, multilayer perceptron and k-nearest neighbors. Autoencoder is used as features reduction tool, and multilayer perceptron and k-nearest neighbors are used as classifiers. The performance of our model is compared against traditional machine learning approaches and other multi-agent system-based systems. The experiments have shown that our hybrid distributed intrusion detection system achieves the detection with better accuracy rate and it reduces considerably the time of detection.

Limitations

- Complexity
- Computational requirements

2.3 Cloud-based Real-time Network Intrusion Detection Using Deep Learning

Description: Deep learning has increased in popularity with researchers and developers investigating and using it for various use cases and applications. This research work focuses on realtime network intrusion detection by making use of deep learning. A cloud-based prototype system was developed to investigate the capability of deep learning based binomial classification and multinomial models to detect network intrusions in real-time. An evaluation study was carried out using the benchmark NSL-KDD dataset to compare deep learning models built using H2O and DeepLearning 4J libraries, with other commonly used machine learning models such as Support Vector Machines, Random Forest, Logistic Regression and Naive Bayes. The results showed that the choice of the deep learning library is an important factor to consider for realtime applications. The H2O deep learning based binomial and multinomial models generally outperformed the other models, achieving over 99.5% accuracy using cross-validation on the NSL-KDD training dataset and over 83% accuracy on the test dataset.

Limitations

- Library Dependency
- Dataset Limitation

2.4 Cloud based Intrusion Detection System using Convolution and Supervised Machine Learning

Description: The ubiquity and pay-for-use services offered by cloud computing draw more

customers to use it as a result of these facilities. Even if cloud computing has many benefits, there are also some drawbacks. Cloud security features including confidentiality, availability, and integrity are susceptible to assaults. Therefore, security solutions are necessary for both cloud service providers and consumers in order to identify assaults and enhance cloud security. The biggest problem with cloud computing is security. The cloud has a variety of incursions. People are becoming increasingly conscious of the significance of network security as computer network technologies advance swiftly and internet technology develops more quickly. The greatest problem in computing is network security because assaults are happening more frequently. Due to these factors, a series of techniques known as intrusion detection systems (IDSs) has developed to prevent unauthorized use of a network's resources. In this study, author will explore convolutional deep learning method along with supervised methods likes support vector machines and KNN models, and author will propose a hybrid solution for intrusion detection using CNN, SVM and KNN models.

Limitations

- Data Privacy and Security Concerns
- Dependency on Quality Training Data
- Resource Intensive
- Complexity and Maintenance
- Limited Generalization

2.5 Enhanced Security in Cloud computing using Neural Network and Encryption

Description: With the fast advancement in cloud computing, progressively more users store their applications and data on the cloud. Cloud computing has lots of features, e.g. virtualization, multi-user, efficiency, cost savings, and most importantly security. Machine learning approaches based on neural networks are being widely applied in cloud infrastructure when training is performed however this may produce possible privacy and security risk as direct access to raw data is required. To address this problem, author propose a new security design using Artificial Neural Networks (ANN) and encryption to confirm a safe communication system in the cloud environment, by letting the third parties access the data in an encrypted form for processing without disclosing the data of the provider party to secure important information. In this paper, to train neural networks using encrypted data author considered the Matrix Operation-based Randomization and

Encipherment (MORE) technique, based on Fully Homomorphic Encryption (FHE). This technique allows the computations to be performed directly on floatingpoint data within a neural network with a minor computational overhead. Author examined the speech and voice recognition problem and the performance of the proposed method has been validated in MATLAB simulation. Results showed that applying neural network training with MORE offers improved accuracy, runtime, and performance. These results highlight the potential of the proposed method to protect privacy and provide high accuracy in a reasonable amount of time when compared to other state-of-the-art techniques.

Limitations

- Computational Overhead
- Key Management Complexity
- Compatibility Challenges
- Latency Issues
- Resource Intensiveness

2.6 Hybrid Intrusion Detection System using Machine Learning Techniques in Cloud Computing Environment

Description: Intrusion detection is one essential tool towards building secure and trustworthy Cloud computing environment, given the ubiquitous presence of cyber attacks that proliferate rapidly and morph dynamically. In our current working paradigm of resource, platform and service consolidations, Cloud Computing provides a significant improvement in the cost metrics via dynamic provisioning of IT services. Since almost all cloud computing networks lean on providing their services through Internet, they are prone to experience variety of security issues. Therefore, in cloud environments, it is necessary to deploy an Intrusion Detection System (IDS) to detect new and unknown attacks in addition to signature based known attacks, with high accuracy. In our deliberation author assume that a system or a network “anomalous” event is synonymous to an “intrusion” event when there is a significant departure in one or more underlying system or network activities. There are couple of recently proposed ideas that aim to develop a hybrid detection mechanism, combining advantages of signature-based detection schemes with the ability to detect unknown attacks based on anomalies. In this work, author propose a network based anomaly detection system at the Cloud Hypervisor level that utilizes a hybrid algorithm: a combination of K-means clustering algorithm and SVM classification

algorithm, to improve the accuracy of the anomaly detection system. Dataset from UNSW-NB15 study is used to evaluate the proposed approach and results are compared with previous studies. The accuracy for their proposed K-means clustering model is slightly higher than others. However, the accuracy author obtained from the SVM model is still low for supervised techniques.

Limitation

- Complexity in Implementation
- Resource Intensive
- Training Data Quality Dependency
- Integration Challenges

2.8 Summary

This summary provides insights into various approaches to enhance security in a cloud computing environment, in particular using traditional machine learning techniques such as random forest focused intrusion detection, and techniques a more advanced such as deep learning and autoencoders and k-nearest neighbors .Used by these methods of accuracy, detection rates f show promise to improve but also pose challenges such as computational complexity, reliability to large sets of training data, integration challenges arise etc. Furthermore, concerns about resource intensity, data privacy, and compatibility with existing systems are highlighted in the of the methods described.

REQUIREMENT SPECIFICATIONS

Chapter 3

REQUIREMENT SPECIFICATION

The requirement specification is a compulsory document in project management system wherever it finds its use. Although its first task is to explain details about the project targets, system operations limits and desired end serves. This specification is like a blueprint of the development team, the stakeholders as well as other parties who are part of the process. Therefore, it ensures that all involved parties have a common understanding of the built product and how it is going to function. By concretely defining specifications for examples including user interfaces, system behaviors, performance standards and regulatory standards the scope helps in diminishing the likelihood of misunderstandings and ambiguities in the development stage. Furthermore, it helps as a medium of communication which allows many participant to work together effectively through interactive meetings and discussion. In brief the detailed specification requirement is a tool used to set the expectation as far as the project objectives and the term and to ensure the product satisfy the user's needs.

3.1 Hardware Requirements

- **Operating System** : Windows only
- **Processor** : i5 and above
- **Ram** : 8gb and above
- **Hard disk** : 25gb and above

3.2 Software Requirements

- Anaconda 3
- Tensor flow
- AWS S3

3.3 Tool Usage

A. SPYDER

Spyder is a widely used open- source platform, an integrated development environment (IDE) for academic endeavors such as scientific computing, data analytics, and development embracing the Python programming language. It has got user-interface which is based on the top tools therefore Python developers and data scientists get efficient work done. We list down the prominent features of Spyder as follows.

- **Editor:** Spyder renders a code editor with syntax highlighting, code auto-completion and code navigation features that help Python code better in writing and editing.
- **Variable Explorer:** With Spyder, you can investigate your variables in a convenient Variable Explorer that shows all data types and dimensions of your code objects together with their current values, etc. This capability caters greatly for data analysis and debugging.
- **Debugger:** Spyder having at your side a debugger means you can use it to put breakpoints, run through your code, inspect variables, and figure out the bugs when your program runs.
- **Integrated Documentation:** The Spyder distribution tool enables the Python documentation to be available for the user, plus the function signatures for simple derivation and other relevant data when coding.
- **Profiler:** Spider has a profile tool to detect and analyze the performance of Python code. This tool, for instance, will help find places where Python code performs inadequately and give insight into optimization areas.
- **Plots and Visualization:** Spyder has interactivity of data visualization thanks to integration with Matplotlib plotting and other libraries. Users have the chance to make plots, charts, and graphs on the spot by involving those options in the IDE.

Totally, Spyder is an advanced and user-friendly IDE that is suitable for scientific computing and data analysis using Python programming language. It encompasses an environment development which comprises, among others, basic tools and features that will speed up coding, debugging, data exploration, as well as visualization endeavors.

B. AMAZON S3

AWS S3 (Simple Storage Service) presents a highly flexible and secure cloud storage tool with a wide range of features prepared by Amazon Web Service (AWS). It offers a variety of use cases and tools that make it essential for many applications.

It offers a variety of use cases and tools that make it essential for many applications:

- **Cloud Storage:** Through AWS S3, developers benefit from readily scalable and durable object storage system for the storage and recovery of data, examples of which are multimedia files like images, videos, documents, and backups.
- **Web Hosting:** A Static Site Hosting solution can be provided by S3 to host static websites and commonly used web assets such as HTML, CSS, JavaScript files, and media content. It is aimed at provisioning of state free and speedy content distribution tools through the AWS's geographically extensive net of data centers.
- **Data Backup and Recovery:** S3 provides one of the cheapest dedicated backup and disaster recovery datasets, to high stakes organizations. Its durability and availability benefits support data being accurate but also accessible for use in case the data is lost or destroyed.
- **Big Data Analytics:** S3 being a common platform supports services such as Amazon Athena, Amazon Redshift, and AWS Glue for advanced analytics of big data. It contains warehouse relational databases, data lakes, and data storage repositories hence allowing bulk data operation and analysis.
- **Data Archiving:** S3 bucket's performance classes are S3 Glacier and S3 Glacier Deep Archive. They were specifically designed for infrequently accessed, long-term archival, and cold storage. They are able to store all the infrequently used data at low cost but of high durability because of the high bandwidth availability.
- **Collaboration and Sharing:** The S3 feature enables team work and shared of files and data among teams and applications. It adopts the access control mechanisms, versioning, and encryption to maintain safe data hence privacy.
- **IoT Data Storage:** S3 is great for use with Open or Closed IoT devices. It enables to keep large numbers of IoT data streams running as it is considered as a storage solution of IoT applications.

Overall, S3 saving from AWS is important for the cloud computing and also offers scalable and safe storage for different type of applications from different companies. It has the right

bells and whistles, integration capacity and budget friendliness making it a primary cloud data storage and management choice.

C. Python Programming

Python is a powerful and universal programming language that combines functionalities with many tools and libraries that can be used for a plethora of different uses. Here are some common use cases and tools associated with Python programming:

Here are some common use cases and tools associated with Python programming:

- **Web Development:** Python platforms like Django and Flask are examples of frameworks that create efficient paths to web development. These environments act as dev kits, providing the tools to build and run scalable as well as reliable web apps.
- **Data Analysis and Visualization:** It is Python most beloved in the field of data science and analytics. Toolkits like NumPy, pandas, and Matplotlib comprise numerical support to data manipulation, analysis, and visualization.
- **Machine Learning and Artificial Intelligence:** The Python framework has been growing in Machine Learning and AI practice with the support of Scikit learn, tensorflow, and PyTorch python libraries. These libraries are helping learners to form and train different machine learning models.
- **Scripting and Automation:** Python is a language of choice for the work with the scripts and automation of routine processed. It allows us to use a short and clear syntax for writing scripts via which makes it possible to automate many operations, including file processing, data processing, and system administration.
- **Scientific Computing:** Python can be applied in a wide range of scientific activities related to computation. because all the mathematical operations are done by the SciPy and SymPy libraries. Among the other useful tools, there are various functions for use in scientific computations, solving of math problems and virtually running models.
- **Game Development:** Python framework can be applied for creating games with Pygame as a library. This offers a playground for creating virtual flash games and the graphical component applications.
- **Internet of Things (IoT):** Python is capable of handling the IoT tasks smoothly because it is both simple and flexible. No matter a library such as Raspberry and

MicroPython, a developer can operate and control the IoT devices and sensors through programming them.

The mentioned ones are just the few number of applications for Python, however the wide applications of the language exists. The Python's popularity as well as its huge library of libraries and tools might cause it to become first choice of many programmers in different topics.

3.4 Functional Requirements

- **Packet capture and preprocessing:** Network packets must be captured and pre-processed to extract appropriate resources for the system analysis.
- **Deep learning model integration:** Combining Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). To examine temporal and spatial patterns of networking.
- **Introduction to Contrast:** To use algorithms to detect anomalies and irregular patterns in the network data indicating possible intrusion.
- **Attack Classification:** To develop methods for classifying known anomalies into specific attack categories providing insight into the nature of the attack.
- **Continuing Education:** Enables the system to learn continuously through well-designed threat scenarios new data updating the model.
- **Real-time processing:** Ensure real-time processing capability to quickly detect and respond to intrusions e.g.They are the ones who do it.
- **Exchanges:** Develop the planning process accordingly without loss of performance.
- **Tree Recording and Reporting:** Use logging techniques to record known discrepancies and generate detailed reports for further analysis.
- **Interface with the existing security system:** Allows seamless integration with existing security infrastructure to enhance the entire network Security.
- **Configurability:** Provide configurable parameters for changing sensitivity levels and optimizing the system depending on the specific network conditions.

3.5 Non-functional Requirements

- **Write:** The system must exhibit high performance, ensuring minimal impact on network speed and it lasts.
- **Specifically:** To achieve high accuracy in infiltration detection and while reducing false positives maintain system credibility.
- **Flexibility:** The system must adapt to changes in network traffic patterns and emerging threats showing resilience to changing attack tactics
- **Exchanges:** Ensure that the system is optimized to accommodate the increasing network demand and changing security requirements.
- **Security:** Implement robust security measures to protect the input detection system from tampering or the use of malicious companies.
- **Useful Benefits:** Establish a flexible user interface for system administrators to administer and maintain NIDS in an effective manner.
- **Research:** Provides comprehensive auditing methods and documentation for audits and post-event analyses.
- **Availability:** Also ensure high availability to ensure continuous monitoring and access when network activity peaks.
- **Resources:** Adjust resource efficiency, including memory and processing power, to improve performance hardware configurations.

3.6 Summary

The specification of the hardware and software requirements for the intrusion detection system (IDS) project involve the operation system, processor, memory, and storage, with the focus being given to Windows, Intel®, i5 or higher processor, and least 8GB, RAM, and 25GB hard disk space. Tools like Spyder or IDE for Python programming, Amazon S3 for your cloud storage, and Python programming itself would be of critical importance. Functional requirements must include: data capture, integration of deep learning models, anomaly detection, attack classification, online detection, logging, integration of the system, configurability, and scalability. The non-functional requirements underline and also cover the performance, accuracy, flexibility, security, usability, comprehensive documentation, availability, resource efficiency and scalability, evaluating the system of the IDS effectiveness and reliability.

SYSTEM DESIGN

Chapter 4

SYSTEM DESIGN

The system design section in technical documents or project reports is the most critical part that isolates the block diagram, the course of actions, and workflow, along with depicting the overall system or product details and the associated processes involved. This documentation detailing the integration of each of the highlighted components of the system (hardware, software, database, network components and interfaces) provides a full picture on how this hardware and software will work together to create the intended system. At the end of the system design chapter, the guide is a user manual of the system design for all stakeholders, including developers, architects, project teams and so on. The manual can act as a framework of the proposed system design for a successful integration, implementation and deployment that emphasizes quality and reliability while putting the end users first.

4.1 System Architecture

Figure 4.1 illustrates the workflow of this study that uses the deep learning (DL) process. During Data Loading, the USNW-NB15 dataset is imported into the system. Furthermore, the feature vector of this dataset is reduced by PCA to lower its dimension entailing better execution and interpretation of the analytic process.

The Data Splitting step divides the dataset into two subsets: a Training Set and a Test Set. Training set is used for training of the machine learning model and the test set is purposefully excluded from this process as it is used for testing the model with them.

Therefore, ADASYN is carried out over the Training Set. Consider ADASYN (Adaptive Synthetic Sampling Approach for Imbalanced Learning) as a type of data oversampling algorithm which tries to enhance the accuracy of the classifier by generating a synthetic minority class sample.

The subsequent step would be to carry out the Special Spectral Feature Extraction operation, which is done twice, featuring spectral data. The composition of this spectral datasets becomes the input features for the deep learning model.

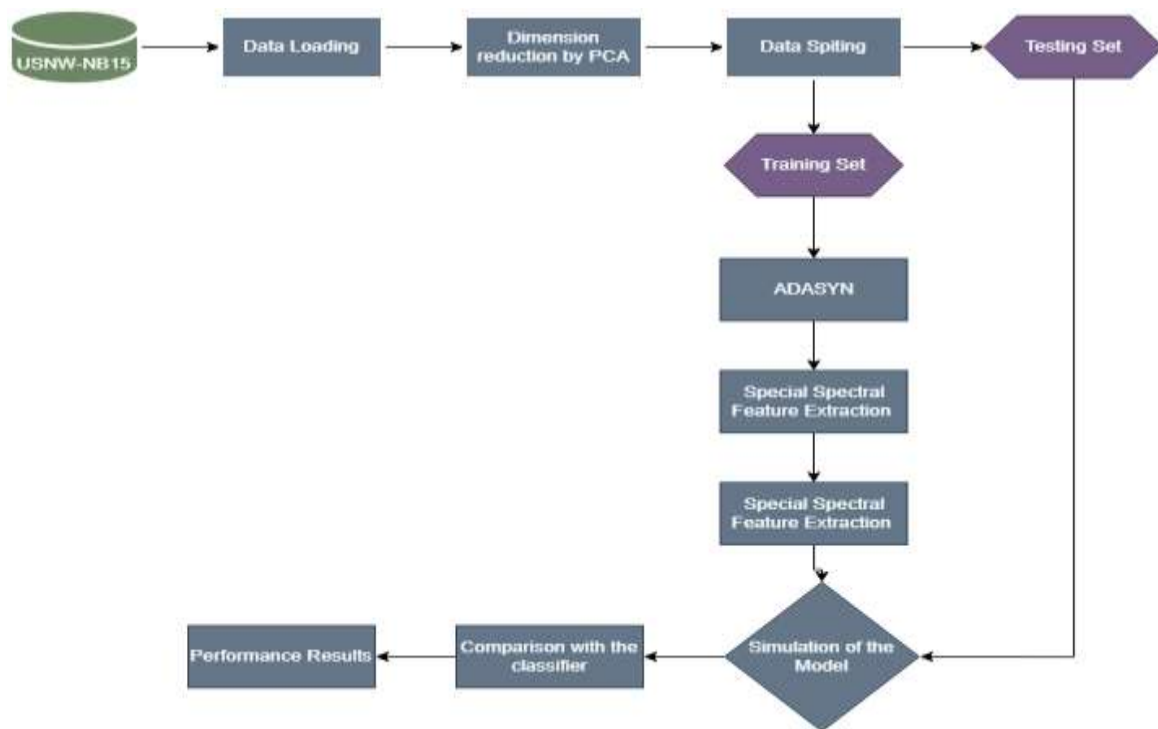


Figure 4.1: The overview of the proposed methodology

In the course of Model Simulation process the machine learning model's getting trained based on features provided by the Training Set. Here, the model is the training process which aims to fit it with the data, and searching for and selecting the optimal parameters.

The experiment also includes the Comparison with a classifier stage, where the trained model performance is compared with that of a classifier. With such a comparison, it is hence much easier to evaluate the model's effectiveness and find the relevant areas that may need upgrading.

Next, the Performance Results are measured all the data that represents whether the model has been correctly produced or not this values may be as accuracy, precision, recall or any other metrics relevant. The data yielded in such a research is crucial for assessing the model's performance and subsequently introducing necessary alterations or enhancements.

4.2 Data Flow Diagrams

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

Level 0 Data flow diagram

A context-level or level 0 data flow diagram shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the Level 0 DFD) the system's interactions with the outside world are modeled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization.

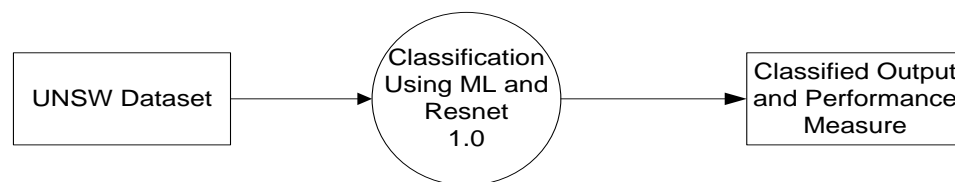


Figure 4.2 Level 0 Data flow diagram

Level 1 Data flow diagram

The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole.

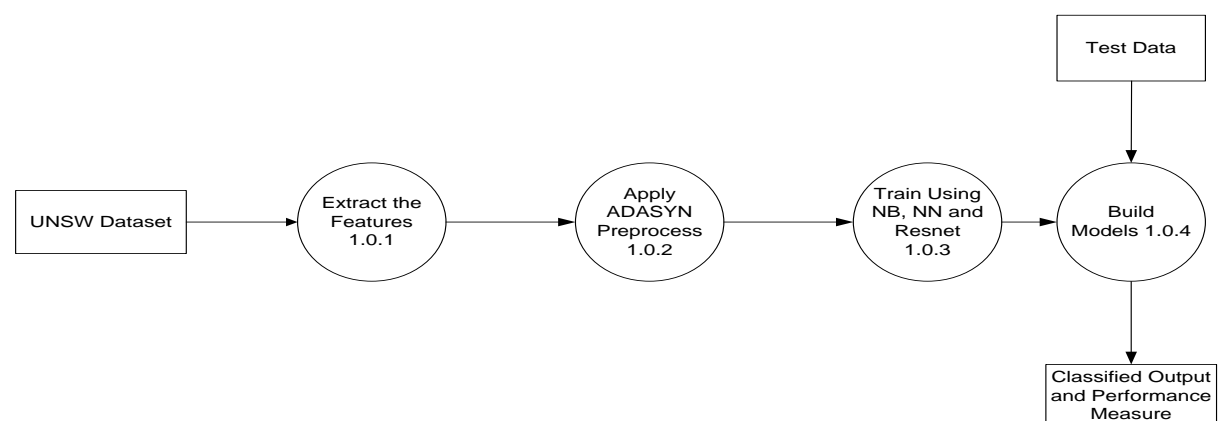


Figure 4.3 Level 1 Data Flow Diagram

4.3 UML Diagrams

4.3.1 Use Case Diagram

A use case diagram is a type of behavioral diagram created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

Figure 4.3.1 refers to the Data classification process utilizing the UNSW datasets is depicted in step-by-step way in use case diagram. In a first steps of process, input data and the UNSW dataset are loaded up into the system, thus allowing all the required information to be ready up for further processing. Next, the preprocessing step is executed, in which the data stored in the feature file undergoes different transformations and brings to use certain preparation methods of data that is shown as the second step in Figure 4.3.1. These are the steps that include data cleaning, normalization, feature selection or any other operation with the goal of good conditioning before your data for the classification task to function perfectly.

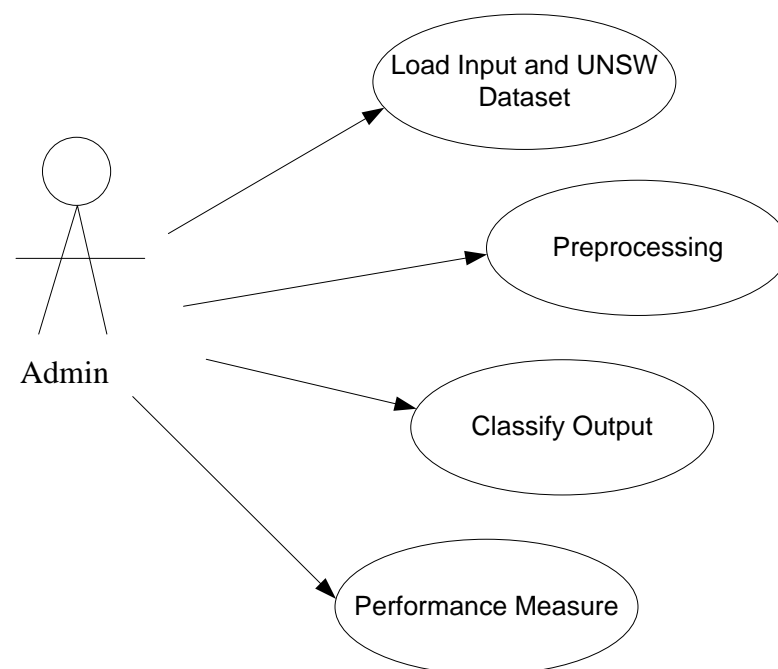


Figure 4.4 Use Case Diagram

When at the last the preprocessing outcome will be accomplished is then the classify output will begin, presented as step (3) in Figure 4.3.1. If the prepared data is fed into a classification algorithm or model, the next stage is the model creation. These models learn by training the UNSW dataset and then they make decisions based on the data and patterns. First of all, after classification Figure 4.3.1 shows, then the fourth step comes. Step 3 goes under performance measure. By comparing the true and predicted classifications from the UNSW dataset, it judges how the model is working. This means that, usually, performance is the metrics being calculated, like accuracy, precision, recall, or F1-score in order to determine the classifier's performance and to be able spot areas for improvement or refinement.

4.3.2 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

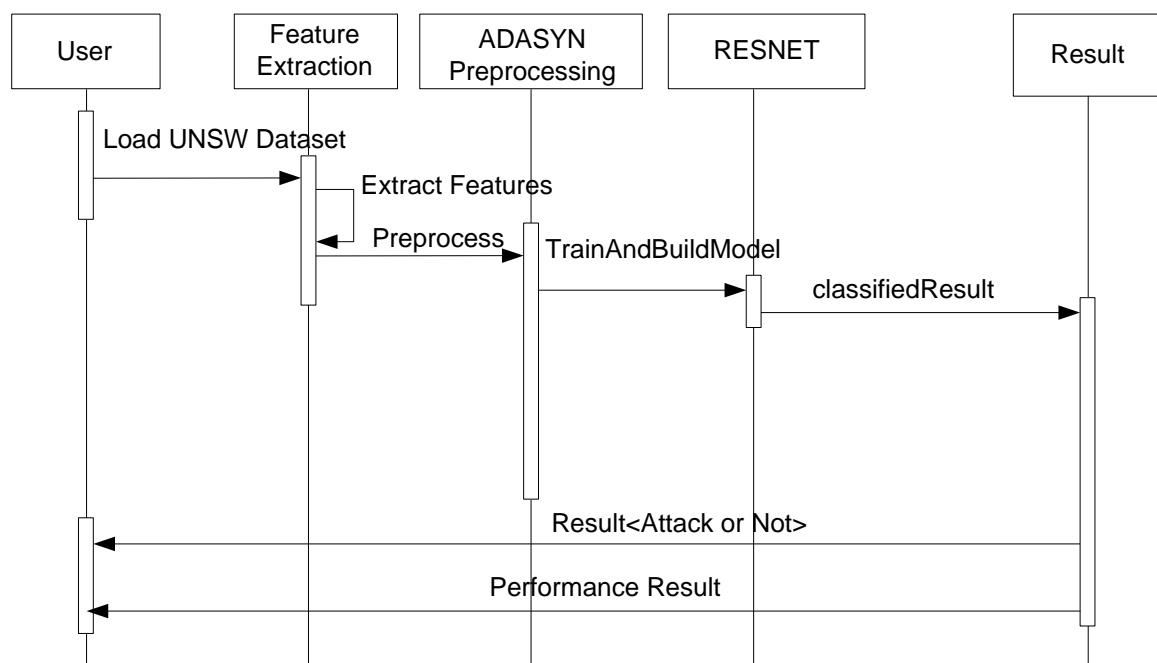


Figure 4.5 Sequence Diagram

Anomaly detection systems (ADS) constitute an irreversible part of the cybersecurity puzzle by discovering irregular network traffic flows that hint at possible threats. The figure below depicts the feature extraction procedure that is utilized to develop an ADS with UNSW dataset.

First, the system loads the UNSW dataset. This set of data represents a resource for cybersecurity research since the network traffic data is annotated. These labels help you distinguish between normal traffic and malicious traffic behavior.

In the end, the system will take out the appropriate features from the data. These features, on the other hand, may be defined as the set of attributes that allow distinguishing the normal network activity from cyber attacks. Illustrations can be items such as the number of packets sent, IP source and destination addresses, or protocol types used.

Then, feature extraction is performed, followed by data preprocessing. This is to make sure that all characteristics are on a similar level. Rescaling or normalization as a rule are used during preprocessing. This, in turn, leads to more accurate machine learning algorithms being used in the future.

With the engineered features at its disposal, the system undergoes the process of model training. The prepared data is then used to train a machine learning model which will, most probably, be a classification model. In training the model, it gets to distinguish between the normal and abnormal traffic patterns with the features it get to extract.

After the training, the model is all set for classifying new network traffics. The trained model proffers classification output in response to any unforeseen network data. In most cases, it will appear as a label that will either state that the traffic is regular or abnormal.

Lastly, the system transforms the classifying output into a comprehensible finding. It means whether the network traffic has been classified as an attack or not. This is an important information for the security person to follow and take the required action.

A performance evaluation is the next block of the chart. The next step in the process is to evaluate the trained model with metrics such as accuracy, precision, and recall. By studying these indicators, security officers can infer on the model's ability to detect anomalies and make changes if necessary.

At a glance, the block diagram illustrates the typical data processing of ML-based anomaly detection. The classifier is trained off the data on the UNSW dataset to distinguish network

traffic patterns. This strategy is very effective on IDS that are always evaluating network for any intrusion attempts.

4.4 Summary

This architecture comprises of a workflow, which includes data loading and splitting, oversampling, spectral feature extraction, model simulation, comparison with classifiers, and performance evaluation for development of a precise intrusion detection system. The data flow diagram shows the flow of data inside the system, which includes data loading, preprocessing, classification, and performance evaluation process. Use case diagrams unfold the processes involved in the data classification through the use of UNSW datasets, including input data loading, preprocessing, classification, and performance measurements. The sequences in this diagram will illustrate the feature extraction process that the system uses for the anomaly detection with the UNSW dataset, which includes loading data, feature extraction, preprocessing, model training, classification, and evaluation of the performance.

SYSTEM IMPLEMENTATION

Chapter 5

SYSTEM IMPLEMENTATION

The chapter System Implementation is the phase where the design of the intrusion detection system moves to its actual implementation: starting from the theoretical design and ending in its practical use. Here, you will discover the hands-on manifestation of proposed methodology which will include the setting-up of hardware and software components, configuration of tools and environments, integrating algorithms and models and running data processing pipelines. The major reason for this chapter is achieving the operating structure, which was a theoretical process in the previous parts of the chapter, and making that structure capable of finding and responding to network intrusions efficiently. Through careful realization, the chapter indeed aims to pass the tests of feasibility and efficacy, clearing the way for advanced stages such as exhaustive testing, evaluation and assessment going to be presented in the subsequent stages after the project lifecycle setup.

5.1 Modules and Components

5.1.1 ADASYN

It stands for "Adaptive Synthetic Sampling Method". It's a data sampling technique used in machine learning for imbalanced classification problems. Imbalanced classification occurs when one class in the dataset has significantly more samples than the other class(es).

ADASYN works by generating synthetic samples for the minority class (the class with fewer samples) based on its distribution in the feature space. It focuses on the regions where the class density is low and hence is particularly effective in dealing with situations where the minority class is spread across different clusters in the feature space.

An intuitive example: Imagine classifying for credit card fraud. If there are only 5 fraudulent transactions per 1,000,000 transactions, then all our model has to do is predict negative for all data, and the model will be 99.9995% accurate! Thus, the model will most likely learn to “predict negative” no matter what the input data is, and is *completely useless*! To combat this problem, the data set must be balanced with similar amounts of positive and negative examples.

Some traditional methods to solve this problem are under-sampling and over-sampling. Under-sampling is where the majority class is down sampled to the same amount of data as the minority class. However, this is extremely data inefficient! The discarded data has important information regarding the negative examples.

Imagine building a house cat classifier, and having 1,000,000 images of different species of animals. But only 50 are cat images (positive examples). After down sampling to about 50 negative example images for a balanced data set, we deleted all pictures of tigers and lions in the original data set. Since tigers and lions look similar to house cats, the classifier will mistake them for house cats! We had examples of tigers and lions, but the model was not trained on them because they were deleted! To avoid this problem of data inefficiency, over-sampling is used. In over-sampling, the minority class is copied x times, until its size is similar to the majority class. The greatest flaw here is our model will overfit to the minority data because the same examples appear so many times.

Here's how ADASYN typically works:

- Identify the minority class and its samples.
- For each sample in the minority class, calculate its k -nearest neighbors.
- Compute the imbalance ratio for each sample, which is the ratio between the number of samples in the majority class and the number of samples in the minority class within the neighborhood.
- Generate synthetic samples for each minority class sample, with the number of synthetic samples proportional to its imbalance ratio. This means that samples in regions of higher imbalance (where the majority class is dominant) will have more synthetic samples generated.
- Combine the original minority class samples with the synthetic samples to form the augmented dataset.

ADASYN is particularly useful when the class imbalance is severe or when the classes are not linearly separable. By creating synthetic samples in the regions where the minority class is underrepresented, ADASYN helps improve the model's ability to learn from the minority class and thus improve overall classification performance.

5.1.2 Over sampling and Under sampling in ADASYN

ADASYN (Adaptive Synthetic Sampling) is a variant of the Synthetic Minority Over-sampling Technique (SMOTE), which is used for addressing class imbalance problems in machine learning. Class imbalance occurs when one class in the data significantly outnumbers the other class(es), leading to biased models that may perform poorly on the minority class. ADASYN specifically focuses on generating synthetic samples for the minority class that are difficult to classify, aiming to improve the model's performance.

Here's how over-sampling and under-sampling are incorporated in ADASYN.

A. Oversampling

- ADASYN initially performs the Oversampling on minority classes to balance the class distribution. It creates synthetic samples by interpolating between existing minority class instances.
- Unlike simple over-sampling techniques like duplication, ADASYN generates synthetic samples based on the distribution of the nearest neighbors of each minority class instance. This ensures that the synthetic samples are representative of the underlying distribution of the minority class.

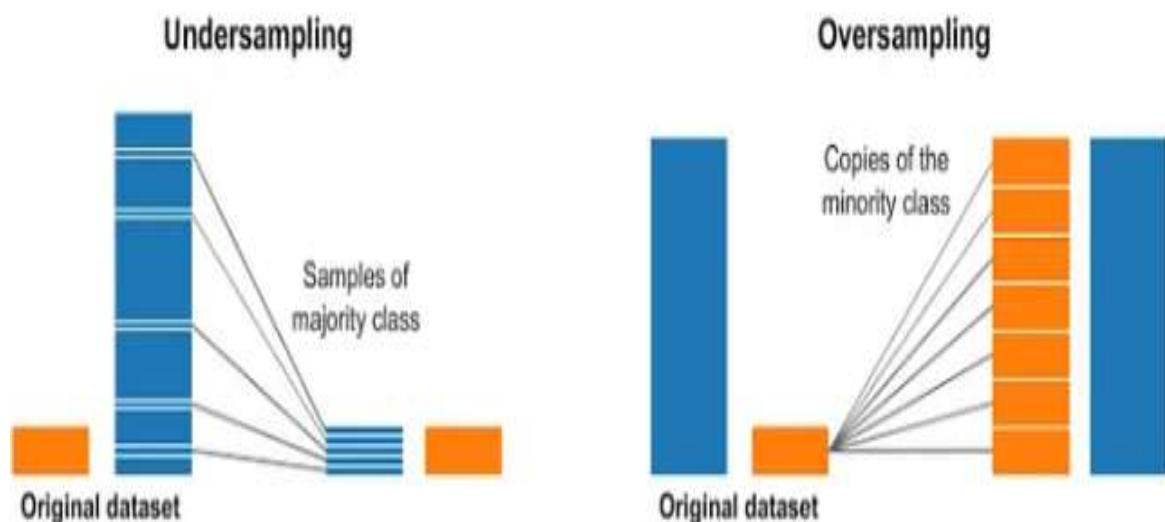


Figure 5.1 Undersampling and Oversampling

B. Undersampling

- After generating the synthetic samples ADASYN assesses the degree of class imbalance by comparing the number of minority class instances to the number of majority class instances in the data set.
- It then selectively under samples the majority class instances to the number of majority class instance in the data set.

The undersampling step helps to reduce the dominance of the majority class preventing the model from being biased towards it. However, it's important to note that under-sampling is applied after the over-sampling step in ADASYN, and the under-sampling rate is typically determined based on the desired balance between the classes.

In summary, ADASYN combines both over-sampling and under-sampling techniques to address class imbalance, with a focus on generating synthetic samples for the minority class while also reducing the dominance of the majority class through selective under-sampling. This approach aims to improve the model's ability to generalize to both classes more effectively than traditional over-sampling or under-sampling methods alone.

5.2 Residual Network

To solve the vanishing/exploding gradient problem, this work introduces the concept called Residual Blocks. It uses a technique called skip connections. Skip connection works by connecting activations of a layer to further layers, this is accomplished by skipping some layers in between forming a residual block. Resnets are built by stacking residual blocks together.



Figure 5.2 Illustration of Skip Connection

The idea behind ResNet is instead of allowing the layers to learn the mapping, the network is fitting the residual mapping. Instead of say $H(x)$, initial mapping, let the network fit. The advantage of skip connection is that if any layer is degrading the performance of architecture then it will be skipped by regularization. It helps in training the deep neural network without the negative impact of vanishing/exploding gradient.

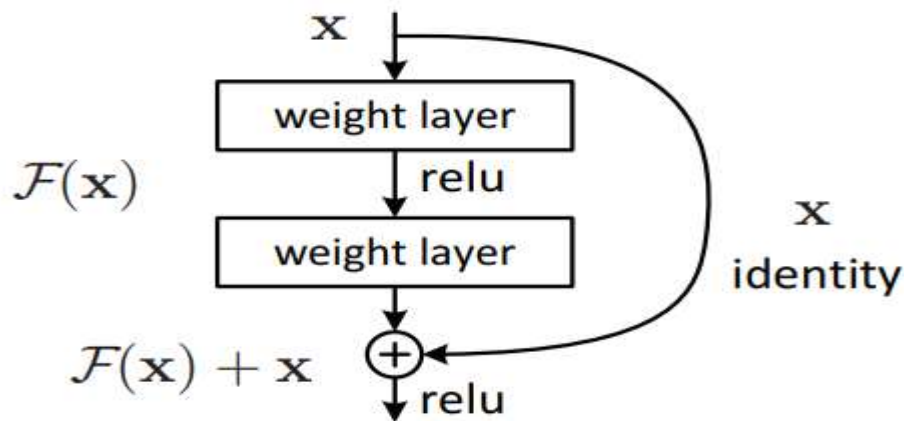


Figure 5.3. Skip Connection

Network Architecture: This network uses a 34-layer plain network architecture inspired by VGG-19 in which then the shortcut connection is added. These shortcut connections then convert the architecture into a residual network. Residual networks (ResNets) are a type of deep neural network architecture that introduced the concept of residual blocks. These blocks contain skip connections (also known as shortcut connections or identity mappings) that bypass one or more layers, allowing the network to learn residual mappings instead of directly learning the desired underlying mapping. This helps in alleviating the vanishing gradient problem and facilitates the training of very deep networks.

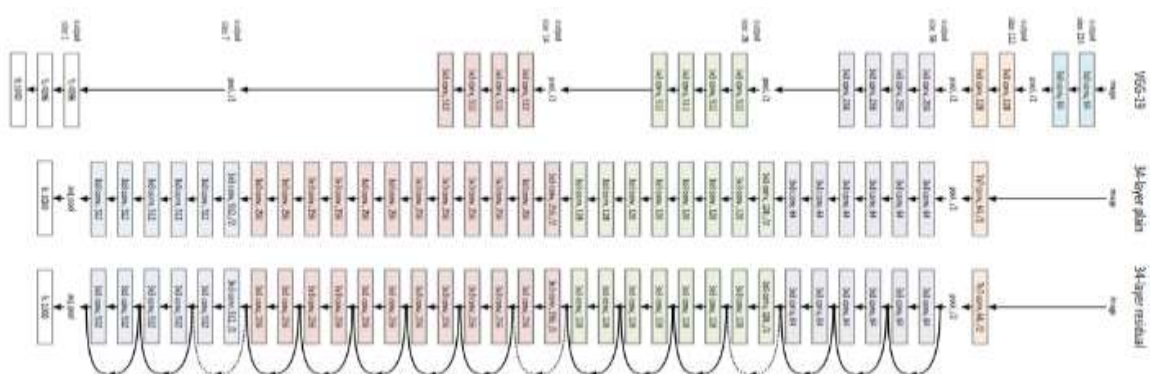


Figure 5.4 Network Architecture of Residual Network

Here's a breakdown of the network architecture of residual blocks with skip connections:

1. Input

- The input to a residual block is a feature map or activation from the preceding layer.

2. Convolutional Layer

- The input feature map undergo several convolutional layers. These convolutional layers apply various transformations to the input, learning hierarchical representations of the data.

3. Activation Function

- After each convolutional layer, an activation function (commonly ReLU - Rectified Linear Unit) is applied to introduce non-linearity into the network.

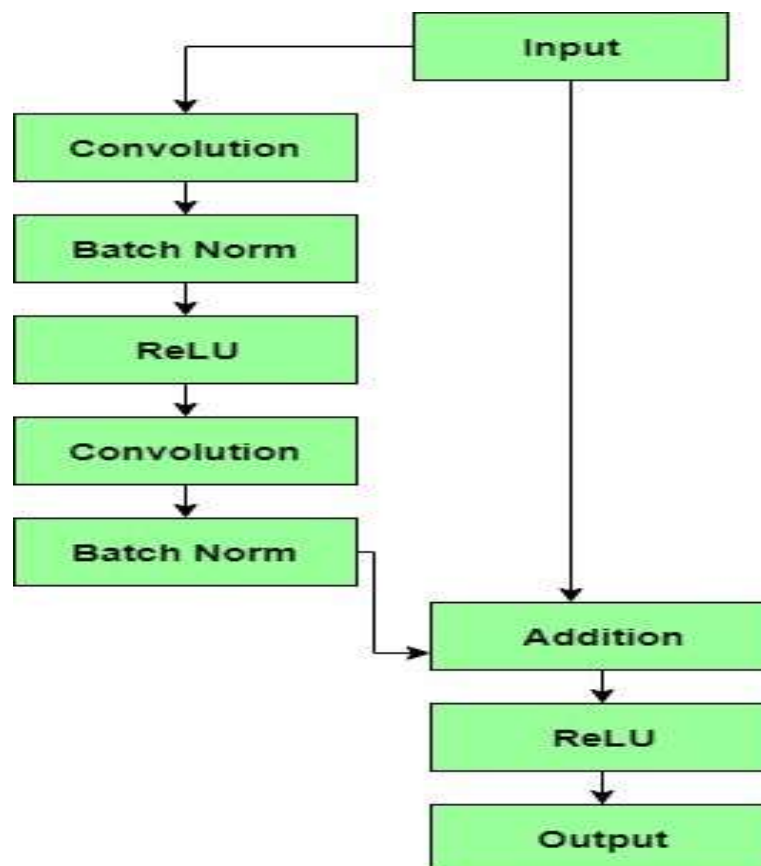


Figure 5.5 Flow Diagram of the Residual Network

4. Skip Connection

- The original input to the residual block is directly added to the output of the convolutional layers.
- This is a skip connection or the short cut connection.

- If the spatial dimensions of the input and output feature maps are the same, the shortcut connection simply performs element-wise addition.
- If the dimensions differ (e.g., due to different filter sizes or strides in convolutional layers), a linear projection (usually a 1x1 convolution) is used to match the dimensions before the addition.

5. Output

- The output of the residual block is the sum of the input to the block and the output of the convolutional layers.
- Mathematically, if x represents the input, $F(x)$ represents the transformation applied by the convolutional layers, and $H(x)$ represents the output of the residual block, then the output $H(x)$ is computed as:

$$H(x) = F(x) + x$$

The skip connection allows the network to learn residual functions, which are easier to optimize. If the identity mapping is the optimal mapping, the weights can be set to zero so that the output becomes equal to the input. This flexibility enables the network to effectively learn both the identity mapping and the residual mapping.

In summary, the network architecture of residual blocks with skip connections includes convolutional layers followed by a skip connection that adds the input to the output of the convolutional layers. This architecture enables the training of very deep neural networks by facilitating the flow of gradients and alleviating the vanishing gradient problem.

5.3 Algorithms and Pseudo codes

ADASYN Algorithm

Input: There are m samples in the training set where

$$i=1,2,3,4,\dots,m$$

$$Nl+Ns=m \text{ where}$$

$$Nl=\text{Large samples}$$

$$Ns=\text{Small samples}$$

Step 1: Calculation of samples imbalance degree

$$d= ns/nl$$

Step 2: Calculation of total number of samples that are needed for Synthesisation

$$Q=\beta(Nl-Ns)$$

Step 3: Calculate R_i using K-nearest neighbours

$$R_i=\gamma/k$$

Step 4: Normalization of R.

$$R_i'=\frac{R_i}{\sum_l^{Ns} R_l}$$

Step 5: Calculation of total number of samples that needs to be generated.

$$g_i = R_i' * G$$

$$G= \text{no of samples to be synthesized}$$

Output: New training dataset

UNSW-NB15

UNSW-NB15 is a universally recognized dataset used for NIDS training and development by the researchers and academics. It offers a variety of training opportunities alongside realistic traffic through the blend of real modern network traffic and synthetic attack simulations. This set of data consists of nearly 2 copies of human genomes. 49 features are employed in this dataset that represents 5 million records and includes IP addresses, protocols and packet statistics. Additionally, it deals with various categories of attacks, from DoS to worms and exploits. The idea that the innovative vaccine UNSW-NB15 is widely accessible makes it a source of fact for anyone working on intrusive detection methods. At the same time, take into consideration the size of the data that might need large processing units to get the job done, and class imbalance where there is unequal distribution of attacks with some types being scarcer than others.

5.4 Summary

In the System Implementation chapter, the theoretical design of the intrusion detection system is translated into practical use, encompassing hardware and software setup, algorithm integration, and data processing pipeline configuration. Key components like ADASYN and Residual Networks are discussed, with ADASYN addressing class imbalance through adaptive synthetic sampling and Residual Networks mitigating gradient vanishing/exploding issues. Algorithms and pseudo-codes are provided for implementation guidance, along with the introduction of UNSW-NB15 dataset, a crucial resource for NIDS development due to its realistic blend of network traffic and attack simulations, albeit necessitating significant processing power and addressing class imbalance challenges.

SYSTEM TESTING

Chapter 6

SYSTEM TESTING

The system testing for intrusion detection includes a complete evaluation of the IDS to make sure it works correctly in detecting and responding to security attacks coming from different networks. Such testing is critically wide, ranging from simulating different types of attacks to assessment of IDS decision-making abilities to analyzes of response mechanisms to verifying integration with existing security policies. Aiming at confirming the IDS's efficiency in the recognition of the existing and possible threats as well as the reduction of false alarms and the rapid incident reporting is the goal. IDS needs to prove scalability, performance, and reliability under regular network conditions if it is to work well in production systems.

6.1 Unit Testing

Unit testing is software testing approach that focus on the testing of isolated units or components of the software separately from other parts of the software. Each unit (usually a function or method) is tested to see if it passes and produces accurate output for the given inputs. Unit tests focus on individual components and execute them in a designated scope.

Table 6.1: Unit Test Case-1

TEST CASE ID	UTC-1
FUNCTIONALITY	To verify the isfloat function
ACTION	Valid Float Input
EXPECTED RESULTS	True
ACTUAL RESULTS	True
TEST RESULT	Pass

The Table 6.1 describes the Test Case to check the functionality of is_float function. The action is to give a valid float input to the function.

Table 6.2: Unit Test Case-2

TEST CASE ID	UTC-2
FUNCTIONALITY	To check Dataset Loading
ACTION	Load Dataset
EXPECTED RESULTS	The system loads the dataset without errors, and the loaded dataset is accessible for future processing
ACTUAL RESULTS	Dataset is Loaded
TEST RESULT	Pass

The Table 6.2 describes the test case to load a know dataset into the system. The action is to check if the dataset is loaded or not.

Table 6.3: Unit Test Case-3

TEST CASE ID	UTC-3
FUNCTIONALITY	Test Data Pre-processing Function
ACTION	Data Transformation
EXPECTED RESULTS	The data pre-processing functions accurately transform the input data as per the specified requirements.
ACTUAL RESULTS	Test Data Pre-Processed
TEST RESULT	Pass

The Table 6.3 describes the test case to apply data pre-processing functions to a dataset. The action is performed to verify the processed data.

6.2 Integration Testing

Integration testing deals with how the components and modules in a software system interconnect and relate. Unlike unit testing that assesses parts in an isolation mode, integration testing looks at how these components work together in forming a complete unit. A multi-level system-level testing verifies that system's operation stays in line with specified requirements and any inconsistencies, interface errors, or integration problems are identified early enough in the development process. These tests validate the fact that the software components work unanimously through the system as a whole and thus the system's integrity and worth are not compromised.

Table 6.4: Integration Test Case-1

TEST CASE ID	ITC-1
FUNCTIONALITY	Integration between Data Pre-processing and Deep Learning Models
ACTION	Data Integration
EXPECTED RESULTS	Models trained successfully on pre - processed data.
ACTUAL RESULTS	Model Trained Successfully
TEST RESULT	Pass

The Table 6.4 describes the test case to train deep learning models using the pre-processed data. The integration is between pre-processed data and training model.

Table 6.5: Integration Test Case-2

TEST CASE ID	ITC-1
FUNCTIONALITY	Model Performance Evaluation
ACTION	Model Accuracy
EXPECTED RESULTS	The model accurately predicts/classifies instances from the test data, demonstrating its effectiveness.
ACTUAL RESULTS	Evaluating the Model Accuracy using Test Data
TEST RESULT	Pass

The Table 6.5 describes the test case to load trained deep learning model and evaluate models accuracy using test data.

Table 6.6: Integration Test Case-3

TEST CASE ID	ITC-3
FUNCTIONALITY	API and Service Integration
ACTION	AWS S3 Integration
EXPECTED RESULTS	The system successfully connects to AWS S3 and retrieves the required test data, demonstrating seamless integration with external services.
ACTUAL RESULTS	Download test data from AWS S3.
TEST RESULT	Pass

The Table 6.6 describes the test case to connect to AWS S3 service and download test data from it.

6.3 System Testing

If successful integration testing is brought forth, system testing is the most significant part of the software development lifecycle, where the entire integrated system is analyzed to ensure it meets all requirements and functions without errors in addressing different environments. During this phase, many tests are conducted, which embrace functionality, performance, usability and security testing, in order to detect defects before release and to make sure the final product is secure enough for the client.

Table 6.7: System Test Case-1

TEST CASE ID	STC-1
FUNCTIONALITY	End-to-end Attack Detection
ACTION	Successful Attack Detection
EXPECTED RESULTS	The application correctly classifies normal and anomalous instances from the test data, displaying accurate results.
ACTUAL RESULTS	Test Data is Classified as Normal and Anomalous
TEST RESULT	Pass

The Table 6.7 describes the test case to Launch the GUI application then upload a dataset containing normal and anomalous instances. Pre-process the data then generate the training models and run the Deep learning algorithms for attack detection then download the test data from AWS S3 and initiate attack detection.

Table 6.8: System Test Case-2

TEST CASE ID	STC-1
FUNCTIONALITY	GUI Functionality Test
ACTION	Upload Dataset
EXPECTED RESULTS	The application successfully uploads the dataset file, displaying a confirmation message.
ACTUAL RESULTS	Dataset Uploaded Successfully
TEST RESULT	Pass

The Table 6.8 describes the test case to launch the GUI application and attempt to upload the dataset file.

Table 6.9: System Test Case-3

TEST CASE ID	STC-3
FUNCTIONALITY	Accuracy measurement and analysis
ACTION	High accuracy validation
EXPECTED RESULTS	To get higher accuracy than other algorithms
ACTUAL RESULTS	Achieved higher accuracy
TEST RESULT	Pass

The Table 6.9 describes the test case to check the performance of the proposed algorithm compared with other algorithms.

6.4 Summary

The System testing chapter comprises three main sections: Unity Testing, Integration Testing, and System Testing. Unit testing is concerned with the testing of individual units or components of the software in isolation. It makes sure they are functional. Integration testing involves the analysis of the interconnection and inter-operability of these components that should work as a system. System testing consistently scrutinizes the entire integrated system to make sure that it completely meets all the requirements and that it is performing as expected across different areas. Each section with test cases where they have to validate specific functionalities or interactions for the system to ensure it has a good performance and reliability.

RESULT AND ANALYSIS

Chapter 7

RESULT AND ANALYSIS

7.1 Snapshots



Figure 7.1 Attacker before uploading Test Dataset to the Cloud



Figure 7.2 Attacker after uploading Test Dataset to the Cloud

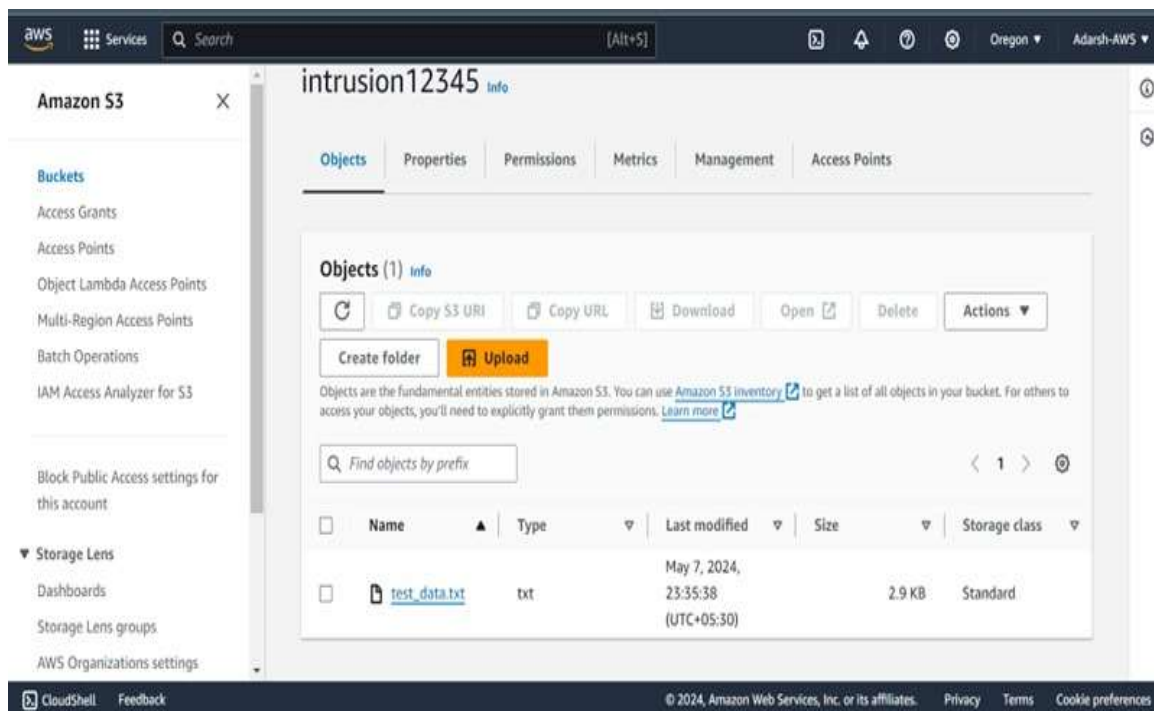


Figure 7.3 Test Data Uploaded to Cloud



Figure 7.4 Before Uploading UNSW Dataset



Figure 7.5 After Uploading UNSW Dataset



Figure 7.6 Feature Selection using ADASYN

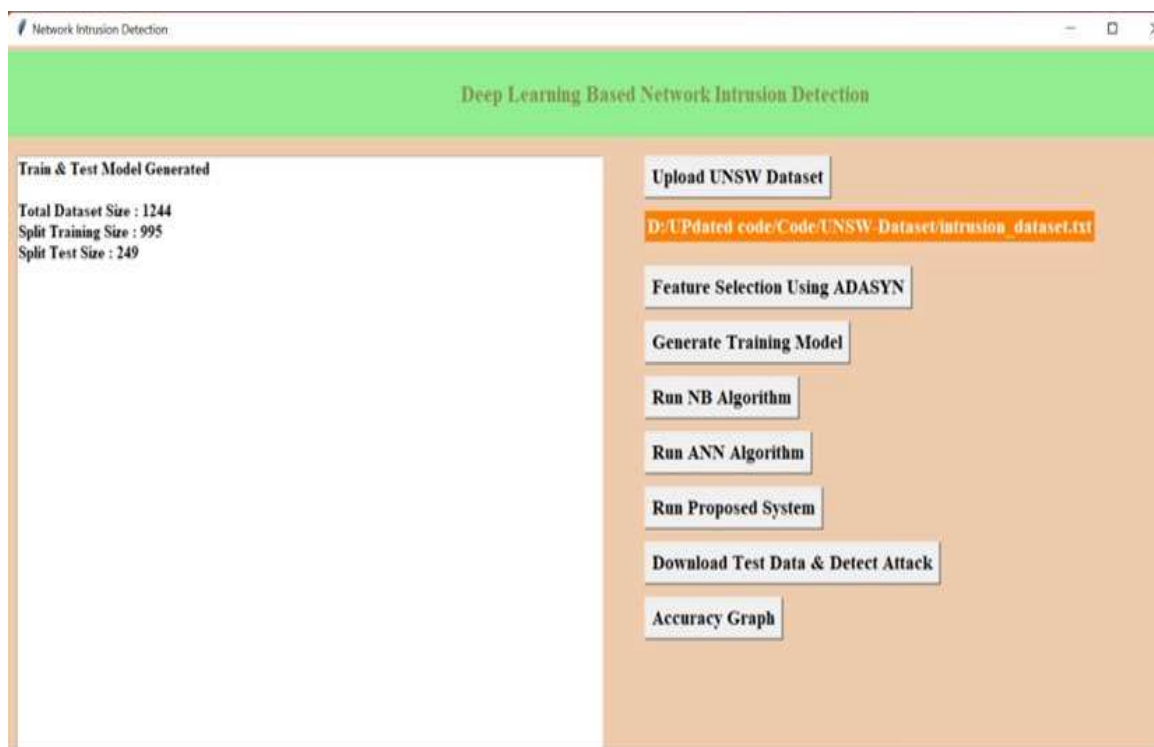


Figure 7.7 Generating Training Model

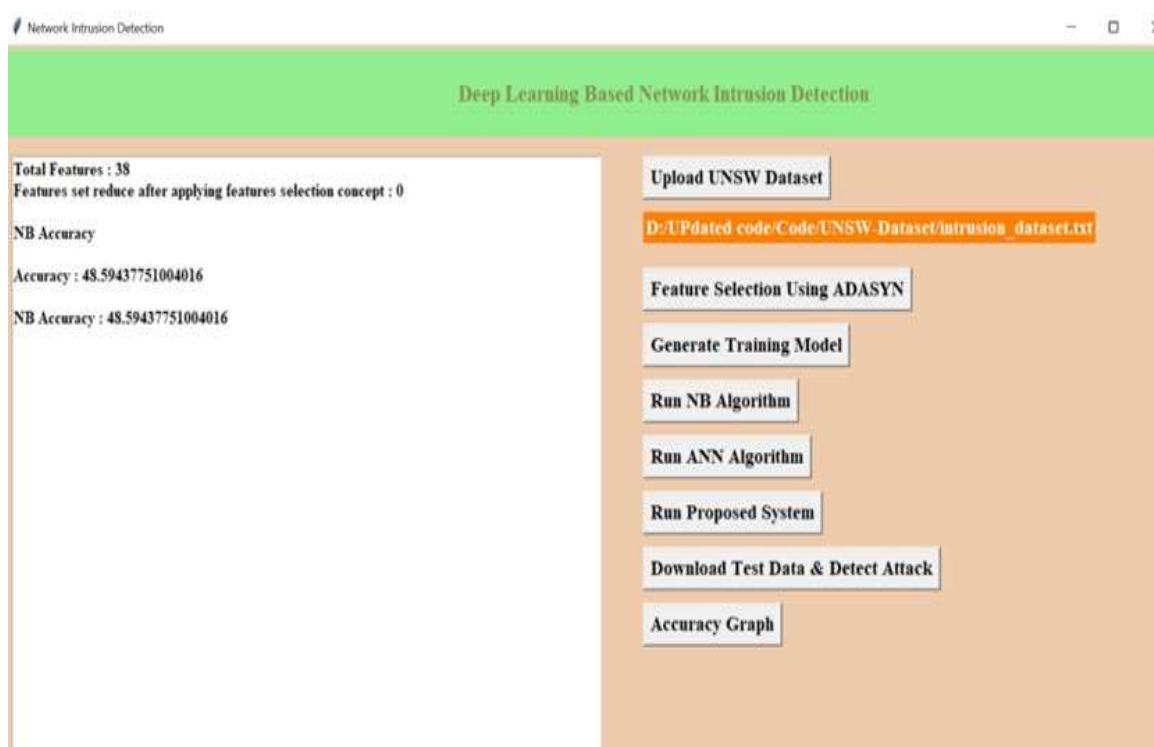


Figure 7.8 Accuracy of NB Algorithm

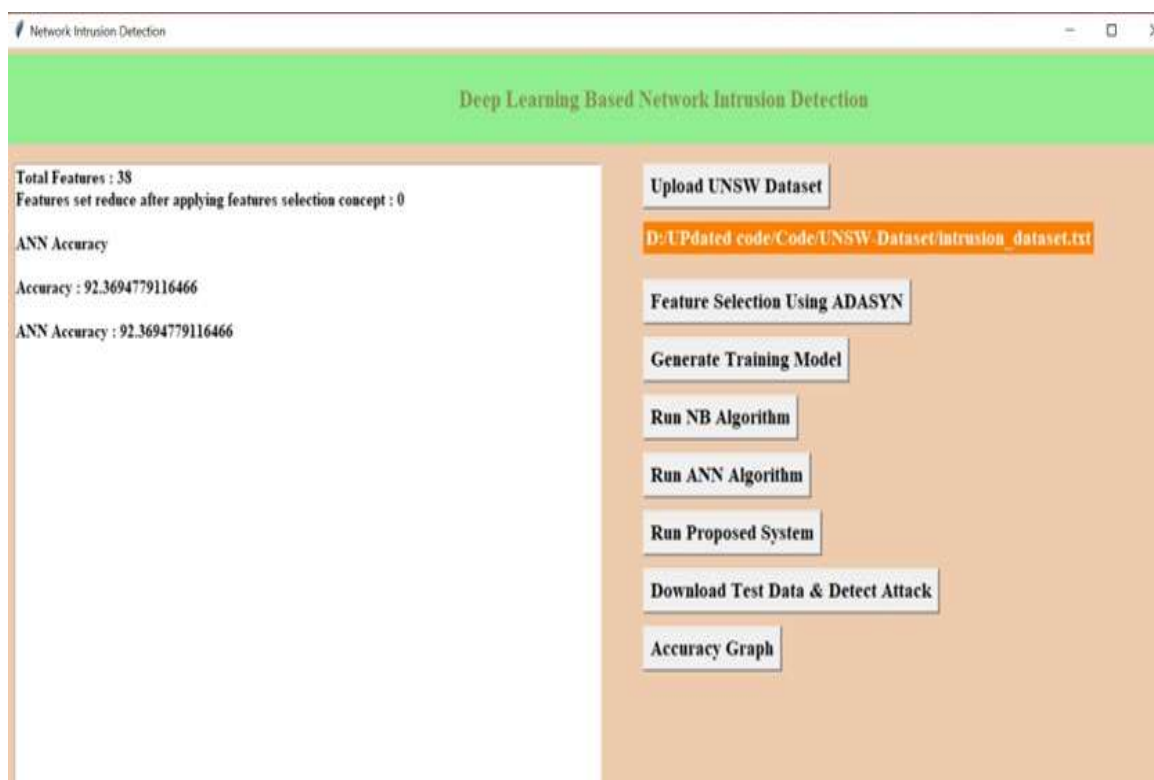


Figure 7.9 Accuracy of ANN Algorithm



Figure 7.10 Accuracy of Proposed Algorithm



Figure 7.11 Downloading the Test Dataset and Detecting the Attack

7.2 Result Analysis

The performance evaluation results are presented in the bar graph, comparing the accuracy of three different algorithms: NB (Naive Bayes) Algorithm, ANN (Artificial Neural Network) Algorithm, and Proposed Algorithm. As we can see from the graph, the Proposed Algorithm showed the highest accuracy among the three models that have been assessed. The Proposed Algorithm outperforms the other two algorithms in terms of accuracy.

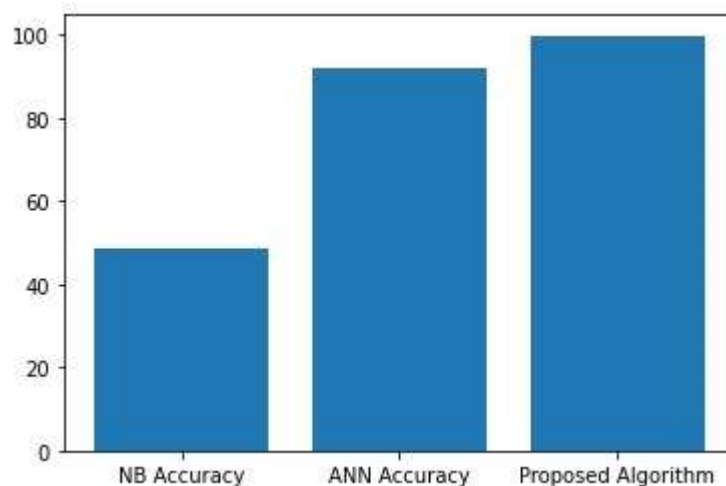


Figure 7.12 Comparing the Accuracy of Different Algorithms

as indicated by its larger bar representation. Although the ANN Algorithm outperformed the NB Algorithm, it lacked the efficiency of the Proposed Algorithm. The algorithm with the NB performance produced the lowest accuracy rate, arrived as the shortest bar in the graph.

7.3 Summary

This Chapter describe the steps involved in the working of the IDS. It starts with the attacker uploading data to the cloud, followed by uploading a separate dataset for comparison. Then, the process involves feature selection and training a model. Then moving ahead we see the comaprison of the accuracy of different algorithms, including a Naive Bayes approach, an Artificial Neural Network approach, and the proposed method itself. Finally, there's a step where the model is used to detect attacks on the uploaded data. Section 7.2, focuses on comparing the effectiveness of these different algorithms, It is a comparison graph that compares the performance of the algorithms that are used for network intrusion detection.

CONCLUSION AND FUTURE ENHANCEMENT

Chapter 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

In conclusion, our examine tackles critical challenges faced by means of intrusion detection systems (IDS), that specialize in records distribution imbalances and interchannel redundancy in current neural network (NN)-based fashions. Our approach aims to reinforce IDS by means of leveraging superior strategies in information balancing, feature extraction, and model architecture. We stress the significance of records balance to save you model bias closer to massive samples even as neglecting smaller ones. The Adaptive Synthetic Sampling Method (ADASYN) correctly addresses this imbalance, making sure a truthful and consultant dataset for schooling. Our split-based ResNet framework introduces key advancements. It enables multiscale characteristic extraction thru more than one convolution phases, enriching network visitors statistics representation. It additionally mitigates interchannel redundancy, enhancing the model's capability to capture nuanced patterns. Furthermore, the incorporation of a soft hobby operation in ResNet enables sensible characteristic utilization, significantly boosting version expressiveness and performance in intrusion detection tasks. Our experimental effects, specifically with the hybrid ResNet model and ADASYN, display brilliant enhancements across diverse assessment criteria. Despite these improvements, optimization for execution efficiency and accuracy for smaller samples stays a focus. Moving forward, we goal to refine our method, that specialize in improving IDS identification skills thru a streamlined residual network structure with more desirable residual blocks. By innovating and integrating superior methodologies, we contribute notably to intrusion detection and cybersecurity.

8.2 Future Enhancement

Analysing the statistics and research provided, several avenues for destiny enhancement in intrusion detection structures (IDS) and community security emerge. Firstly, exploring advanced sampling techniques past ADASYN and RENN may want to show fruitful. Techniques like SMOTE-NC or Borderline-SMOTE offer ability enhancements in dealing with imbalanced datasets and producing artificial samples more effectively. Secondly, endured advancements in deep getting to know architectures which include variations of

ResNet, DenseNet, or EfficientNet can beautify function extraction and getting to know capabilities for complicated community visitors information. Thirdly, ensemble methods like Random Forests or Gradient Boosting Machines may be incorporated to improve version robustness and decrease fake positives. Additionally, growing adaptive characteristic choice algorithms, incorporating real-time stream processing frameworks, integrating explainable AI strategies, and discovering modern techniques for zero-day assault detection are promising areas for boosting IDS talents and addressing rising cybersecurity challenges efficiently. These avenues together goal to make IDS more adaptive, accurate, and responsive in mitigating evolving community threats.

REFERENCES

REFERENCES

- [1] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, “A survey on wireless security: Technical challenges, recent advances, and future trends,” *Proc. IEEE*, vol. 104, no. 9, pp. 1727–1765, Sep. 2016, doi: 10.1109/JPROC.2016.2558521.
- [2] A. Kavianpour and M. C. Anderson, “An overview of wireless network security,” in *Proc. IEEE 4th Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, New York, NY, USA, Jun. 2017, pp. 306–309, doi: 10.1109/CSCloud.2017.45.
- [3] H. Sallay and S. Bourouis, “Intrusion detection alert management for highspeed networks: Current researches and applications,” *Secur. Commun. Netw.*, vol. 8, no. 18, pp. 4362–4372, Dec. 2015.
- [4] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016, doi: 10.1109/COMST.2015.2494502.
- [5] S. Z. Lin, Y. Shi, and Z. Xue, “Character-level intrusion detection based on convolutional neural networks,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Rio de Janeiro, Brazil, Jul. 2018, pp. 1–8, doi: 10.1109/IJCNN.2018.8488987.
- [6] B. Selvakumar and K. Muneeswaran, “Firefly algorithm based feature selection for network intrusion detection,” *Comput. Secur.*, vol. 81, pp. 148–155, Mar. 2019.
- [7] Y. Qi, M. Liu, and Y. Fu, “Research on SVM network intrusion detection based on PCA,” *Inf. Netw. Secur.*, vol. 2, pp. 15–18, Feb. 2015.
- [8] D. Zheng, Z. Hong, N. Wang, and P. Chen, “An improved LDA-based ELM classification for intrusion detection algorithm in IoT application,” *Sensors*, vol. 20, no. 6, p. 1706, Mar. 2020.
- [9] C. Iwendi, S. Khan, J. H. Anajemba, M. Mittal, M. Alenezi, and M. Alazab, “The use of ensemble models for multiple class and binary class classification for improving intrusion detection systems,” *Sensors*, vol. 20, no. 9, p. 2559, Apr. 2020.
- [10] H. Wang, J. Gu, and S. Wang, “An effective intrusion detection framework based on SVM with feature augmentation,” *Knowl.-Based Syst.*, vol. 136, pp. 130–139, Nov. 2017.
- [11] L. Xiao, Y. Chen, and C. K. Chang, “Bayesian model averaging of Bayesian network classifiers for intrusion detection,” in *Proc. IEEE 38th Int. Comput. Softw. Appl. Conf. Workshops*, Västerås, Sweden, Jul. 2014, pp. 128–133, doi: 10.1109/COMPSACW.2014.25.

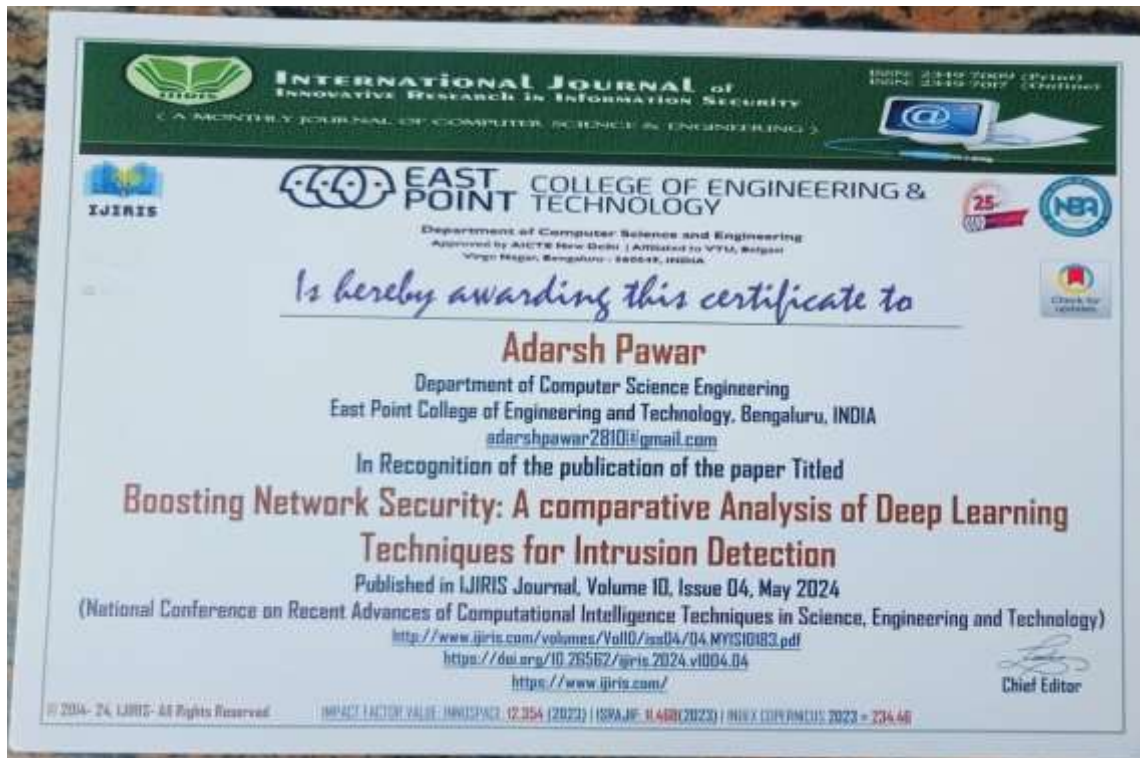
- [12] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2670–2679, Apr. 2015.
- [13] S. S. Roy, A. Mallik, R. Gulati, M. S. Obaidat, and P. V. Krishna, "A deep learning based artificial neural network approach for intrusion detection," in *Proc. Int. Conf. Math. Comput. (ICMC)*, Haldia, India, Jan. 2017, pp. 44–53.
- [14] S. M. H. Bamakan, H. Wang, T. Yingjie, and Y. Shi, "An effective intrusion detection framework based on MCLP/SVM optimized by timevarying chaos particle swarm optimization," *Neurocomputing*, vol. 199, pp. 90–102, Jul. 2016.
- [15] R. R. Reddy, Y. Ramadevi, and K. V. N. Sunitha, "Effective discriminant function for intrusion detection using SVM," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Jaipur, India, Sep. 2016, pp. 1148–1153, doi: 10.1109/ICACCI.2016.7732199.
- [16] L. I. Peng and Z. Wen-Huan, "Mixed intrusion detection algorithm based on k-means and decision tree," *Comput. Modernization*, pp. 12–16, Dec. 2017.
- [17] H. M. Tahir, W. Hasan, A. Said, N. H. Zakaria, N. Katuk, N. F. Kabir, M. H. Omar, O. Ghazali, and N. I. Yahya, "Hybrid machine learning technique for intrusion detection system," in *Proc. 5th Int. Conf. Comput. Informat. (ICOCI)*, Istanbul, Turkey, Aug. 2015, pp. 2289–3784.
- [18] X. Tan, S. Su, Z. Zuo, X. Guo, and X. Sun, "Intrusion detection of UAVs based on the deep belief network optimized by PSO," *Sensors*, vol. 19, no. 24, p. 5529, Dec. 2019.
- [19] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, "Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark," *IEEE Access*, vol. 6, pp. 59657–59671, 2018, doi: 10.1109/ACCESS.2018.2875045.
- [20] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/ACCESS.2017.2762418.
- [21] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Da Nang, Vietnam, 2017, pp. 712–717, doi: 10.1109/ICOIN.2017.7899588.
- [22] W. Ming and L. Jian, "Network intrusion detection model based on convolutional neural network," *J. Inf. Secur. Res.*, pp. 990–994, Nov. 2017.

- [23] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50850–50859, 2018, doi: 10.1109/ACCESS.2018.2868993.
- [24] P.-F. Wu and H.-J. Shen, "The research and amelioration of patternmatching algorithm in intrusion detection system," in *Proc. IEEE 14th Int. Conf. High Perform. Comput. Commun. IEEE 9th Int. Conf. Embedded Softw. Syst.*, Liverpool, U.K., Jun. 2012, pp. 1712–1715, doi: 10.1109/HPCC.2012.256.
- [25] V. Dagar, V. Prakash, and T. Bhatia, "Analysis of pattern matching algorithms in network intrusion detection systems," in *Proc. Int. Conf. Adv. Comput.*, 2016, pp. 1–5.
- [26] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," *J. King Saud Univ.- Comput. Inf. Sci.*, vol. 29, no. 4, pp. 462–472, Oct. 2017.
- [27] B. Ingre, A. Yadav, and A. K. Soni, "Decision tree based intrusion detection system for NSL-KDD dataset," in *Proc. Int. Conf. Inf. Commun. Technol. Intell. Syst.*, Ahmedabad, India, 2017, pp. 207–218, doi: 10.1007/978-3-319-63645-0_23.
- [28] P. Nancy, S. Muthurajkumar, S. Ganapathy, S. V. N. S. Kumar, M. Selvi, and K. Arputharaj, "Intrusion detection using dynamic feature selection and fuzzy temporal decision tree classification for wireless sensor networks," *IET Commun.*, vol. 14, no. 5, pp. 888–895, Mar. 2020, doi: 10.1049/iet-com.2019.0172.
- [29] M. A. Jabbar, R. Aluvalu, and S. S. Satyanarayana Reddy, "Intrusion detection system using Bayesian network and feature subset selection," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, Coimbatore, India, Dec. 2017, pp. 1–5, doi: 10.1109/ICCIC.2017.8524381.
- [30] T.-T.-H. Le, J. Kim, and H. Kim, "An effective intrusion detection classifier using long short-term memory with gradient descent optimization," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, Busan, South Korea, Feb. 2017, pp. 1–6, doi: 10.1109/PlatCon.2017.7883684.
- [31] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020, doi: 10.1109/ACCESS.2020.2972627.
- [32] P. Wei, Y. Li, Z. Zhang, T. Hu, Z. Li, and D. Liu, "An optimization method for intrusion detection classification model based on deep belief network," *IEEE Access*, vol. 7, pp. 87593–87605, 2019, doi: 10.1109/ACCESS.2019.2925828.

- [33] M. Gao, L. Ma, H. Liu, Z. Zhang, Z. Ning, and J. Xu, "Malicious network traffic detection based on deep neural networks and association analysis," *Sensors*, vol. 20, no. 5, p. 1452, Mar. 2020.
- [34] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Udupi, India, Sep. 2017, pp. 1222–1228, doi: 10.1109/ICACCI.2017.8126009.
- [35] Y. Ding and Y. Zhai, "Intrusion detection system for NSL-KDD dataset using convolutional neural networks," in *Proc. 2nd Int. Conf. Comput. Sci. Artif. Intell. (CSAI)*, 2018, pp. 81–85.
- [36] H. Yang and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," *IEEE Access*, vol. 7, pp. 64366–64374, 2019, doi: 10.1109/ACCESS.2019.2917299.
- [37] Q. Zhang, Z. Jiang, Q. Lu, J. Han, Z. Zeng, S.-H. Gao, and A. Men, "Split to be slim: An overlooked redundancy in vanilla convolution," 2020, arXiv:2006.12085. [Online]. Available: <http://arxiv.org/abs/2006.12085>
- [38] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446–452, 2015.

CERTIFICATES

PUBLICATION CERTIFICATE





PLAGIARISM CERTIFICATE

East point College point college of Engineering and Technology

Certificate of Plagiarism Check for Synopsis

Author Name	Adarsh
Course of Study	B.E
Name of Guide	Dr. Heena Kousar
Department	Computer Science and Engineering
Acceptable Maximum Limit	30%
Submitted By	pavankumarb@hotmail.com
Paper Title	Boosting Network Security: A Comparative Analysis of Deep Learning Techniques for Intrusion Detection
Similarity	23%
Paper ID	1843310
Submission Date	2024-05-21 11:58:51

 Signature of Student

 Signature of Guide

 Head of the Department

Professor & Head
Dept. of Computer Science Engineering
East Point College of Engineering & Technology
Bangalore - 560 043.

* This report has been generated by DrillBit Anti-Plagiarism Software

